清 华 大 学

攻 读 工 学 硕 士 学 位

研究生选题报告

论文题目 基于推理的面向不可见任务与非自由选择结构的过程挖掘算法研究

研 究 生 郭秦龙 学 号 2012212856

院（系、所） 软件学院 专 业 软件工程领域

联 系 电 话 15120003381  电 子 邮 件 guoqinlong@gmail.com

指 导 教 师 王建民 专业技术职务 教授

联 系 电 话 62781776 电 子 邮 件 jimwang@tsinghua.edu.cn

联合指导教师 专业技术职务

联 系 电 话 电 子 邮 件

入 学 日 期 2013年9月

2013 年 9 月 25 日

目录

[1. 选题背景 2](#_Toc367904937)

[1.1背景 2](#_Toc367904938)

[1.2选题意义 3](#_Toc367904939)

[2. 国内外研究现状 4](#_Toc367904940)

[2.1几类典型的挖掘算法 4](#_Toc367904941)

[2.1.1 Alpha算法 4](#_Toc367904942)

[2.2.2 Heuristics算法 4](#_Toc367904943)

[2.2.3 Genetic 算法 5](#_Toc367904944)

[2.2.4 Region算法 6](#_Toc367904945)

[2.2 基于推理的过程挖掘算法 6](#_Toc367904946)

[2.2.1 Alpha++算法 6](#_Toc367904947)

[2.2.2Alpha#算法 11](#_Toc367904948)

[3.研究内容 15](#_Toc367904949)

[3.1研究的新算法实现方法 15](#_Toc367904950)

[3.1.1.研究不可见任务与非自由选择结构结合情况的分类 15](#_Toc367904951)

[3.1.2Alpha++算法与Alpha#算法流程融合 16](#_Toc367904952)

[3.1.3.研究扩展各自挖掘算法 16](#_Toc367904953)

[3.2研究新算法的正确性与挖掘能力 16](#_Toc367904954)

[4.研究方案 17](#_Toc367904955)

[4.1新算法具体实现方法的研究方案 18](#_Toc367904956)

[4.2不可见任务与非自由选择结构结合情况的分类的研究方案 18](#_Toc367904957)

[4.3Alpha++算法与Alpha#算法流程融合的研究方案 19](#_Toc367904958)

[4.4新算法正确性与挖掘能力的研究方案 19](#_Toc367904959)

[5.预期成果及可能的创新点 19](#_Toc367904960)

[6.研究计划 19](#_Toc367904961)

[致谢 20](#_Toc367904962)

[参考文献 21](#_Toc367904963)

1. 选题背景

1.1背景

随着工作流管理技术的飞速发展，为此， 越来越多的企业引入各种过程感知信息系统来支持业务过程[1]，包括WfMS[2-4](如Staffware[5],IBM MQSeries[6]、COSA[7]）CRM软件，ERP软件（SAP[8],Oracle[9]）。也有许多高校研发了工作流相关系统如BeehiveZ[10-11],ProM[12]。这些系统都可以有效支持业务过程的建模、分析、仿真和优化。

过程感知系统同一般是由过程模型来驱动，然而创建过程模型是一项复杂、耗时的工作，而且得到的过程模型往往是主观的，不完全的或过于抽象的或者与实际流程存在偏差，同时企业业务不断变更要求模型也进行相应的变更。除此之外，目前有不少企业为了更好的提升服务质量，根据企业的过程日志文件进行过程挖掘，产生了很多业务过程模型，从而更好的分析用户行为特点。过程挖掘就是为了解决以上困难而提出的。

过程挖掘的基本思想在1995年由美国新墨西哥州立大学的Cook教授提出[13]，目标是从软件过程的事件日志中自动发现过程模型，并随之做了大量的后继工作，作者称该技术为过程发现。算法采用有限状态自动机来表达过程模型，能够处理并行结构和噪声。1998年美国IBM艾曼登研究中心的IBM院士Agrawal将其引入业务过程领域，并正式命名为过程挖掘。[14]

目前国际上一种比较认可的过程挖掘定义为“过程挖掘是指那些从实际执行集合中提取出结构化过程描述的方法[15]”。

过程挖掘宣言[16]中将过程挖掘定义为“过程挖掘技术能够从现代信息系统普遍产生的事件日志中抽取信息，该技术为各种应用领域中的过程发现、监测和改进提供了新的手段。”

过程挖掘可以实现三种功能：发现、符合性检查、增强[17]。图1-1中展示了三种过程挖掘的输入输出。发现是指利用事件日志进行挖掘，通过挖掘，获得过程模型。可以看到图中，发现的输入是事件日志，输出是过程模型。符合性检查，是指通过给定的事件日志和模型，判断模型和事件日志的记录是否相符合，可以看到在图中，符合性检查的输入是事件日志和模型，输出是诊断结果。第三种结果是增强，增强是对于模型和模型在实际应用中产生的日志，对模型进行改进，提高模型执行的效率。在图中可以看到，增强功能的输入是事件日志和模型，输出是一个新的模型。

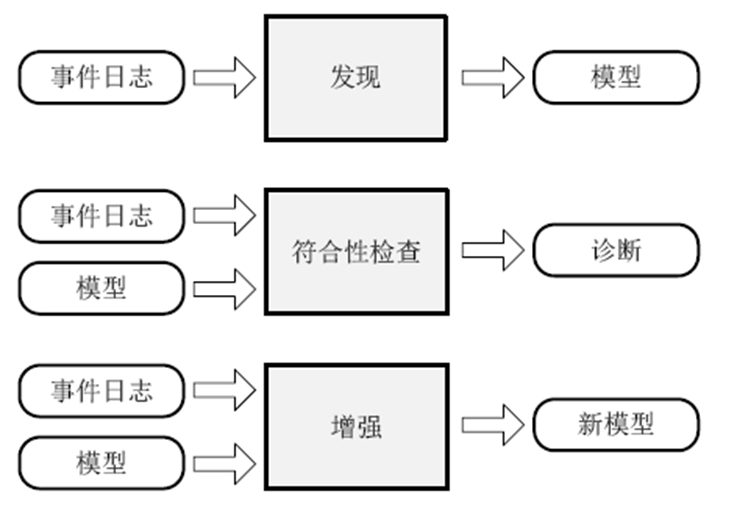
W.M.P. van.der Aalst和他的研究团队在工作流和过程挖掘领域做了广泛深入的研究[18-22]，他们的工作几乎涉及过程挖掘领域的各个方面，提出了经典的Alpha算法[23]及大量的改进算法。Alpha算法从事件间的局部依赖入手，挖掘一个基于Petri网的工作流模型。充分讨论了Alpha算法发现能力问题，并给出相关的证明。正是因为Alpha算法的挖掘能力不足但却具有严格的分析证明，因而跟随进研究比较多，清华大学的闻立杰老师等人针对Alpha算法处理非自由选择结的不足提出了改进的Alpha++[23]算法,针对处理不可见任务不足提出了Alpha#算法[24]等等。

图 1.1 三种过程挖掘的输入输出

1.2选题意义

在上面的背景中可以知道，随着工作流技术的飞速发展与工作流系统的逐渐相继问世，过程挖掘技术为了解决工作流系统中模型而诞生。同时，过程挖掘算法具有*发现*，*符合性检查*与*增强*三种作用。在本课题中，研究一种基于推理的面向不可见任务与非自由选择结构的过程挖掘算法，将同样含有过程挖掘上述作用。除此之外，本课题的研究将主要有以下两方面的意义：

1.设计一种新型的过程挖掘算法：新过程挖掘算法可以挖掘同时含有不可见任务和非自有选择结构的过程模型，将会是目前Alpha系列算法中第一个支持同时挖掘出这两种结构的过程挖掘算法。

2.算法挖掘能力证明：新算法是基于推理来实现的，可以通过形式化的方法来证明算法的正确性，并且对于算法的挖掘能力进行形式化的分析。

2. 国内外研究现状

本章重点介绍国内外研究中的过程挖掘算法，2.1节介绍几类比较典型的过程挖掘算法；2.2重点介绍两个基于推理的过程挖掘算法：Alpha++与Alpha#。

2.1几类典型的挖掘算法

#### 2.1.1 Alpha算法

Alpha算法[25]关注的重点在于事件日志中不同活动之间的因果关系。Alpha算法根据事件日志中不同出现的先后顺序定义了活动间四种不同的关系。不妨以a, b表示事件日志中的两个不同的活动，那么四种关系分别是：

1.

2. ;

3.;

4..

根据上述的四种定义的四种关系，Alpha算法找出所有满足如下关系的集合:

|  |  |  |
| --- | --- | --- |
|  |  | 2-1 |

即集合中每个元素是二元组(A,B)，其中A和B都是日志中活动的集合。A和B中的每个元素都满足“”关系，而A中每个元素和B中每个元素都满足“”关系。在获取集合后，其中中每一个最大的元素都对应挖掘后模型的库所，而事件日志中的每个活动对应挖掘后过程模型的变迁。根据中的关系，可以确定挖掘后模型的每一条边，进而获取挖掘以后的过程模型。

#### 2.2.2 Heuristics算法

Heuristics算法 [26]挖掘得到的过程模型是以Casual Net[27]的形式保存的，Casual Net可以通过一定的规则转换成为Petri网。Heuristics 算法计算过程中考虑了事件日志中每一个事件轨迹的频率，这样在一定程度上可以排除日志中噪声对挖掘结果的影响。Heuristics算法流程主要分为两个步骤。

第一步，Heuristics算法将事物日志转换为中间结构Dependency Graph。Dependency Graph是一种图的数据结构，直接反映了事件日志中活动间的拓扑结构。即图中的点是事物日志中的活动，图中的边代表日志中活动间的顺联关系。通过如下的两个标准，Heuristics算法在依赖图中剔除部分的边，对于事件日志进行降噪：

|  |  |  |
| --- | --- | --- |
|  |  | 2-2 |
|  | | | 2-3 |

获得事件日志对应的依赖图后，Heuristics算法第二步将依赖图转换为Casual Net. Casual Net 是一个六元组(。其中A代表活动的集合，在事件日志中已经有所定义。代表模型的初始活动和终结活动，可以通过人为进行设置。D指活动间的依赖关系，这一部分可以通过算法上一步生成的依赖图直接获得。I和O分别指每个活动的前驱活动集合和后继活动结合。根据Casual Net的定义，我们发现组成Casual Net的六元组中，只有I和O是在需要确定的。在这一步骤中，Heuristics算法再一次考虑事件日志中的每个事件轨迹的出现频率，对于每个活动，算法仅将发生了具有一定频率的事件轨迹中的前后顺序的活动添加到I和O集合。当Casual Net中每个元素都确定了以后，算法结束，生成的Casual Net即为事件日志的挖掘结果。

#### 2.2.3 Genetic 算法

Genetic[28]算法整合了遗传算法的流程，可以进行全局的搜索，并且可以在一定程度上避免事件日志噪声的影响。而且，通过设置遗传过程中的选择函数，Genetic算法可以对于挖掘以后的结果进行偏好导向。与遗传算法流程类似，Genetic过程挖掘算法主要有如下的几步：

Genetic算法首先根据事件日志构造初始种群。初始种群的构造有两种方法：第一种是根据事件日志中的活动，随机的生成若干过程模型作为初始种群。第二种方法是根据事件日志，利用一些诸如Heuristics之类高效率的挖掘算法，产生若干过程模型，作为初始种群。第二种方法相对于第一种方法可以一定程度上减少后续的迭代中的次数。

在获取初始种群后，算法的第二步是个体评价，即计算种群中每个过程模型的适应度。过程模型的适应度是通过计算事件日志与模型之间的拟合程度。即适应度计算的是事件日志中有多少事件轨迹可以由模型所表达，模型又会表达出多少事件日志所没有的事件轨迹。

算法的第三步是通过选择运算，交叉运算和变异运算生成下一代种群。选择运算指在个体评价中，选择适应度较高的部分过程模型直接保留到下一代。交叉运算是指将两个或两个以上的过程模型进行重新组合，组成一个新的模型。变异运算是指在过程模型内部修改，生成新的模型。

在获得了下一代的种群后算法重复第二步和第三步，计算每一代种群的适应度，直到找到适应度最大的过程模型后停止算法，并将适应度最大的模型作为结果输出。

算法通过进行全局搜索可以获得最优的过程模型。但是考虑到效率的问题，有时算法并不会以找到适应度最大的模型的方式结束。一种结束算法的方法是在生成给定个数的过程模型后结束，并在生成的所有模型中挑选最适合的模型。另一种结束算法的方法是在满足连续若干代种群的最大适应度呈下降趋势后结束算法。

#### 2.2.4 Region算法

Region[29]算法通过使用变迁系统[30]作为算法的中间结构的挖掘算法。因此算法包含了变迁系统的优点，如可以充分反映事件日志中活动间的关系，又如Region算法可以避免“状态爆炸”，即控制挖掘后的过程模型结构不会很庞大。Region算法主要分为如下两个步骤。

首先，Region算法从事件日志中提取变迁系统。变迁系统是一个三元组(S, A, T),其中S指状态的集合，A指事件中活动的集合。T是集合的子集，即，代表一个状态经过活动A后变成另一个状态。其中事件日志中执行到的每一步对应一个变迁系统的状态，而变迁系统的状态根据可以对应若干事件日志执行的步骤。根据变迁系统的定义，算法这一步骤是将读取事件日志，构造所有的状态，并且根据状态间的变换关系，确定变迁集合T。

在获取变迁系统以后，算法的第2步是从变迁系统中获得过程模型。在本算法中，过程模型是用Petri网存储的。因为在第一步确定变迁系统时，变迁已经与事件日志中的活动相对应了。所以这一步主要确定Petri网中的库所和边。在算法的这一步中，提出了区域的概念。区域是指变迁系统中若干库所构成的集合，其中，变迁系统中所有相同的变迁要么进入该集合，要么从该集合出去。每一个区域对应挖掘以后的过程模型的库所，变迁系统中变迁与区域的连接关系对应过程模型中的每一条边。因此算法得到了挖掘以后的过程模型。

2.2 基于推理的过程挖掘算法

基于推理的过程挖掘算法是指通过分析事件日志中的任务间关联关系，来构建出符合事件日志的过程模型的方法。如2.1.1节中的Alpha算法便是一个典型的基于推理的过程挖掘算法。针对Alpha算法无法处理工作流模型中含有非自由选择结构和不可见任务的情况，Alpha++算法与Alpha#算法分别应对上述问题。

#### 2.2.1 Alpha++算法

Alpha++算法中将工作流模型中任务的依赖关系分为了两类：显性依赖关系(Explicit Dependency)和间接依赖关系(Implicit Dependency)，并且通过找寻事件间的间接依赖关系来挖掘工作流模型中的非自由选择结构。

**定义1**(*显性依赖*) 是一个工作流网，那么对于任意两个变迁，a，b之间存在一个显性依赖关系当且仅当以下两个条件满足：

连接性：

连续性：存在一个标识s，,满足且。

**定义2**(*间接依赖*) 是一个合理的工作流网，那么对于任意两个变迁，a，b之间存在一个间接依赖关系当且仅当以下三个条件满足：

连接性：

不连续性：不存在一个标识s,,满足且

可达性：存在一个标号s，满足, 并且另外一个标号>,满足

如以图2.1所示，T1与T3，T2与T3均为显性依赖。而T1与T4之间，T2与T5之间则是间接依赖。如果一个模型中的仅含有显性依赖关系，那么这个模型中就不含有非自有选择结构，大多数过程挖掘算法都可以将模型正确的挖掘出来。而当工作流模型中含有间接依赖关系时，则这个模型中含有非自有选择结构，绝大部分过程挖掘算法不可以将正确的模型挖掘出来，这正是Alpha++算法所解决的问题。

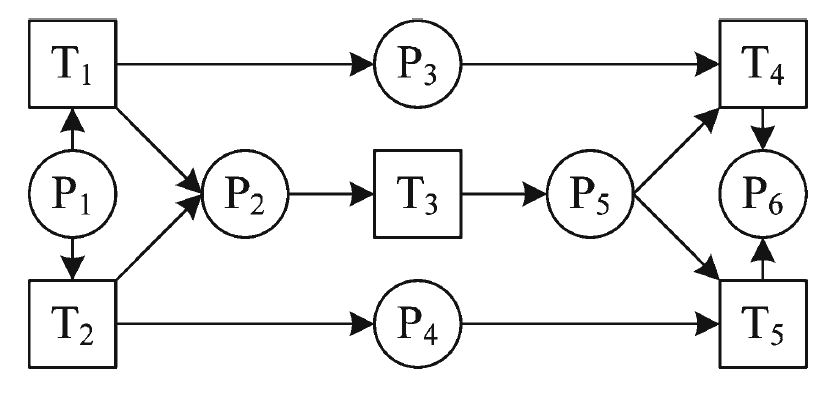


图2.1一个同时具有显性与间接依赖的模型

Alpha++算法中利用符号代表活动间的间接依赖关系。Alpha++算法沿用Alpha算法中关于活动间的关系四种关系的定义（即,,和），并且为了能够计算活动间间接依赖，Alpha算法添加了表2.1任务间的关系

表2.1 Alpha++添加的事件任务关系

|  |  |
| --- | --- |
|  | 当且仅当存在一个事件轨迹且,使得,且,且 |
|  | 当且仅当，且存在一个任务,q且满足与 |
|  | 当且仅当，且存在一个任务,q且满足与 |
|  | 当且仅当，且存在一个事件轨迹 ,满足,,,且,有, |
|  | 当且仅当或 |

在定义了上述五种任务间的关系后，Alpha++算法指出了三种计算间接依赖的方法。

表2.2 Alpha++中三种关系

|  |  |
| --- | --- |
|  | 当且仅当,且存在一个任务,和两个不同的库所,其中,且a,a, b,b, 并且不存在任务,满足或者 |
|  | 当且仅当,|a|>1且存在一个任务满足且存在一个库所满足不存在任务,或，存在一个任务满足或 |
|  | 当且仅当,||>1且存在一个任务满足且存在一个库所满足不存在任务,或，存在一个任务满足或 |
|  | 当且仅当或 |
|  | 当且仅当存在两个任务,满足，,,,,且 |

第一种间接依赖对应的是如图2.2所示的情况：任务t包含两个入库所p1和p2。由于存在，所以，存在一个事件轨迹，,且有, 。当a执行以后，p1中含有一个令牌，t进入使能态还需要p2也拥有一个令牌。如果有 ,那么a’会在a执行之前执行，这样的话存在一种情况，即a执行以后p2中已经含有了一个令牌。因此，当任务a执行完成以后，t处于使能状态。同理，如果,也存在任务a执行完毕后t处于使能状态的情况。反之，如果且，那么t不可能在a执行以后直接执行，与矛盾。因此，需要在a与p2之间存在一条边（如虚线所示），即a与b’存在间接依赖关系（）。

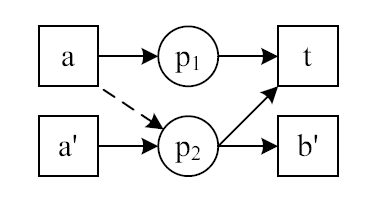


图2.2 所对应的模型片段样例

第二种间接依赖对应的是如图2.3所示的情况（以第一种情况为例，第二种情况类似。）：当任务t执行完毕以后，p1与p2各有一个令牌。因此，存在一种情况是p3中含有一个令牌，a，b同时处于使能状态。但是如果此时y2或者y2’执行，那么根据条件a不应该处于使能状态。同理，当a执行时，y2与y2’不应该执行，那么p2中就一直含有一个令牌，这样就出现了矛盾情况。因此需要在p2与a之间连接一条边（如虚线所示），即t与a之间存在一条间接依赖，即.

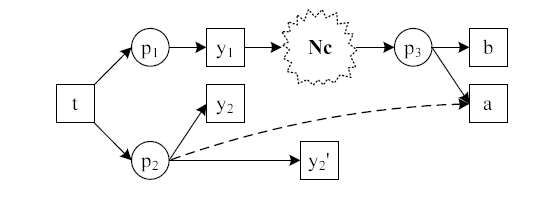


图2.3 所对应的模型片段样例

第三种间接依赖对应的是如图2.4所示的情况:我们考虑最简单的情况，其他情况可以转换成为这种情况，当任务a执行以后，p1中存在一个令牌。过了一段时间以后,p3中存在一个令牌。因此ai是处于使能状态的，bj不处于使能状态。如果任务b执行，那么库所p1和p2中各含有一个令牌。这样ai与bj均处于使能状态。但是这于出现矛盾。但是由于，a与ai之间应该有一个库所相连。（如图中虚线所示），即a与ai存在间接依赖，即。

Alpha++算法在找到三种间接依赖关系后，将间接依赖关系认为是普通的依赖关系“”并如Alpha算法中的公式2-1来找寻变迁、库所集合，并将找到集合转换为对应的工作流模型。

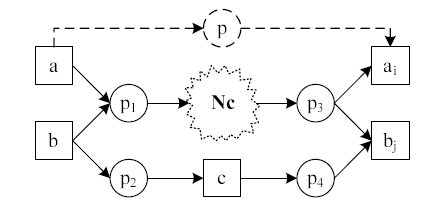


图2.4 所对应的模型片段样例

尽管Alpha++算法可以在大多数情况下挖掘出事件日志对应的工作流模型，但是一些类型的工作流模型，Alpha++算法却无法正确的挖掘出来：

1. 不可见任务：在Alpha++算法中没有处理不可见任务，因此不可见任务不可以挖掘出来。如图2.5所示的模型中含有不可见任务不可以挖掘出来。

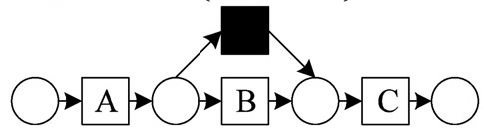


图2.5 一个含有不可见任务的模型

1. 长度为一的循环与间接依赖相结合：Alpha++算法中没有处理长度为1的循环与不可见任务相结合的情况，因此此类不可见任务不可以挖出来。如图2.6中所示的模型中，A、D与D、E之间均有间接依赖，但是因为D在长度为一的循环中，因此这个模型Alpha++算法无法挖掘出来。

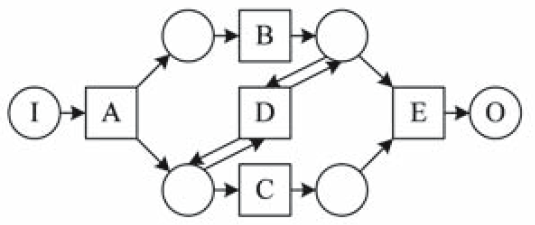


图2.6 一个长度为1的循环与间接依赖结合的模型

#### 2.2.2Alpha#算法

不可见任务(invisible task)是工作流模型中一类常见的结构元素，由一种不含有具体任务信息的变迁来表示。不可见任务与其相连的边表达了工作流网中的拓扑结构信息。Alpha算法无法处理含有不可见任务的模型，Alpha#算法中根据不可见任务的功能，将其分为了如下五个种类：

**Skip**

令是一个起始库所为i,终止库所为o的含有不可见任务工作流模型，对于任意的,t是SKIP类型的SKIP类型的不可见任务，当且仅当存在一个基本路径,满足,且如果,那么t是SHORT-SKIP类型，否则如果(),t是LONG-SKIP类型。

图2.7和图2.8分别展示了SHORT-SKIP和LONG-SKIP这两种类型的不可见任务。SKIP类型的不可见任务表达了工作流模型中省略执行一部分任务的语义。

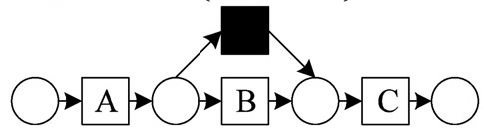


图2.7 的过程模型

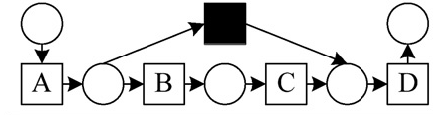


图2.8 图2.7 的过程模型

**Redo**

令是一个含有不可见任务工作流模型。对于任意的,t是REDO类型的不可见任务，当且仅当存在一条基本路径，满足。如果，那么是SHORTR-REDO类型，否则()，是LONG-REDO类型。

图2.8和图2.9分别是SHORT-REDO和LONG-REDO类型的不可见任务。REDO类型的不可见任务表达了工作流模型中一部分任务重复执行的语义信息。

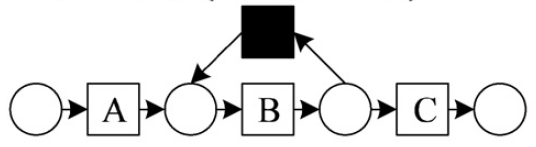


图2.8 的过程模型

**Switch**

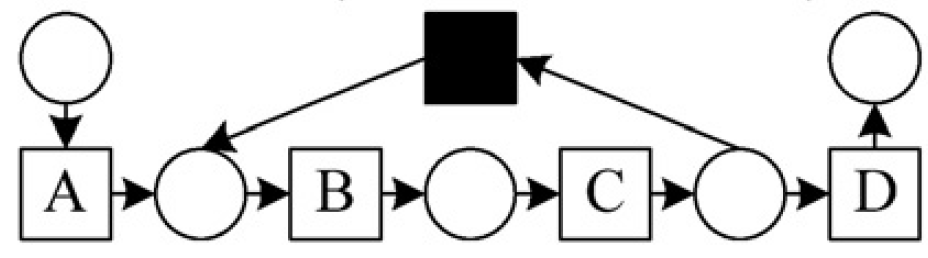


图2.9 的过程模型

令是一个含有不可见任务的工作流模型。于任意的,t是SWITCH类型的不可见任务，当且仅当存在两条基本路径和，满足，且没有其他从到的基本路径了。

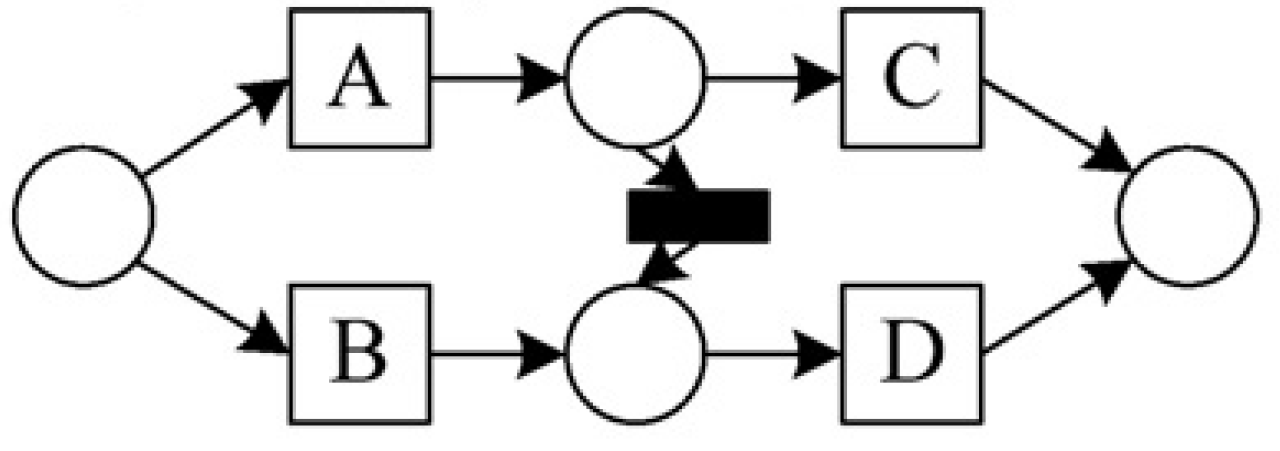


图2.9 的过程模型

图2.9是一种SWITCH类型的不可见任务，这种类型的不可见任务表达了多个执行的分支中可以进行跳转的语义。

**Initialize**

令是一个含有不可见任务的工作流模型，起始库所为。对于任意，t是一个INITIALIZE类型的不可见任务，当且仅当

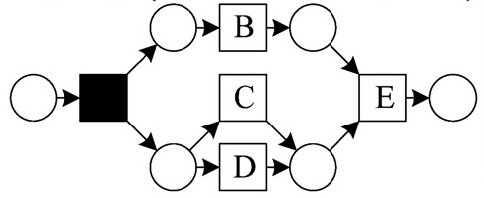


图2.10 的过程模型

图2.10是一种INITIALIZE类型的不可见任务，在开始库所后有一个不可见任务引出两个任务分支。

**Finalize**

令是一个含有不可见任务的工作流模型，结束库所为o。对于任意，t是一个FINALIZE类型的不可见任务，当且仅当

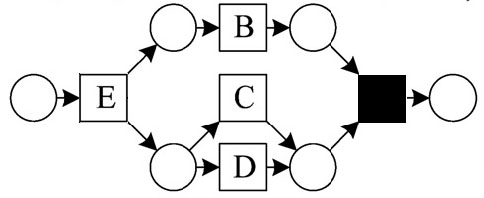


图2.11 的过程模型

Alpha#与Alpha++算法类似，除了隐形依赖的以外，引入了如下的任务间关系,,,,,

同时，针对不可见任务，Alpha#算法引入了任务间的虚假依赖关系(mendacious dependency)，并将这种依赖依赖关系与不可见任务进行了对应。

在Alpha#算法中,通过如下的规则来发现关系：令是一个含有不可见任务的工作流模型，是N的事件日志。

**a**当且仅当:

*.*

在图2.12所示的工作流模型片段中，存在一个不可见任务t。如果变迁y与变迁x相同，那么变迁t是一个SHORT-SKIP类型的不可见任务。如果变迁X可达变迁Y，那么变迁t是一个LONG-SKIP类型的不可见任务。如果变迁a与b相同，那么变迁t是一个SHORT-REDO类型的不可见任务。如果变迁b可达变迁a，那么变迁t是一个LONG-REDO类型的不可见任务。如果上述的情况都不存在，那么变迁t是一个SWITCH类型的不可见任务。

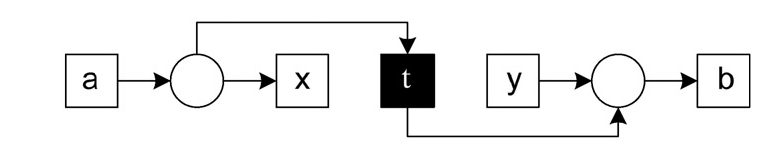


图2.12 的模型片段

Alpha#算法在找到虚假依赖关系后，将虚假依赖关系认为是普通的依赖关系“”并如Alpha算法中的公式2-1来找寻变迁、库所集合，并将找到集合转换为对应的工作流模型。

尽管Alpha#算法可以在大多数情况下挖掘出事件日志对应的工作流模型，但是一些类型的工作流模型，Alpha#算法却无法正确的挖掘出来：

1. 非自由选择结构：如图2.13所示，由于e与不可见任务之间含有非自有选择结构，不可见任务无法直接在a执行以后就执行，导致挖掘不成功。

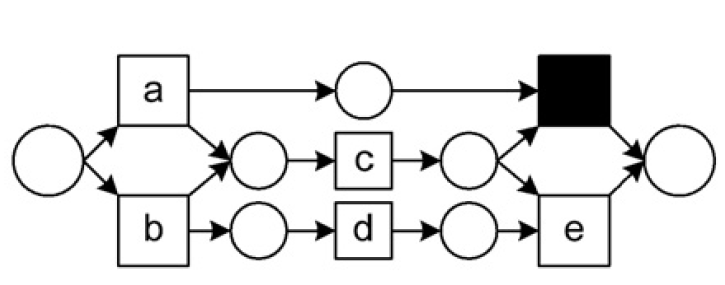


图2.13 一个同时含有非自由选择结构与不可见任务结构的过程模型

1. 不可见任务作为并行的一个分支的情况：如图2.14所示，在任务a与任务d之间含有一个不可见任务，但是在该模型的事件日志中，由于C的存在，a无法与d直接相连，因此该不可见任务无法被发现。

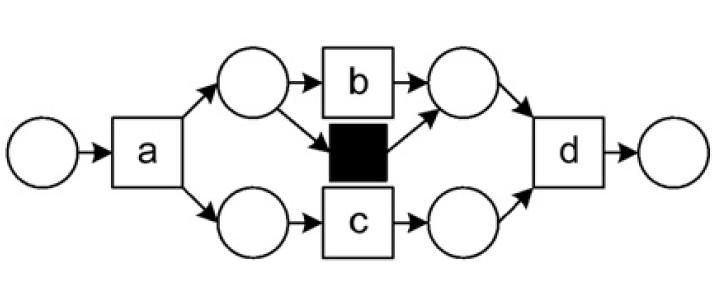


图2.14 一个不可见任务在并行结构中的一个分支的过程模型

3.研究内容

实现基于推理规则的面向不可见任务与非自由选择结构的过程挖掘算法,需要分析现有基于推理规则的挖掘算法的实现方法,还需要了解这些挖掘算法的挖掘能力。本课题是以Alpha++算法与Alpha#算法为切入点，研究两个算法的融合，最后实现具备挖掘不可见任务与非自有选择结构能力的过程挖掘算法。在今后的工作中，具体的研究内容如下：

3.1研究的新算法实现方法

#### 3.1.1.研究不可见任务与非自由选择结构结合情况的分类

在Alpha++算法与Alpha#算法中，分别对于工作流模型中含有非自有选择结构和不可见任务的形式进行了分类。（如在Alpha#算法中，将不可见任务分成了INITIALIZE,FINALIZE, SKIP,REDO,SWITCH等五种情况）。针对结构的分类可以细化挖掘算法所需要解决的问题，为算法设计提供思路。因此在设计新算法阶段，需要参考非自有选择结构与不可见任务分类的工作，重点研究工作流模型中同时含有这两种结构的各种不同情况（如图2.13和图3.1中列举出两个模型片段，分别是不可见任务与非自有选择结构不同的结合方式），为细化新算法的设计提供基础。

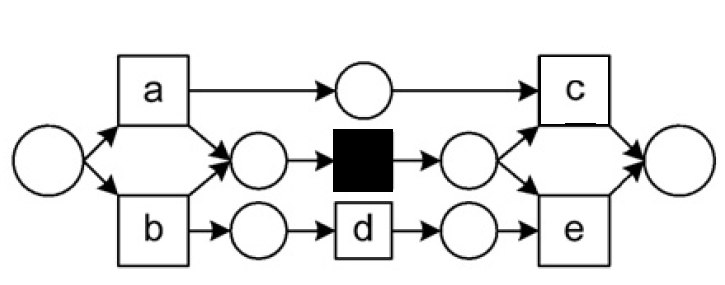


图3.1 一个同时含有非自由选择结构与不可见任务的过程模型

#### 3.1.2Alpha++算法与Alpha#算法流程融合

Alpha++算法与Alpha#的算法流程相似，均分为三个阶段：抽象，归纳和构造。而两个模型由于处理了不同的任务间关系（隐形依赖和虚假依赖），导致两个过程挖掘算法的挖掘能力上的不同。在新算法的设计中，因为要能够同时支持非自由选择结构和不可见任务的挖掘，需要重点研究如何将Alpha++算法与Alpha#算法的流程进行融合，公用两个算法中相同的部分，同时也将两个算法中不同的部分有机的融合为一个整体。

#### 3.1.3.研究扩展各自挖掘算法

Alpha++算法与Alpha#算法，除去了不可以挖掘非自由选择结构与不可见任务相结合的情况，在节2.2.1与节2.2.1中还有列举各自的问题。如Alpha++算法不可以挖掘非自有选择结构中含有长度为1的循环的情况，Alpha#算法不可以挖掘不可以挖掘不可见任务在并行中任务的一个分支的情况。在新算法的具体设计中，需要研究算法在挖掘中遇到这些问题的原因，在新算法的设计中弥补这些问题，扩大新算法的挖掘能力。

3.2研究新算法的正确性与挖掘能力

在设计出新算法以后，重要的一个环节就是对算法的正确性的证明与挖掘能力的评估。

关于算法正确性：因为算法是基于推理规则完成的，通过工作流模型的行为语义分析，可以形式化的对于算法的挖掘模型进行证明，即证明在一定的条件下，过程挖掘算法所挖出来的模型一定是正确的。这个证明正是我在本课题中所需要研究的内容。

关于挖掘能力：对于大多数过程挖掘算法，均可以通过一定的人为设计，找出一个合适工作流模型集合，使得过程挖掘算法在这个模型集合中的性能较其他算法优秀。在这种情况下，并不可以说明这个过挖掘算法下整体的表现优秀。因此，在本课题中设计的新算法，我们希望可以验证出它在任何Alpha++与Alpha#表现优秀的模型中均表现优秀，并且在上述两个算法中表现不好的模型集合中也表现优秀，即如图3.2所示。（文氏图中圈的内容代表了挖掘算法所能够正确挖掘出来的模型的集合。）

以上部分是主要的研究内容，在研究方案中做了一些初步的研究和探讨，后面还需要进一步的动手实践才能完成整个论文的研究。

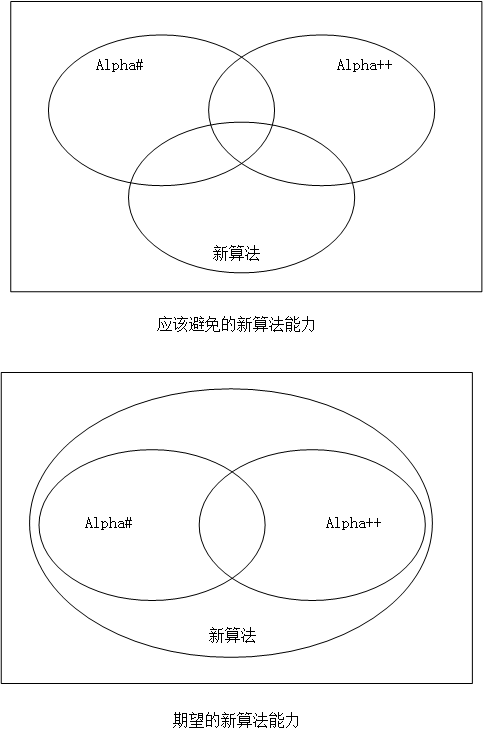


图3.2应该避免的算法的挖掘能力，与期望的算法挖掘能力

文氏图每个算法对应的圈表示了该算法可以正确挖掘出来的模型的集合

4.研究方案

与第3章研究内容相对应，研究方案也需要从新算法的设计和挖掘能力评估两个方面来考虑，具体针对每个部分的研究方案如下：

**4.1新算法具体实现方法的研究方**案

如图4.1所示，新算法的研究主要分为四个部分，分别是不可见任务与非自由选择结构的结合情况、设计新算法（Alpha++算法与Alpha#算法融合）、算法的正确性证明，以及算法的挖掘能力证明。并且在证明阶段，发现算法出现了问题需要对算法的设计进行迭代运算。

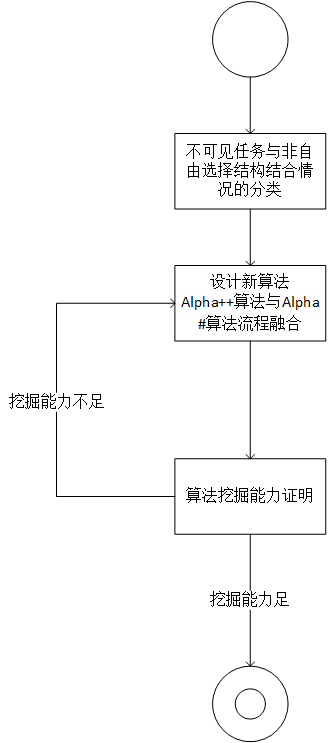


图4.1 新算法具体实现方法中的任务流程

4.2不可见任务与非自由选择结构结合情况的分类的研究方案

Alpha++算法将根据挖掘的情况，将挖掘不可见任务分为了三种情况，见图2.2，图2.3图2.4。Alpha#算法根据不可见任务的类型，将工作流模型不可见任务分为了五种情况，并且将这五种情况统一为一种不可见任务的，见图2.12。

针对非自有选择结构与不可见任务的结合情况，一个可行的方法针对Alpha++算法中三种非自有选择结构中所有的变迁均穷举来尝试变成不可见任务，然后将所有的情况中相同的情况进行分类，进而找出不可见任务与非自由选择结构结合情况的分类情况。

4.3Alpha++算法与Alpha#算法流程融合的研究方案

Alpha++算法与Alpha#算法的流程中均首先对挖掘事件日志中任务间的关系进行分析，利用公式2-1来进行构造找出库所变迁的结合进而构造出模型。

两个算法的不同点在于一个关注任务间的虚假依赖（Alpha#算法，不可见任务），一个关注任务间的隐形依赖（Alpha++算法，非自由选择结构）。

因此在新设计的算法中，尝试在分析事件日志中任务关系时，同时分析虚假依赖和隐形依赖关系，来达到挖掘模型中同时含有不可见任务与非自有选择结构的目的。

4.4新算法正确性与挖掘能力的研究方案

Alpha++算法与Alpha#算法等基于推理规则的过程挖掘算法均使用工作流模型的行为语义来证明算法在一定的工作流模型集合上的正确性。

本课题设计的新的过程挖掘算法为同样基于推理规则来完成，加之新算法的步骤与Alpha++算法和Alpha#算法相类似，因此在算法正确性方面，仿照上述两个算法同样使用行为语义来证明。

在新算法的挖掘能力的评估方面，通过集合论的相关证明手段，需要证明在Alpha++算法与Alpha#算法可以表现优秀的模型在新的挖掘算法中表现优秀。

5.预期成果及可能的创新点

**预期科研成果**

1.设计并实现一个基于推理规则面向非自有选择结构和不可见任务的过程挖掘算法，算法的实现集成在BeehiveZ或ProM中。

2发表科技论文1-2篇。

**可能的创新点**

1.提出一种基于推理规则的过程挖掘算法，能够挖掘同时含有非自有选择结构和不可见任务的工作流模型。

2 验证提出的过程挖掘算法的正确性，并且基于推理规则对其挖掘能力进行分析。

6.研究计划

**1.文献调研阶段（2013年7月-2013年9月）**

调研了现阶段过程挖掘的国内外现状，了解几类经典过程挖掘算法机理，并重点研究基于推理规则的过程挖掘算法。

**2.算法的设计与实现（2013年9月 - 2013年12月）**

根据前面阅读的大量文献和学习的相关技术，提出并实现一个新的基于推理规则的过程挖掘算法。

**3.算法挖掘能力理论验证（2014年1月 - 2014年5月）**

针对上一阶段设计的过程挖掘算法，分别对算法挖掘正确性与挖掘能力两个方面进行形式化的验证，并根据验证结果对设计的算法进行调整。

**4.实验的性能评价（2014年6月 - 2014年8月）**

针对设计的过程挖掘算法，设计性能测试方案。根据方案，对新算法和现有经典过程挖掘算法进行大量对比实验，记录结果，并进行综合分析。

**5.总结与科技论文撰写阶段（2014年9月 - 2014年12月）**

总结过程挖掘算法在设计、理论验证和实际新能分析中，分析工作中的创新点与贡献点，完成一到两篇科技论文。

**6.撰写毕业论文（2015年1月 - 2015年3月）**

总结研究过程出现的问题和技术难点，撰写毕业论文。

下面是具体的工作计划表：

|  |  |
| --- | --- |
| **工作内容** | **时间段** |
| 文献调研阶段 | 2013.7 - 2011.9 |
| 算法的设计与实现 | 2013.10- 2013.12 |
| 算法挖掘能力理论验证 | 2014.1- 2014.5 |
| 实验性能评价 | 2014.6- 2014.8 |
| 总结与科技论文撰写阶段 | 2014.9 - 2014.12 |
| 毕业论文撰写 | 2015.1 - 2015.3 |

致谢

感谢王建民老师与闻立杰老师在我的开题报告与科研路上的孜孜教导。

参考文献

[1] Dumas M, Van der Aalst W M, Ter Hofstede A H. Process-aware information systems: bridging people and software through process technology[M]. Wiley-Interscience, 2005.

[2] W. M. P. van der Aalst, K.M. van Hec. Workftow Managcment: Models, Methods and Systems. MIT press, Cambridge, MA, 2002

[3] W. M. P. van der Aalst, & K.M. van Hee著.王建民 闻立杰 译.工作流管理—模型、方法和系统.清华大学出版社,北京,2004

[4]Workflow Management Coalition. [EB/OL][2013-9-24] <http://www.wfmc.org>

[5]TIBCO,Inc.Staffware Process Suite. [EB/OL][2013-9-23] http://www.staffware.com

[6] IBM, Inc. IBM MQSeries [EB/OL][2013-9-23]. [http://www.mqseries.net/](http://www.mqseries.net/index.html)

[7]COSA, Inc. COSA Business Process Management. [EB/OL][2013-9-25]

<http://www.cosa-bpm.com>

[8]SAP.SAP. [EB/OL][2013-9-23] <http://www.sap.com>

[9] Oracle Business Process Suite. [EB/OL][2013-9-25]

<http://www.oracle.com/us/technologies/bpm/suite/overview/index.html>

[10] 武年华,金涛,查海平等.BeehiveZ:一个开放的业务过程模型管理框架[J].计算机研究与发展,2010,47(z1):450-454.

[11] Jin T, Wang J, Wen L. Efficiently Querying Business Process Models with BeehiveZ[C]//BPM (Demos). 2011.

[12] van Dongen B F, de Medeiros A K A, Verbeek H M W, et al. The ProM framework: A new era in process mining tool support[M]//Applications and Theory of Petri Nets 2005. Springer Berlin Heidelberg, 2005: 444-454.

[13]J.E. Cook， A.L. Wolf. Automating Process Discovery Through Event-Data Analysis. Proceedings of the 17th international conference on Software engineering. New York， NY， USA，1995. 73–82.

[14] R. Agrawal， D. Gunopulos， F. Leymann. Mining Process Models from Workflow Logs. In: I. Ramos， G. Alonso， H.J. Schek， editors， Proceedings of Sixth International Conference on Extending Database Technology， 1998. 469–483.

[15] R Agrawal，D Gunopulos， F Leymann. Mining process models from workflow logs[C] Proc.ofthe 6th Inter.Conf.on Extending Database Technology (EDBT)， Valencia， Spain， Expanded version available as IBM Research Report，R J 10100，1998

[16] van der Aalst W, Adriansyah A, de Medeiros A K A, et al. Process mining manifesto[C]//Business process management workshops. Springer Berlin Heidelberg, 2012: 169-194.

[17] Van der Aalst W M P, van der Aalst W. Process mining: discovery, conformance and enhancement of business processes[M]. Springer, 2011.

[18] J.E. Cook, Z. Du， C. Liu， et al. Discovering models of behavior for concurrent work?ows.Computers in Industry， 2004， 53(3):297–319.

[19] J.E. Cook. Process Discovery and Validation Through Event-Data Analysis: [PhD Thesis].USA: University of Colorado， Sep， 1996

[20] J.E. Cook, A.L. Wolf. Discovering Models of Software Processes from Event-Based Data.ACM Transactions on Software Engineering and Methodology， 1998， 7(3):215–249

[21] J.E. Cook, A.L. Wolf. Event-Based Detection of Concurrency. Proceedings of the Sixth International Symposium on the Foundations of Software engineering. New York， NY， USA，1998. 35–45.

[22] J.E. Cook， Z. Du. Discovering thread interactions in a concurrent system. Journal of Systems and Software， 2005， 77(3):285–297.

[23] Wen L, van der Aalst W M P, Wang J, et al. Mining process models with non-free-choice constructs[J]. Data Mining and Knowledge Discovery, 2007, 15(2): 145-180.

[24] Wen L, Wang J, van der Aalst W M P, et al. A novel approach for process mining based on event types[J]. Journal of Intelligent Information Systems, 2009, 32(2): 163-190.

[25]W.M.P. van der Aalst， A.J.M.M.Weijters， and L.Maruster. Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering， 16(9):1128–1142， 2004.

[26] Medeiros A. Genetic Process Mining. PhD thesis [D], Eindhoven University of Technology, Eindhoven, 2006.

[27] Rozinat A, Aalst W. Conformance checking of processes based on monitoring real behavior. [J] Information Systems, 2008.

[28] A.K.A. de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. Genetic process mining: an experimental evaluation. [J] Data & Knowledge Engineering, 14:245{304, 2007.

[29] W.M.P. van der Aalst, V. Rubin, B.F. van Dongen, E. Kindler, and C.W. G¨unther. Process Mining: A Two-Step Approach using Transition Systems and Regions.[R] BPM Center Report BPM-06-30, BPMcenter.org, 2006

[30] Rozinat A, Aalst W. Conformance checking of processes based on monitoring real behavior. [J] Information Systems, 2008.