一种基于逻辑 Petri 网的过程挖掘方法

杜玉越,朱鸿儒,王 路,刘 伟
(山东科技大学山东省智慧矿山信息技术重点省级实验室,山东青岛 266590)

摘 要: 逻辑 Petri 网是抑制弧 Petri 网和高级 Petri 网的抽象和扩展,可在过程挖掘中简洁准确的表示活动之间复杂的业务逻辑关系. 本文在传统 Petri 网挖掘方法的基础上,为了进一步提高复杂系统挖掘模型的简洁度和拟合度,尤其是对并行活动间存在复杂与或关系的系统,提出了一种基于逻辑 Petri 网的过程挖掘方法,并给出了逻辑 Petri 网中逻辑变迁的挖掘算法. 它可以充分挖掘活动之间的业务逻辑,并且业务逻辑可用逻辑表达式表示. 通过与相应 Petri 网模型的实例比较分析,例证了本文方法的正确性和有效性,且逻辑 Petri 网模型更加适合日志行为.

关键词: 过程挖掘; Petri 网; 逻辑 Petri 网; 逻辑变迁; 挖掘算法
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2016)11-2742-10
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.11.025

A Method of Process Mining Based on Logic Petri Nets

DU Yu-yue, ZHU Hong-ru, WANG Lu, LIU Wei
(*Key Laboratory for Wisdom Mine Information Technology of Shandong Province, Shandong University of Science and Technology, Qingdao, Shandong* 266590, *China*)

Abstract: Logic Petri nets are the abstraction and extension of the Petri nets with inhibitor arcs and high level Petri nets, and can describe the Business Logic among activities concisely and accurately in process mining. To further improve the simplicity and fitness of the mining models of complex systems, especially the systems with the complex AND-OR relation in parallel activities, a method of process mining is proposed based on Logic Petri nets to improve Petri net models in this paper. An algorithm of mining logic transitions is presented to transform the business logic among activities in event logs into logic expressions adequately. Experiment results illustrate that the mining logic Petri nets can represent the event logs more properly and succinctly than the corresponding Petri nets.

Key words: process mining; Petri net; logic Petri net; logic transition; mining algorithm

1 引言

过程挖掘通过挖掘实际的日志信息,发现并改进现实的模型[1],主要包括过程发掘、一致性检测和过程改进. 在过程挖掘研究中,由于 Petri 网[2]描述和分析并发事件的优越性,许多研究都以 Petri 网作为描述过程的工具. α 算法[3]是根据活动的顺序关系进行过程挖掘,当模型中不包括重复活动、不可见变迁以及特定结构式时,能挖掘得到系统的正确模型. 由于 α 算法不能有效挖掘包括重复活动、不可见变迁和特定结构的系统模型,出现了 α 算法的多种扩展. 针对 α 算法不能挖掘不可见变迁问题,文献[4]提出了可以有效挖掘不可见变迁的 alpha#算法. 针对 α 算法无法判定非自由选择结构问题,文献[5]提出了相应的过程挖掘方法. 除 α 算法外,还有基于区域的过程挖掘算法[6,7]. 文献[6]从日志挖掘低等级的变迁系统,通过变迁系统得到模型,从而克服了 α 算法的一些局限性. 为了进行多角度挖掘,出现了基于颜色 Petri 网 CPN(Color Petri Net)的过程挖掘算法[7,8],并通过对 Petri 网进行拓展以考虑时间和成本对工作流决策的影响. 这些方法通常以 Petri 网[9,10]为描述工具,但当并行活动之间的逻辑关系更加复杂时(如存在复杂的与或关系),仅使用 Petri 网不足以对其进行简洁地描述,且得到的模型不能充分反映日志信息. 图 1 和图 2 是由相同旅游预订日志 $L_1 = \{\sigma_1$

收稿日期:2015-06-08;修回日期:2015-12-28;责任编辑:蓝红杰
基金项目:国家自然科学基金(No. 61170078, No. 61472228);山东省泰山学者建设工程专项经费;山东省自然科学基金(No. ZR2014FM009);青岛市科技计划基础研究项目(No. 13-1-4-116-jch);山东省优秀中青年科学家科研奖励基金(No. BS2015DX010)

$=t_1t_2t_5$, $\sigma_2=t_1t_2t_4t_5$, $\sigma_3=t_1t_4t_2t_5$, $\sigma_4=t_1t_4t_2t_3t_5$, $\sigma_5=t_1t_2t_4t_3t_5$, $\sigma_6=t_1t_2t_3t_4t_5$, $\sigma_7=t_1t_3t_5\}$得到的 Petri 网模型,其中图 1 是使用 α 算法挖掘得到的 Petri 网模型,图 2 是由文献[7]给出的先得到 C-net 后转化得到的 Petri 网模型. t_1 代表开始预订,t_2 代表预订机票,t_3 代表预订车票,t_4 代表预订旅馆,t_5 代表完成预定. 黑方框代表不可见变迁,即在日志中没有对应的事件. 预订机票(t_2)和预订宾馆(t_4)之间业务逻辑关系为:(1)预定机票 t_2 可以单独发生;(2)在 t_2 和 t_4 同时发生时它们的关系是并行的. 传统挖掘方法可能认为 t_2 和 t_4 是并行的,但忽视了 t_2 可以单独发生的情况. 此外,由图 2 知,在挖掘得到的 Petri 网模型中,需要添加大量的不可见变迁.

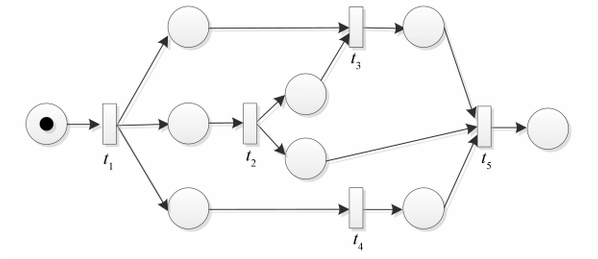

图1　传统α算法挖掘的Petri网模型

针对上述问题,有些学者提出先挖掘得到 C-net 模型,再将其转化成等价的 Petri 网模型,并使得到的模型比原有方法得到的模型更准确. 为了得到更为简洁和更高拟合度的 Petri 网模型,本文采用另一种方法,在 α 算法挖掘结果的基础上,利用任务发生的频次关系得到活动之间的与或关系,通过活动之间的关系推导得到逻辑 Petri 网中的逻辑表达式,从而对挖掘模型进行约束,使得挖掘结果更加简洁、贴合日志,进而得到逻辑 Petri 网模型[11,12,13]. 同时,最后得到的逻辑 Petri 模型,也可以有效解决由 α 算法表示偏好(representational bias)引起的死锁问题.

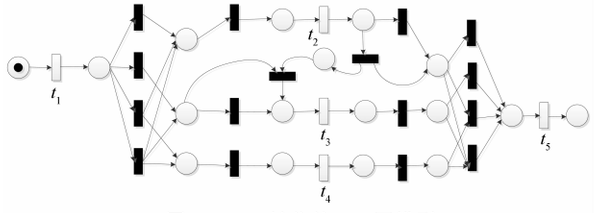

图2　C-net转化的Petri网模型

2　基本概念

本节介绍逻辑真值与逻辑 Petri 网的相关概念. 在逻辑 Petri 网中,通过逻辑表达式约束网中逻辑变迁的引发. 为了讨论逻辑表达式的值,首先需要引入 Petri 网和逻辑真值的概念.

定义 1　**(Petri 网)** 四元组 $PN=(P,T;F,M)$ 称为一个 Petri 网 PN,其中 P 是一个有限库所集,T 是一个有限变迁集,$F\subseteq(P\times T)\cup(T\times P)$ 是一个有限弧集,$M:P\to N$ 称为网 PN 的一个标识.

有关 Petri 网更详细的定义请参见文献[2,9,10].

定义 2　**(逻辑真值)** 设 f 为库所集合 P 的逻辑表达式,M 为 P 的标识向量,对 $\forall p\in P$,记 $p|_M$ 为在 M 下 p 的逻辑真值,记 $f|_M$ 为逻辑表达式 f 在 M 下的逻辑真值,$f(p_1,p_2,\cdots,p_n)|_M=f(p_1|_M,p_2|_M,\cdots,p_n|_M)$ 其中 $p|_M$ 的值定义如下:

$$p|_M=\begin{cases}.T., & \text{若 } M(p)\geqslant 1\\ .F., & \text{若 } M(p)=0\end{cases}$$

在定义 2 中,$.T./.F.$ 为逻辑值为真/假. 下面给出逻辑 Petri 网的概念.

定义 3　**(逻辑 Petri 网)** 一个逻辑 Petri 网定义为一个六元组 $LPN=(P,T;F,I,O,M)$,其中

(1)P 是一个有限库所集.

(2)$T=T_D\cup T_I\cup T_O$ 是一个有限变迁集,$T\cap P=\varnothing$. 若 $t\in T_I\cap T_O$,则 ${}^{\cdot}t\cap t^{\cdot}=\varnothing$.

(a)T_D 表示 Petri 网中的变迁集;

(b)T_I 表示逻辑输入变迁集,对 $\forall t\in T_I$,t 的输入库所 ${}^{\cdot}t$ 受逻辑表达式 $f_I(t)$ 的限制;

(c)T_O 表示逻辑输出变迁集,对 $\forall t\in T_O$,t 的输出库所 $t^{\cdot}$ 受逻辑表达式 $f_O(t)$ 的限制;

(3)$F=(P\times T)\cup(T\times P)$ 是一个有限弧集;

(4)I 是由逻辑输入变迁到逻辑输入函数的映射,对于 $\forall t\in T_I$,$I(t)=f_I(t)$;

(5)O 是由逻辑输出变迁到逻辑输出函数的映射,对于 $\forall t\in T_O$,$O(t)=f_O(t)$;

(6)$M:P\to Z_0$ 是 Petri 网的表示函数,$Z_0=\{0,1,2\cdots\cdots\}$;

(7)变迁引发规则:

(a)对 $\forall t\in T_D$,变迁引发规则与传统 Petri 网一致. 即设 $M[t>M'$,$M'(p)$ 定义如下:

$$M'(p)=\begin{cases}M(p)+1, & \text{若 } p\in t^{\cdot}-{}^{\cdot}t;\\ M(p)-1, & \text{若 } p\in {}^{\cdot}t-t^{\cdot};\\ M(p), & \text{其他.}\end{cases}$$

(b) $\forall t\in T_I$,若 $f_I(t)|_M=.T.$,则逻辑输入变迁 t 可引发,记做 $M[t>M'$,且对 $\forall p\in {}^{\cdot}t$,$M'(p)=0$;对 $\forall p\notin t^{\cdot}\cup {}^{\cdot}t$,$M'(p)=M(p)$;对 $\forall p\in t^{\cdot}$,$M'(p)=1$;

(c)对 $\forall t\in T_O$,若 $\forall p\in {}^{\cdot}t$,$M(p)=1$,则逻辑输出变迁 t 可引发,且对 $\forall p\in {}^{\cdot}t$,$M'(p)=0$;对 $\forall p\in t^{\cdot}$,需满足 $f_O(t)|_{M'}=.T.$;对 $\forall p\notin t^{\cdot}\cup {}^{\cdot}t$,$M'(p)=M(p)$.

图 3 给出了一个逻辑 Petri 网的例子. t_1 为逻辑输入变迁,$I(t_1)=(p_1\otimes p_2)\wedge p_3$ 是 t_1 的逻辑输入函数,$(p_1\otimes p_2)$ 表示 t_1 触发时,p_1 和 p_2 中不能同时存在托肯.

$(p_1 \otimes p_2) \wedge p_3$ 表示 t_1 在如下两种情况下满足触发条件:(1)p_1 和 p_3 中存在托肯;(2)p_2 和 p_3 中存在托肯. t_2 为经典 Petri 网的变迁,触发规则与经典 Petri 相同. t_3 为逻辑输出变迁,其逻辑输出函数 $O(t_3) = p_7 \vee p_8$ 表示 t_3 引发后会有三种情况:(1)p_7 和 p_8 中有托肯;(2)p_7 中有托肯;(3)p_8 中有托肯.

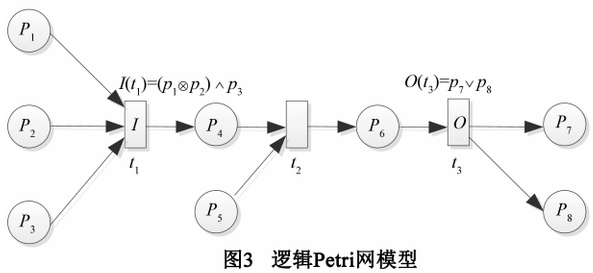

图3 逻辑Petri网模型

3 逻辑变迁的挖掘方法

本节给出逻辑变迁挖掘算法的相关定义以及算法描述. 挖掘算法主要实现单个逻辑变迁的挖掘. 由于逻辑输入变迁和逻辑输出变迁的相关结论可以通过逆网进行转化,因此本文仅给出挖掘逻辑输入变迁的挖掘方法. 不同挖掘方法得到的 Petri 网模型可能不同,故将其转化为逻辑 Petri 网的方法也有所区别. 因此,本文研究在 α 算法得到的 Petri 网模型基础上,通过进一步日志挖掘得到逻辑 Petri 网的方法. 首先引入日志和迹的概念.

定义 4 (迹,日志) $PN = (P, T; F, M)$ 为一个 Petri 网,日志 $L \subseteq T*$,迹 $\sigma \in L$. 这里对于 $\forall t \subseteq T$, $\exists \sigma \in L$,满足 $t \in \&(\sigma)$,即 L 中记录了网中所有可以被记录的活动. 用 $\&(\sigma)$ 表示由迹 σ 中所有活动构成的集合.

在挖掘到的 Petri 网模型中,若把变迁 t 转换为逻辑输入变迁,需要基于日志信息挖掘 t 与前面相邻变迁之间的逻辑关系. 因此,下面引入前序变迁集的概念.

定义 5 (前序变迁集) 设 $PN = (P, T; F, M)$ 为一个由日志 L 得到的 Petri 网,$t_k \in T$,t_k 的前序变迁集定义为 $T_{bef} | t_k = \{t | p \in {}^{\cdot} t_k \wedge t \in {}^{\cdot} p\}$.

在定义 5 中,$T_{bef} | t_k$ 包括了模型中 t_k 所有的前序变迁. 通过将日志在 $T_{bef} | t_k$ 上进行投影,可得到 $T_{bef} | t_k$ 中变迁与 t_k 的逻辑关系. 例如,在图 1 中,$T_{bef} | t_5 = \{t_2, t_3, t_4\}$,$\sigma_6 = \langle t_1 t_2 t_3 t_4 t_5 \rangle$,则 σ_6 在 $T_{bef} | t_5$ 上的投影为 $\sigma_6' = \langle t_2 t_3 t_4 \rangle$. 由于挖掘模型中可能存在循环结构,在对原日志 L 的迹 σ 投影时,应在 t_k 处将 σ 分段. 如 $\sigma = < t_1, t_2 \cdots t_k \cdots, t_n >$ 中只有一个 t_k,应拆成 $\sigma_1 = < t_1, t_2 \cdots t_k >$ 和 $\sigma_2 = < t_k \cdots t_n >$. 在 PN 中,将 t_k 转换为逻辑输入变迁时,对 $\forall t_m \in T_{bef} | t_k$,若存在路径 $\theta = < t_m, \cdots, t_m >$,则需要满足条件 $t_k \in \theta$. 也就是说,对前序变迁集中的变迁,若该变迁在 PN 的一个循环结构中,则这个结构必经过 t_k. 若将经过分段并投影后得到新的日志记为 L',则 $a >_{L'} b$ 表示在 L' 中存在一个迹 $\sigma = < t_1, t_2, t_3, \cdots, t_n >$, $i \in \{1, \cdots, n-1\}$,使 $t_i = a$ 和 $t_{i+1} = b$. 下面给出 $T_{bef} | t_k$ 中变迁逻辑关系的定义.

定义 6 (变迁逻辑关系) 设 $PN = (P, T; F, M)$ 为一个由日志 L 得到的 Petri 网,$t_k \in T$,t_k 的前序变迁集为 $T_{bef} | t_k$. L' 为 L 在 $T_{bef} | t_k$ 投影得到的日志,σ 为日志的一条迹,对 $T_1, T_2 \subseteq T_{bef} | t_k$, $(a >_{L'} b) | \sigma$ 表示在 σ 上活动 a 和 b 的关系为 $a >_{L'} b$. 令 $A = \{\sigma | \sigma \in L' \wedge (a >_{L'} b) | \sigma, a \in T_1, b \in T_2\}$, $B = \{\sigma | \sigma \in L' \wedge (b >_{L'} a) | \sigma, a \in T_1, b \in T_2\}$, $C = \{\sigma | \sigma \in L' \wedge (a \in \&(\sigma))\}$, $D = \{\sigma | \sigma \in L' \wedge (b \in \&(\sigma))\}$. 令 $k_1 = |A| / \min\{|C|, |D|\}$, $k_2 = |B| / \min\{|C|, |D|\}$. 给定常数 m 和 n 用于度量变迁的并发程度,且 $0 < m < n < 1$. 基于日志的活动逻辑关系 $lr(a, b)$ 定义为:

$$lr(a,b) = \begin{cases} a \otimes b, \text{若 } k_1 + k_2 < m; \\ a \rightarrow b (b \leftarrow a), \text{若 } k_1 > n \text{ 且 } k_2 < m; \\ a \wedge b, \text{若 } k_1 + k_2 > n \text{ 且 } \min\{k_1, k_2\} > m; \\ a \vee b, \text{其他}. \end{cases}$$

在定义 6 中,$a \rightarrow b$ 或 $b \leftarrow a$ 表示 b 依赖于 a, $\vee$ 表示或关系, $\wedge$ 表示与关系. k_1 和 k_2 是介于 0 到 1 之间的数,用于描述变迁 a 和 b 的依赖关系. 当 k_1 接近于 0 时,表示在 L' 中极少出现 $a >_{L'} b$. 当 k_1 接近于 1 时,表示 L' 中 a 和 b 的关系为 $a \rightarrow b$. m 和 n 是用于度量 a 和 b 的并发程度的数值. 当 $lr(a, b) = a \otimes b$ 时,日志 L' 中极少出现 $a >_{L'} b$ 或 $b >_{L'} a$,即 k_1 和 k_2 都接近 0. 因此,这里记为 $k_1 + k_2 < m$. 当 $lr(a, b) = a \wedge b$ 时,a 和 b 的并发程度较高,且 $k_1 + k_2$ 接近于 1,记为 $k_1 + k_2 > n$. 当 $k_1 + k_2 > n$ 时,还存在 k_1 很大而 k_2 很小的情况,即 $a \rightarrow b$. $a \wedge b$ 与 $a \rightarrow b$ 区别在于 $a \wedge b$ 满足条件 $\min\{k_1, k_2\} > m$. 这里 m 和 n 的值是在分析问题时,由用户自定义的数值.

定义 6 给出了前序变迁集中变迁的四种逻辑关系. 在此基础上,可以进一步将前序变迁集中变迁的逻辑关系反映到其后继库所上,使库所的逻辑关系与变迁的逻辑关系保持一致.

定义 7 (库所逻辑关系) 设 $PN = (P, T; F, M)$ 为一个由日志 L 得到的 Petri 网,$t \in T$,$T_{bef} | t$ 为 t 的前序变迁集. $a, b \in T_{bef} | t$, $P_a = \{p | p \in a^{\cdot} \cap {}^{\cdot} t\}$, $P_b = \{p | p \in b^{\cdot} \cap {}^{\cdot} t\}$, $P_{a-b} = P_a - P_b$, $P_{b-a} = P_b - P_a$. 库所逻辑关系 $pl(lr(a, b))$ 定义为:

(1)若 $lr(a, b) = a \wedge b$,则 $pl(lr(a, b)) = f(P_a) \wedge f(P_b) = .T.$ 即 t 触发时,在 $T_{bef} | t$ 中 a 和 b 都触发;

(2)若 $lr(a, b) = a \otimes b$,则 $pl(lr(a, b)) = (f(P_a) \wedge \neg f(P_{b-a})) \vee (f(P_b) \wedge \neg f(P_{a-b})) = .T.$,即 t 触发

时,在 $T_{bef}|t$ 中 a 和 b 只有一个能触发;

(3)若 $lr(a,b)=a\vee b$,则 $pl(lr(a,b))=f(P_a)\vee f(P_b)=.T.$,即 t 触发时,在 $T_{bef}|t$ 中需要 a 或 b 触发;

(4)若 $lr(a,b)=a\rightarrow b$,$pl(lr(a,b))=(f(P_a)\wedge\neg f(P_b))\vee(f(P_b)\vee\neg f(P_a))=.T.$,即 t 触发时,只有 P_a 或 P_b 中存在托肯.

在定义 7 中,当 $a\otimes b$ 时,$P_a\cap P_b\neq\varnothing$,此时讨论的集合为 P_{a-b} 和 P_{b-a}. 日志 $L_4=\{abc,ac\}$ 由 α 算法挖掘的 Petri 网模型如图 4 所示. 但是,在该模型中,不能反映日志中的 $a\rightarrow t$. 这个问题可在相应的逻辑 Petri 网中解决,且其优化后的逻辑 Petri 网模型如图 5 所示,其中 $I(t)=(f(P_a)\wedge\neg(f(P_b)\vee((f(P_b)\wedge\neg f(P_a)))$. 下面的定理 1 将证明若 $\forall c\in T_{bef}|t,c\neq a$,则 $lr(c,b)\neq c\rightarrow b$. 即在 $T_{bef}|t$ 中,只有 a 会与 b 存在依赖关系的情况下. P_a 和 P_b 的逻辑关系 $pl(lr(a,b))=(f(P_a)\wedge\neg f(P_b))\vee(f(P_b)\wedge\neg f(P_a))$,即在处理 $a\rightarrow b$ 时,应将其视为 $a\otimes b$.

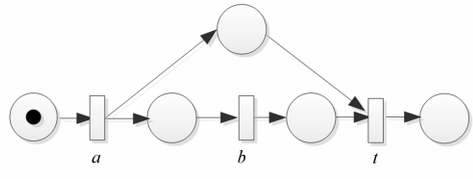

图4　{a→b, b→t, a→t}挖掘得到的Petri网模型

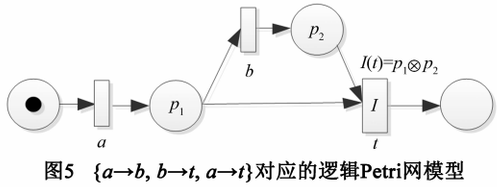

图5　{a→b, b→t, a→t}对应的逻辑Petri网模型

定理 1　设 $LPN=(P,T;F,I,O,M)$ 为一个由日志 L 得到的 Petri 网,$t\in T,T_{bef}|t$ 为 t 的前序变迁集,L' 是 L 在 $T_{bef}|t$ 上的投影,$a,b\in T_{bef}|t$ 且 $a,b\notin T_O$. 若 $a\rightarrow b$,且 $M[t>$,则 P_a 与 P_b 的逻辑关系满足 $(f(P_a)\wedge\neg f(P_b))\vee(f(P_b)\wedge\neg f(P_a))=.T.$.

证明:先考虑不引入不可见变迁的情况,且 $P_a\in{}^{\cdot}b$. 采用反证法,令 $P_a\notin{}^{\cdot}b$,由 $a,b\in T_{bef}|t,a\rightarrow b$,则 $\exists M[a>M'$,$(f(P_a)|_{M'})\wedge\neg f({}^{\cdot}b)|_{M'})=.T.$. 令 $M'[b>M''$,有 $(f(P_a)|_{M''})\wedge\neg f(P_b)|_{M''})=.T.$. 若 t 触发,则表达式应为 $(f(P_a)|_{M''})\wedge\neg f(P_b)|_{M''})$,且原日志中不存在迹 $<a,t>$,这与原有日志矛盾. 因此,$P_a\in{}^{\cdot}b$.

因为 $P_a\in{}^{\cdot}b$,故 $\forall M,M[b>,f(P_b)|_M=.T.,f(P_a)|_M=.F.$,且 $f(P_b)|_M\wedge\neg f(P_a)|_M=.T.$. 由 $a\in T_i|t_1$,故对 $\forall M[a>M',M'[t>$,有 $f(P_b)|_{M'}=.F.,f(P_a)|_{M'}=.T.$,且 $f(P_a)|_{M'}\wedge\neg f(P_b)|_{M'}=.T.$. 因此,逻辑输入变迁 t 的逻辑输入函数应为 $(f(P_a)\wedge\neg f(P_b))\vee(f(P_b)\wedge\neg f(P_a))$. [证毕]

由定理 1,当 $a\rightarrow b$ 时,P_a 和 P_b 的行为与 $a\otimes b$ 相同. 故在下面处理 $a\rightarrow b$ 时,可把它作为 $a\otimes b$ 处理. 从逻辑变迁的逻辑输入函数可知,两者是等价的.

定义 7 描述了逻辑输入变迁的前序库所逻辑与其前序变迁集中变迁逻辑之间的关系. 由定义 7 和定理 1,根据日志可得到逻辑变迁的逻辑输入表达式,再将逻辑关系转化成逻辑变迁的逻辑输入函数,可得到逻辑输入表达式 $I(t)$. 为了得到逻辑变迁的逻辑输入函数,需要引入逻辑前序矩阵的概念.

定义 8　**(逻辑前序矩阵)** $PN=(P,T;F,M)$ 是由日志 L 得到的 Petri 网,$t_1\in T,T_{bef}|t$ 为 t 的逻辑前序变迁集. t 的逻辑前序矩阵定义为 $(\boldsymbol{AI}|t)[n][n](n=|T_{bef}|t|)$. 设 $i,j\in[0,n)$,当 $i\neq j$ 时,$(\boldsymbol{AI}|t_1)[i][j]=lr(i,j)$,反之,$(\boldsymbol{AI}|t_1)[i][j]=\varnothing$.

由定义 8,逻辑前序矩阵可表达前序变迁集中任意两个变迁的逻辑关系. 对于给定的前序变迁集中的任意个变迁,通过逻辑前序矩阵可以得到这些变迁之间的逻辑关系. 下面给出前序逻辑表达式的概念.

定义 9　**(前序变迁表达式)** $PN=(P,T;F,M)$ 是由日志 L 得到的 Petri 网,$t\in T,T_{bef}|t$ 为 t 的逻辑前序变迁集,t 的逻辑前序矩阵为 $(\boldsymbol{AI}|t)$. 对 $\forall T'\subseteq T_{bef}|t,\lambda(T',\boldsymbol{AI}|t_1)$ 称为由 $\boldsymbol{AI}|t$ 得到的关于 T' 的前序变迁表达式.

基于逻辑前序矩阵挖掘前序变迁集中变迁的逻辑关系时,要考虑⊗和∧,以及⊗和∨是否满足分配律. 如 $a,b,c\in T_{bef}|t_1,lr(a,b)=a\wedge b,lr(a,c)=a\wedge c,lr(b,c)=b\otimes c$,若先挖掘得到⊗,可得到表达式 $f_1=a\wedge(b\otimes c)$;若先挖掘得到∧,可得到表达式 $f_2=(a\wedge b)\otimes(a\wedge c)$. 但是,下面将证明这两种顺序是等价的,即∧对⊗满足分配率. 这里证明 $pl_{f1}(lr(a,lr(b,c)))=pl_{f2}(lr(a,c),lr(a,b))$,其中 $pl_{f1}(lr(a,b))$ 表示在变迁表达式 f_1 下得到的库所逻辑表达式.

定理 2　设 $PN=(P,T;F,M)$ 为一个由日志 L 得到的 Petri 网,$t\in T,T_{bef}|t$ 为 t 的前序输入变迁集,$a,b,c\in T_{bef}|t$. 若 $lr(a,b)=a\wedge b,lr(a,c)=a\wedge c$,且 $lr(b,c)=b\otimes c$,则对于逻辑表达式 $f_1=a\wedge(b\otimes c)$ 与 $f_2=(a\wedge b)\otimes(a\wedge c)$,有 $pl_{f1}(lr(a,lr(b,c)))=pl_{f2}(lr(a,c),lr(a,b))$.

证明:由定义 6,左边 $=pl_{f1}(lr(a,lr(b,c)))=f(p_a)\wedge f(p_{b,c})=f(p_a)\wedge((f(P_c)\wedge\neg f(P_{b-c}))\vee(f(P_b)\wedge\neg f(P_{c-b})))=((f(p_a)\wedge f(P_c)\wedge\neg f(P_{b-c}))\vee(f(p_a)\wedge f(P_b)\wedge\neg f(P_{c-b}))$. 右边 $=pl_{f2}(lr(a,c),lr(a,b))=(f(p_{a,c})\wedge\neg f(p_{(a,b)-(a,c)})\vee(f(p_{a,b})\wedge\neg f(p_{(a,c)-(a,b)}))=((f(p_a)\wedge f(P_c)\wedge\neg f(P_{b-c}))\vee(f(p_a)\wedge f(P_b)\wedge\neg f(P_{c-b}))=pl_{f1}(lr(a,lr(b,c)))$.

因此,左边 = 右边,即原等式成立. [证毕]

类似的还可以得到⊗对∧也满足分配率,即⊗和

∧都满足分配率. 同样可以推出⊗和∨也满足分配率. 在 Petri 网中给出一个变迁 t 的逻辑前序矩阵 $\boldsymbol{AI}|t$,根据定理2,下面将给出前序变迁集中变迁逻辑关系的挖掘算法.

定义 10 (逻辑输入函数)设 $LPN=(P,T;F,I,O,M)$是由日志 L 得到的逻辑 Petri 网,$t\in T$,$T_{bef}|t$ 为 t 的逻辑前序变迁集,λ 为由矩阵 $\boldsymbol{AI}|t$ 得到的逻辑表达式. t 的逻辑输入函数定义为 $f_I(t)=pl(\lambda)$.

定义 10 给出了逻辑表达式与逻辑输入函数的关系,下面的算法 1 将实现由前序变迁矩阵挖掘出前序变迁集中变迁的逻辑关系.

算法 1 前序变迁集逻辑关系挖掘算法

设 $PN=(P,T;F,M)$为一个由日志 L 得到的 Petri 网,$t\in T$,$T_{bef}|t$ 为 t 的输入变迁集. $\boldsymbol{AI}|t$ 是 t 的逻辑前序矩阵. 变迁集合 $T'\subseteq T_{bef}$,A 为由 $\boldsymbol{AI}|t$ 得到的只包含 $T'_{bef}|t$ 中变迁的矩阵,并假设每列对应变迁依次为 $t_1,t_2\cdots..t_{|A|}$.

输入:$A,k=0,T'\subseteq T_{bef}|t$

输出:$\lambda(T',\boldsymbol{AI}|t)$

(1)遍历 $\boldsymbol{A}[i][j](0\leqslant i<j<n)$,若 $\boldsymbol{AI}[i][j]=$'←',则调换 t_i,与 t_j 顺序重构 $\boldsymbol{A}$;若 $\boldsymbol{AI}[i][j]=$'→',则根据定义 10,使模型中 $a^{\cdot}\cap{}^{\cdot}b=p_a$ 并修改 PN,且令 $\boldsymbol{A}[i][j]=\otimes$. 根据 $\boldsymbol{A}$ 中关系,更新 $\boldsymbol{AI}$;

(2)逐列遍历 $\boldsymbol{A}[i][j]$,$j=1$ 时,$\lambda=lr(t_1,t_2)$;

(3)$j++$,若 $j>|t|$,转(4);若 $\boldsymbol{A}[i][j]==$'⊗',$\lambda_k=lr(t_i,t_j)$,$T'=T'-\{t_i,t_j\}$,$k=k+1$,转(3);

(4)令 $m\in\{1,|t|\}$. 若 $m>|t|$,转(5). 若 $t_m\in T'$,令 $\lambda_k=t_m$,$T'=T'-t_m$,$m=m+1$,转(4);

(5)对于 $p,q\in(1,k)$,遍历 $\boldsymbol{A}[i][j](j>i)$,若 $\boldsymbol{AI}[i][j]=$'∧',且满足 $\lambda_p=\lambda_p\wedge\lambda_q,\lambda_q=\varnothing$;

(6)$\lambda=\lambda_1\vee\lambda_2\vee\cdots\vee\lambda_k$,算法结束.

在算法 1 中,为了处理'→'与'∧',下面分别给出相应的转换规则.

规则 1 ($a\rightarrow b$ 的转换规则)$LPN=(P,T;F,I,O,M)$为一个由日志 L 得到的逻辑 Petri 网,$t\in T$,$T_{bef}|t_1$ 为 t 的前序输入变迁集. $a,b\in T_{bef}|t$,若 $a\rightarrow b$ 且 $a\notin T_O$,根据定理 1,使 $a^{\cdot}\cap{}^{\cdot}b=p_a$,即合并 $a^{\cdot}\cap{}^{\cdot}b$ 与 p_a. 在 $\boldsymbol{AI}|t$ 中把 $lr(a,b)$ 置为⊗. 若 $a\in T_O$,则只在 $\boldsymbol{AI}|t$ 中将 $lr(a,b)$ 置为⊗.

对于规则 1,虽然在逻辑输出变迁中也可以得到类似规则,但变迁 a 和 b 在相邻的逻辑输入变迁和逻辑输出变迁之间时,可能会导致同样的结构被处理两次. 为了避免这个问题,规定同一个 $a\rightarrow b$ 结构只能被处理一次,且在其他变迁中记 $a\rightarrow b$ 为 $a\otimes b$.

规则 2 ($a\otimes b$ 的转换规则)$LPN=(P,T;F,I,O,M)$为一个由日志 L 得到的逻辑 Petri 网,$t\in T$,$T_{bef}|t$ 为 t 的输入变迁集. $a,b\in T_{bef}|t$ 且 $a,b\notin T_O$. $pl(a\otimes b)=(f(p_a)\wedge\neg f(p_{b-a}))\vee(f(p_b)\wedge\neg f(p_{a-b}))$,其中

(1)当 $p_a\cap p_b=\varnothing$时,$pl(a\otimes b)=f(p_a)\otimes f(p_b)$;

(2)当 $p_b=p_a$ 时,$pl(a\otimes b)=f(p_a)\vee f(p_b)=f(p_a)$;

(3)当 $p_b\neq p_a$ 和 $p_b-p_a=\varnothing$,即 $pl(a\otimes b)=f(p_a)\vee(f(p_b)\wedge f(p_{a-b}))$时,$F=F-a\times(p_a\cap p_b)$,$p_a=p_a-p_b$,且 $pl(a\otimes b)=(f(p_a)\wedge\neg f(p_b))\vee(f(p_b)\wedge\neg f(p_a))$;

(4)当 $p_b-p_a\neq\varnothing$和 $p_a-p_b\neq\varnothing$时,令 $p_{ab}=p_a\cap p_b$,$p'_a=p_a-p_{ab}$,$p'_b=p_b-p_{ab}$,有 $pl(a\otimes b)=(f(p_a)\wedge\neg f(p_{b-a}))\vee(f(p_b)\wedge\neg f(p_{a-b}))=(f(p'_a)\wedge f(p_{ab})\wedge\neg f(p'_b))\vee(f(p'_b)\wedge f(p_{ab})\wedge\neg(f(p'_a))=f(p_{ab})\wedge(f(p'_a)\otimes f(p'_b))$. 此时,$F=F-a\times p_{ab}-b\times p_{ab}$. $pl(a\otimes b)=f(p'_a)\otimes f(p'_b)$. 若 $\forall p\in p_{ab}$,且 $|{}^{\cdot}p|=0$,则 $F=F-p\times t$,$P=P-\{p\}$.

规则 2 给出了化简逻辑 Petri 网模型和逻辑表达式的方法. 如在图 1 中,$pl(t_1,t_4)=(f(p_{t1})\wedge\neg f(p_{t4-t1}))\wedge(f(p_{t4})\wedge\neg f(p_{t1-t4}))$. 因为 $p_{t1-t4}=\varnothing$,故 $F=F-a\times(p_{t1}\cap p_{t4})$,$p_{t4}=p_{t4}-p_{t1}=p_2$,且 $pl(t_1,t_4)=p_1\otimes p_2$. 这样就能利用引入的逻辑表达式简化逻辑 Petri 网模型. 将挖掘得到的模型转化成它的逆网,可以得到逻辑输出变迁的类似定义.

在执行算法 1 时,利用规则 1 和规则 2 可以改善原有挖掘结果. 在处理 $t_1\otimes t_2$ 中第三种情况时,将出现库所的消减. 算法 1 没有考虑(的优化顺序,这会影响最终得到的逻辑 Petri 网的挖掘效果. 图 6 是由 $L_4=\{\langle t_1,t_3,t_5,t_7\rangle^2,\langle t_1,t_5,t_3,t_7\rangle^3,\langle t_2,t_4,t_6,t_7\rangle^2,\langle t_2,t_6,t_4,t_7\rangle^4\}$挖掘得到的 Petri 网. 若将 t_7 视为逻辑输入变迁,图 7 给出了其逻辑前序矩阵. 使用算法 1,首先寻找 $\boldsymbol{AI}|t_7$ 中为⊗的元素,有 $\lambda_1=t_3\otimes t_4$,$\lambda_2=t_3\otimes t_6$,$\lambda_3=t_4\otimes t_5$,且 $\lambda_4=t_5\otimes t_6$. 根据 $t_3\wedge t_5$,得 $\lambda_1=\lambda_1\wedge\lambda_3=t_4\otimes(t_3\wedge t_5)$,$\lambda_2=\lambda_2\wedge\lambda_4=t_6\otimes(t_3\wedge t_5)$. 由 $t_4\wedge t_6$,得 $\lambda_1=\lambda_1\wedge\lambda_2=(t_3\wedge t_5)\otimes(t_4\wedge t_6)$. 此时,得到如图 8 所示的逻辑 Petri 网模型,且根据 $t_3\wedge t_5$ 得 $\lambda_1=\lambda_1\wedge\lambda_4=(t_3\otimes t_4)\wedge(t_5\otimes t_6)$. $\lambda_2=\lambda_2\wedge\lambda_3=(t_3\otimes t_6)\wedge(t_4\otimes t_5)$. $\lambda_1=\lambda_1\wedge\lambda_2=(t_3\otimes t_4)\wedge(t_5\otimes t_6)\wedge(t_3\otimes t_6)\wedge(t_4\otimes t_5)=(t_3\wedge t_5)\otimes(t_4\wedge t_6)$. 因此,得到如图 9 所示的逻辑 Petri 网模型.

图 8 和图 9 的前序变迁表达式相同,但最后的逻辑输入表达式以及逻辑 Petri 网模型不同. 图 8 中出现了一次规则 2 中的第(4)种情况,而图 9 中出现了两次这种情况. 由于第(4)种情况涉及到库所的消减,因此,图 9 的库所和弧比图 8 的都更少. 为了使得到的逻辑 *Petri* 网模型尽量简洁,在处理⊗时,采用贪心算法. 优先处理出度较多的变迁,且在处理时优处理规则 2 中的第(4)种情况. 因此,可以得到改进后的算法 2.

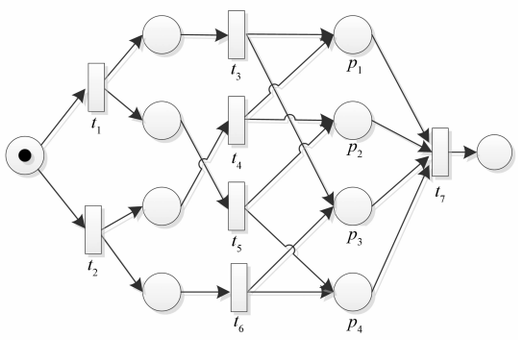

图6　由L_4得到的Petri网

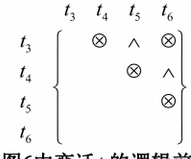

图7　图6中变迁t_7的逻辑前序矩阵

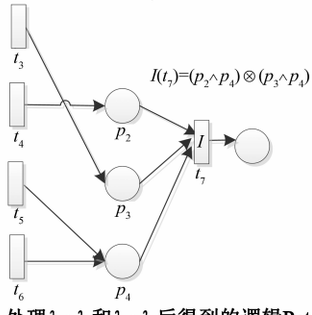

图8　处理$\lambda_1\wedge\lambda_3$和$\lambda_2\wedge\lambda_4$后得到的逻辑Petri网

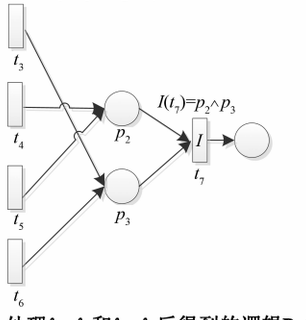

图9　处理$\lambda_1\wedge\lambda_4$和$\lambda_2\wedge\lambda_3$后得到的逻辑Petri网

算法 2　改进的前序变迁集逻辑关系挖掘算法

$PN=(P,T;F,M)$为一个由日志 L 得到的 Petri 网,$t\in T$,$T_{bef}|t$ 为 t 的输入变迁集. $\boldsymbol{AI}|t$ 是 t 的逻辑前序矩阵. 变迁集合 $T'_{bef}\subseteq T_{bef}$,$\boldsymbol{A}$ 为由 $\boldsymbol{AI}|t$ 得到的只包含 T'_{bef}中变迁的矩阵,$\lambda=f(\boldsymbol{A})$,并设每列对应变迁依次为 $t_1,t_2\cdots\cdots t_{|A|}$.

输入:$\boldsymbol{A}$,$k=0$,$T'=T_{bef}|t$

输出:λ

(1)遍历 $\boldsymbol{A}[i][j](0\leqslant i<j<n)$,若 $\boldsymbol{AI}[i][j]=$'←',则调换 t_i,与 t_j 顺序,重构 $\boldsymbol{A}$;若 $\boldsymbol{AI}[i][j]=$'→',则根据定义 10 使模型中 $a\cdot\cap\cdot b=p_a$,并修改 PN,令 $\boldsymbol{A}[i][j]=\otimes$. 根据 $\boldsymbol{A}$ 中关系更新 $\boldsymbol{AI}$;

(2)逐列遍历 $\boldsymbol{A}[i][j]$,当 $j=1$ 时,$\lambda=lr(t_1,t_2)$;

(3)$j++$,若 $j>|t|$,转 4);若 $\boldsymbol{A}[i][j]==\otimes$,则 $\lambda_k=lr(t_i,t_j)$,$T'=T'-\{t_i,t_j\}$,$k=k+1$,转 3);

(4)令 $m\in\{1,|t|\}$. 若 $m>|t|$,转 5);若 $t_m\in T'$,则令 $\lambda_k=t_m$,$T'=T'-t_m$,$m=m+1$,转 4);

(5)利用贪心算法处理 $\lambda_1,\lambda_2\cdots\lambda_k$. 记 out-degree($\lambda$)为逻辑表达式 λ 中变迁出度的和,优先处理 out-degree(λ)最高的逻辑表达式;

(6)对于 $p,q\in(1,k)$,遍历 $A[i][j](j>i)$,若 $\boldsymbol{AI}[i][j]=$'∧',且 $\lambda_p=\lambda_p\wedge\lambda_q$,则 $\lambda_q=\varnothing$;

(7)$\lambda=\lambda_1\vee\lambda_2\vee\cdots\vee\lambda_k$,算法结束.

由算法 2,可以确定前序变迁集中变迁之间的逻辑关系. 在图 1 中,$\{t_2,t_3,t_4\}$通过算法 2 可以得到$(t_2\wedge t_4)\otimes(t_3\wedge t_4)$,代表迹 $\sigma\in\{t_1t_2t_3t_4t_5,\ t_1t_4t_2t_3t_5,\ t_1t_2t_4t_3t_5\}$. 但此时无法表示 t_2 与 t_3 单独发生的情况. 为了解决这个问题,算法 3 将日志在前序变迁集上进行投影. 将投影后长度相同的迹分为一组,并将每组中的变迁用算法 2 得到其逻辑关系. 这样就可以得到能反映日志的前序变迁表达式,进而计算相应的逻辑输入函数.

算法 3　逻辑输入表达式挖掘算法

设 $PN=(P,T;F,M)$为一个由日志 L 得到的 Petri 网,$t\in T$,$T_{bef}|t$ 为 t 的前序输入变迁集. $\boldsymbol{AI}|t$ 是 t 的逻辑前序矩阵. $f_l(t)$为 t 的逻辑输入函数,LPN 为转化后的逻辑 Petri 网.

输入:$\boldsymbol{AI}|t$,$\lambda=\varnothing$,PN

输出:$f_l(t)$,LPN

(1)将 L 在 $T_i|t$ 上进行投影,得到 L'. 假设 L'中的迹最短长度为 m,最大长度为 n;

(2)设 $k=1$,且集合 $\boldsymbol{A}$ 为长为 m 的迹中出现变迁的集合,根据算法 1,由 $\boldsymbol{A}$ 得到 λ_1,并使 $\lambda=\lambda_1$;

(3)$k++$,若 $m+k-1>n$,转(4);设 $\boldsymbol{A}$ 为长为 $m+k-1$ 的迹中出现变迁的集合. 根据算法 1,由 $\boldsymbol{A}$ 得到 $\lambda=\lambda(\lambda_k$,转(3);

(4)计算 $f_l(t)=pl(\lambda)$,算法结束.

本节给出了一种单个逻辑输入变迁的挖掘方法. 在此基础上,按优化所有路由选择变迁得到的逻辑 Petri 网模型,比原有 Petri 网模型能更充分反映日志中的信息. 而优化的顺序不同会引起最终结果的不同,这会在下面的例子中给与展示.

4　实例分析

本节给出第 3 节挖掘方法的实例分析,从而展示以逻辑 Petri 网挖掘模型比 Petri 网挖掘模型更符合日志行为. 以某三甲医院实际就诊日志为例,给出其逻辑 Petri 网挖掘模型与相应 Petri 网挖掘模型的比较分析,其中 Petri 网模型是由启发式(Heuristic)算法得到的模型转化而来的.

例 1　本例中的日志和图 1 中的日志轨迹相同,$L_1=\{\sigma_1=t_1t_2t_5,\sigma_2=t_1t_2t_4t_5,\sigma_3=t_1t_4t_2t_5,\sigma_4=t_1t_4t_2t_3t_5,\sigma_5$

$=t_1t_2t_4t_3t_5,\sigma_6=t_1t_2t_3t_4t_5,\sigma_7=t_1t_3t_5\}$. 首先将 t_5 转换为逻辑输入变迁:由模型可知 $T_i|t_5=\{t_2,t_3,t_4\}$,由定义6,可得到日志并发度 $k(t_2,t_3)=0.6,k(t_3,t_2)=0,k(t_2,t_4)=0.25,k(t_4,t_2)=0.25,k(t_3,t_4)=0.25,k(t_4,t_3)=0.25$. 取 $m=0.3,n=0.5$,则 $lr(t_2,t_3)=t_2\rightarrow t_3,lr(t_2,t_4)=t_2\wedge t_4,lr(t_3,t_4)=t_3\wedge t_4$.

构建逻辑前序矩阵 $\boldsymbol{AI}|t_5$ 如下:

$$\begin{array}{c} \\ t_2 \\ t_3 \\ t_4 \end{array}\begin{array}{c} \begin{array}{ccc} t_2 & t_3 & t_4 \end{array} \\ \left\{\begin{array}{ccc} & \rightarrow & \wedge \\ \leftarrow & & \wedge \\ \wedge & \wedge & \end{array}\right\} \end{array}$$

将 L_1 在 $T_i|t_5$ 上的投影,记为 L_1',则 $L_1'=\{\sigma_1'=t_2,\sigma_2'=t_2t_4,\sigma_3'=t_4t_2,\sigma_4'=t_4t_2t_3,\sigma_5'=t_2t_4t_3,\sigma_6'=t_2t_3t_4,\sigma_7'=t_3\}$. 根据算法3,将 L_1' 分组:$\{\sigma_1',\sigma_7'\},\{\sigma_2',\sigma_3'\},\{\sigma_4',\sigma_5',\sigma_6'\}$. 其中,$\{\sigma_1',\sigma_7'\}$ 所构成的逻辑前序矩阵为

$$\begin{array}{c} \\ t_2 \\ t_3 \end{array}\begin{array}{c} \begin{array}{cc} t_2 & t_3 \end{array} \\ \left\{\begin{array}{cc} & \otimes \\ \otimes & \end{array}\right\} \end{array}$$

$\{\sigma_2',\sigma_3'\}$ 构成的逻辑前序矩阵为:

$$\begin{array}{c} \\ t_2 \\ t_4 \end{array}\begin{array}{c} \begin{array}{cc} t_2 & t_4 \end{array} \\ \left\{\begin{array}{cc} & \wedge \\ \wedge & \end{array}\right\} \end{array}$$

$\{\sigma_4',\sigma_5',\sigma_6'\}$ 构成的逻辑前序矩阵为:

$$\begin{array}{c} \\ t_2 \\ t_3 \\ t_4 \end{array}\begin{array}{c} \begin{array}{ccc} t_2 & t_3 & t_4 \end{array} \\ \left\{\begin{array}{ccc} & \otimes & \wedge \\ \otimes & & \wedge \\ \wedge & \wedge & \end{array}\right\} \end{array}$$

根据上述三个逻辑前序矩阵,依次得到 $\boldsymbol{\lambda}_1=t_1\otimes t_2,\boldsymbol{\lambda}_2=t_2\wedge t_4,\boldsymbol{\lambda}_3=t_2\wedge t_4,\boldsymbol{\lambda}_4=(t_2\otimes t_3)\wedge t_4$,且 $\boldsymbol{\lambda}=(t_2\otimes t_3)\vee(t_2\wedge t_4)\vee((t_2\otimes t_3)\wedge t_4)$. 根据相应转换规则,合并 p_{t2} 与 $^{\cdot}t_3$ 得到图10中的逻辑 Petri 网. 根据定义7,$I(t_5)=pl(\boldsymbol{\lambda})=(p_3\otimes p_4)\vee(p_3\wedge p_5)\vee(p_3\otimes p_4)\wedge p_5)=(p_3\otimes p_4)\vee((p_3\otimes p_4)\wedge p_5)$. 将 t_3 转化为逻辑输入变迁时,由于 t_3 的前序变迁集为 $\{t_1,t_2\}$,且 $lr(t_1,t_2)=t_1\rightarrow t_2$,则合并 p_{t1} 与 $^{\cdot}t_2$ 后,有 $I(t_3)=p_1\otimes p_3$. 处理 t_1 时,由于 $t_2\rightarrow t_3$ 在处理 t_5 时已被处理,记 $lr(t_2,t_3)=t_2\otimes t_3$,得 $O(t_1)=p_1\vee(p_1\wedge p_2)$.

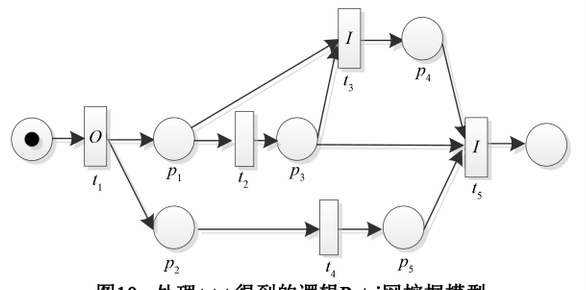

图10 处理$t_5t_3t_1$得到的逻辑Petri网挖掘模型

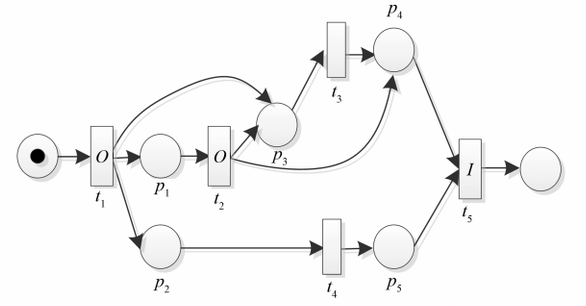

图11 处理$t_1t_2t_5$得到的逻辑Petri网挖掘模型

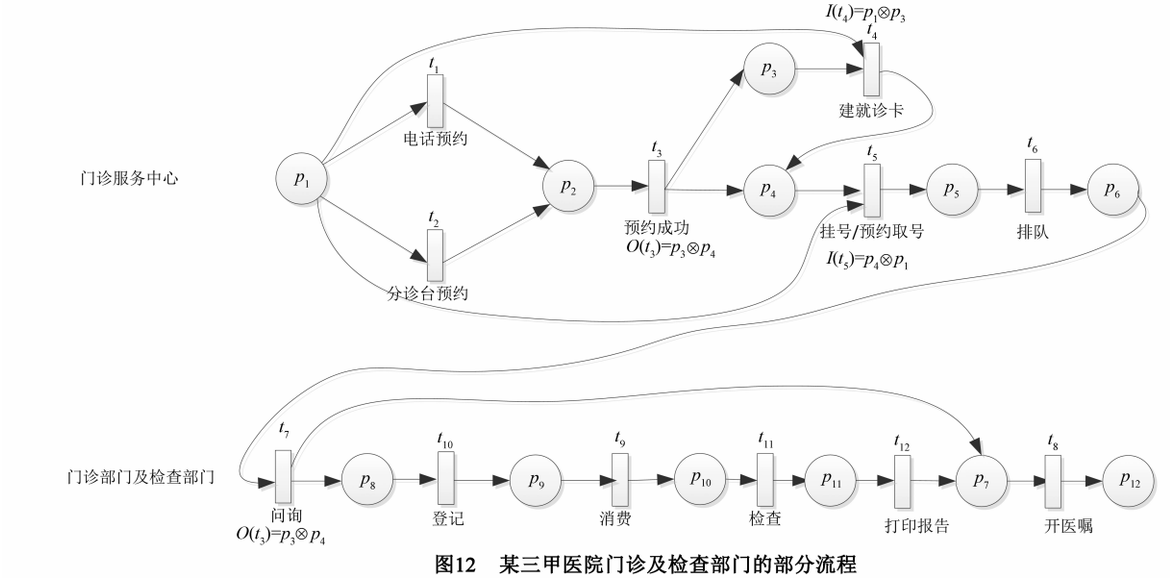

图12 某三甲医院门诊及检查部门的部分流程

若处理变迁 $t_1t_2t_5$,可以得到图11 中的逻辑 Petri 网模型. 在图 11 中,$O(t_1)=(p_1\otimes p_3)\vee((p_1\otimes p_3)\wedge p_2)$,$O(t_2)=p_3\otimes p_4$,$I(t_5)=p_4\vee(p_4\wedge p_5)$. 图 11 中的模型可正确反映原日志中的模型. 依次处理变迁 $t_3t_1t_5$ 和 $t_3t_5t_1$ 可以得到和图 10 相同的逻辑 Petri 网,依次处理 $t_2t_1t_5$ 和 $t_2t_5t_1$ 可以得到和图 11 相同的逻辑 Petri 网模型. 对比图 1 和图 10,日志 L_1 在图 1 中只有 $\sigma_4,\sigma_5,\sigma_6$ 重演,而日志 L_1 在图 11 中的迹均可以被重演. 因此,挖掘得到的逻辑 Petri 网模型相比原 Petri 网模型,能更加充分的反映日志中的信息. 对比图 2 和图 10,两者均可以重演日志 L_1 中的迹,但通过比较比较库所和变迁的数目可以发现图 10 中的模型更为简洁.

例 2:图 12 给定某三甲医院门诊服务和检查的部分流程. 图 13 为图 12 中 t_{11}代表的子流程,包括 6520 条迹,以及九个事件:挂号、PET-CT、普通 CT、胸部增强 CT、生化全套、血沉、血气分析、血常规和诊断,在将其进行格式转化成为 XES 文件后,基于该日志,在 ProM6. 5 下用 ProM 中启发式算法挖掘得到的 Petri 网模型,如图 13 所示,其中黑方框表示不可见变迁. 可见变迁中左边第一列变迁名为“挂号”,第二列从上到下变迁名分别为“PET-CT”、“胸部增强 CT”和“普通 CT”,第三列从上到下变迁名分别为“生化全套”、“血沉”、“血气分析”和“血常规”,第四列变迁名为“诊断”. 在 ProM 上使用 α 算法挖掘得到的 Petri 网,如图 14 所示. 在图 14 的基础上,利用本文方法进一步挖掘日志信息,得到的逻辑 Petri 网如图 15 所示,其中“挂号”为逻辑输出变迁,且其逻辑输出表达式为 $O_{\text{register}}=p_1\otimes p_3$. “诊断”为逻辑输入变迁,且其逻辑输入表达式为 $I_{\text{diagnose}}=p_3\otimes p_4$($p_6$. 表 1 对由 Heuristic-net 得到的 Petri 网、α 算法挖掘得到的 Petri 网、C-net 网及相应的逻辑 Petri 网的性能指标,进行了比较分析. 由表 1 可知,图 13 和图 15 的拟合度(Fitness)比图 14 中的模型要好,图 14 和图 15 比图 13 的模型规模更小,且主要体现在库所的规模以及不可见变迁的数量上. 从泛化度(Generalization)以及精确度(precision)[14]上比较,图 15 的模型比图 13 的好.

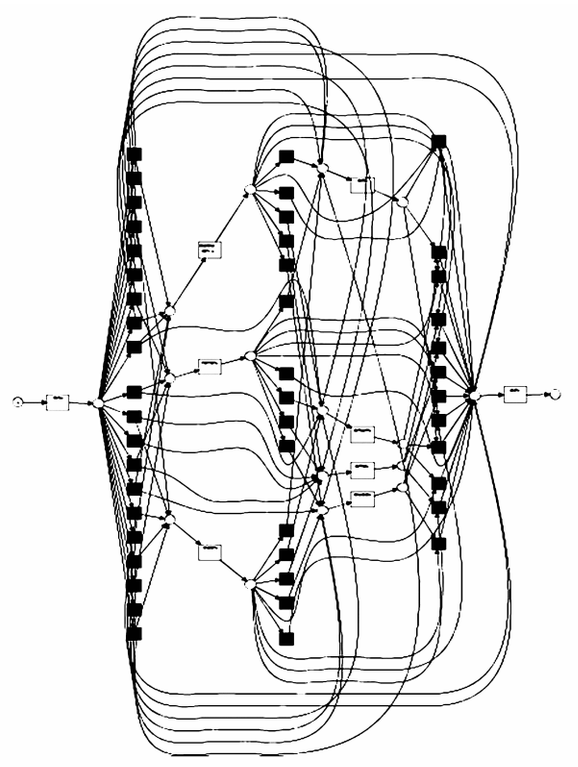

图13　利用Heuristic算法得到的Heuristic-net

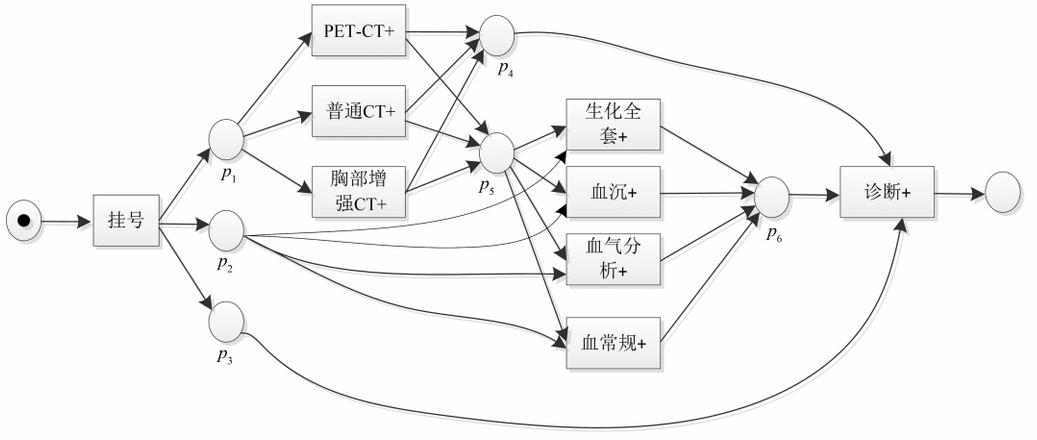

图 14　利用ProM中α算法得到的Petri网

在时间消耗上,α 算法和启发式算法挖掘 Heuristic 网挖掘效率最快,这是因为它们在大规模日志上没有重复的操作,如表 1 所示. 本文方法需要在 α 算法的基础上,还需通过日志得到定义 6 中相应的频次关系. 因

此,本文方法在时间上的花费会比前两种算法要高. 图16 为通过启发式算法挖掘得到 C-net 后,在 ProM 下得到的 BPMN 模型(可转化为与图 15 等价的逻辑 Petri 网). 挖掘 C-net 需要多次遍历日志[1],如对于图 1 中的变迁 t_1,本文方法需要确定$\{t_2,t_3\}$,$\{t_3,t_4\}$和$\{t_2,t_4\}$三组变迁之前的频次关系,而挖掘它的 C-net 需确定所有可能的绑定关系,即$\{t_1,t_2\}$,$\{t_1,t_3\}$,$\{t_1,t_4\}$,$\{t_1,\{t_2,t_3\}\}$,$\{t_1,\{t_3,t_4\}\}$),$\{t_1,\{t_2,t_3\}\}$,$\{t_1,\{t_2,t_4\}\}$,$\{t_1,\{t_2,t_3,t_4\}\}$. 由表 1 知,C-net 的平均时间比本文逻辑 Petri 网花费时间更多. C-net 中没有库所和变迁的概念.

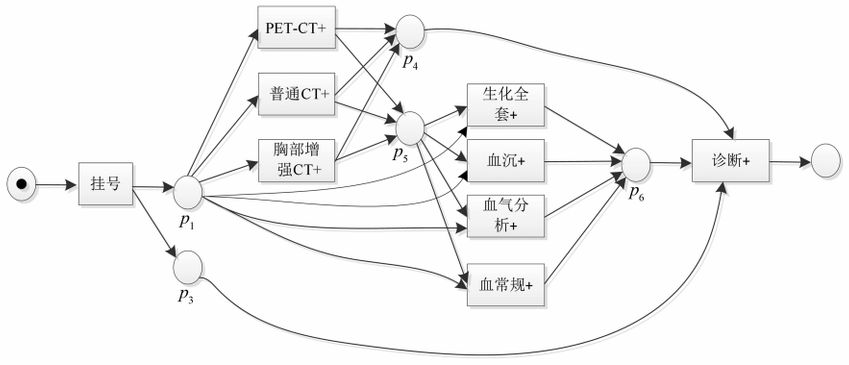

图15 改进后的逻辑Petri网

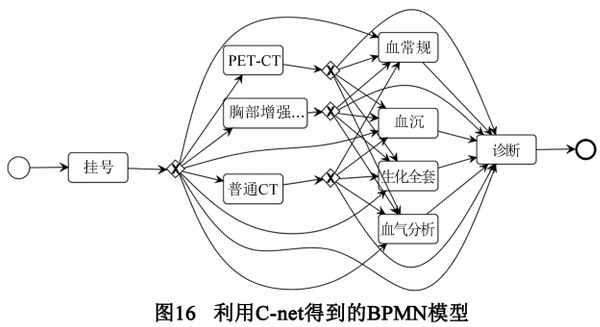

图16 利用C-net得到的BPMN模型

表 1 例 2 中四种模型的比较

	库所	变迁	拟合度	泛化度	精确度	平均时间消耗(s)
Heuristic-net	18	54	0. 9635	0. 8021	0. 8511	0. 6795
Petri 网	8	9	0. 7879	0. 8350	0. 8125	0. 5683
C-net			0. 9768	0. 9066	0. 9332	7. 8626
逻辑 Petri 网	7	9	0. 9998	0. 9378	0. 9363	3. 5039

5 总结

本文针对复杂系统的过程挖掘问题,特别是对于系统中在并行活动之间存在复杂与或关系的过程挖掘,将逻辑 Petri 网应用到过程发掘中,在原有 α 算法的基础上,考虑事件的频次,提出了一种逻辑变迁的挖掘方法. 挖掘得到的逻辑 Petri 网模型能够正确描述活动之间复杂的业务逻辑关系. 本文方法可以作为 α 及其拓展方法的补充,即在原有方法需要更加深入的挖掘日志信息时,可将逻辑 Petri 作为工具进行更深入的挖掘. 因此,本文方法的效率会低于一般的 α 算法或扩展的 α 算法,比如 alpha#算法[4]. 算法效率的提高及批量日志的自动挖掘与比较,将作为下一步的研究内容.

参考文献

[1] Van der Aalst W. Process Mining: Discovery, Conformance and Enhancement of Business Processes[M]. Berlin Heidelberg: Springer, 2011. 1 – 30.

[2] 杜玉越,薛洁,李彦成. 基于服务簇的服务组合替换与分析[J]. 电子学报,2014,42(11):2231 – 2238.
Du Yuyue, Xue Jie, Li Yancheng. Substitution and analysis of service composition based on service clusters[J]. Acta Electronica Sinica, 2014, 42(11): 2231 – 2238. (in Chinese)

[3] Van der Aalst W, et al. Workflow mining: discovering process models from event logs[J]. Knowledge and Data Engineering, IEEE Transactions on, 2004, 16(9): 1128 – 1142.

[4] Wen L, Wang J, van der Aalst W M P, et al. Mining process models with prime invisible tasks[J]. Data & Knowledge Engineering, 2010, 69(10): 999 – 1021.

[5] 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.

[6] Van der Aalst W, et al. Process mining: a two-step approach to balance between underfitting and overfitting[J]. Software & Systems Modeling, 2010, 9(1): 87 – 111.

[7] Busi N, Pinna G, van der Aalst W. An Iterative Algorithm for Applying the Theory of Regions in Process Mining

[M]. Beta, Research School for Operations Management and Logistics, 2007. 16 – 38.

[8] Rozinat A, Mans R S, Song M, et al. Discovering colored Petri nets from event logs [J]. International Journal on Software Tools for Technology Transfer, 2008, 10(1): 57 – 74.

[9] Du Y Y, Jiang C J. On the design and temporal Petri net verification of grid commerce architecture [J]. Chinese Journal of Electronics, 2008, 17(2): 247 – 251.

[10] 梅晓勇,李师贤,等. 一种支持组合事务的执行语义分析方法[J]. 电子学报,2012,40(7):1386 – 1396.
Mei Xiaoyong, Li Shixian, et al. An execution semantic analysis method for composition transaction[J]. Acta Electronica Sinica, 2012, 40(7): 1386 – 1396. (in Chinese)

[11] Du Y Y, Qi L, Zhou M C. Analysis and application of logical Petri nets to e-Commerce systems [J]. Systems, Man, and Cybernetics: Systems, IEEE Transactions on, 2014, 44(4): 468 – 481.

[12] Du Y Y, Jiang C J, Zhou M C. A Petri Net-based model for verification of obligations and accountability in cooperative systems[J]. IEEE Transactions on Systems, Man, and Cybernetics--Part A: Systems and Humans, 2009, 39(2): 299 – 308.

[13] Du Y Y, Ning Y H, Qi L. Reachability analysis of logic Petri nets using incidence matrix[J]. Enterprise Information Systems, 2014, 8(6): 630 – 647.

[14] Van der Aalst W, Adriansyah A, van Dongen B. Replaying history on process models for conformance checking and performance analysis[J]. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2012, 2(2): 182 – 192.

作者简介

杜玉越　男,1960 年生于山东聊城. 博士,教授,博士生导师. 研究方向为过程挖掘、工作流技术、Petri 网应用等.
E-mail: yydu001@163.com

朱鸿儒　男,1991 年生于山东淄博. 硕士研究生. 研究方向为过程挖掘、工作流、Petri 网.
Email: zhr20122013@163.com