软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.006162]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于完全有限前缀展开的行为等价过程树生成算法*

朱 锐[1,2,3,4], 黄 月[1,4], 金 芝[2,3], 李 彤[4,5*], 汤雅惠[1,4]

[1](云南大学 软件学院,云南 昆明 650091)

[2](北京大学 信息科学技术学院,北京 北京 100871)

[3](高可信软件技术教育部重点实验室(北京大学),北京 100871)

[4](云南省软件工程重点实验室,云南 昆明 650091)

[5](云南农业大学 大数据学院,云南 昆明 650201)

通讯作者: 李彤, E-mail: tli@ynu.edu.cn

摘 要: 过程树能够兼具过程模型的行为和结构,对简化模型结构的复杂度方面具有重要意义.现有过程树转化仅能将基于块结构的简单过程模型转化为过程树,但是无法将具有复杂结构的过程模型转化过程树.为此,提出了一种基于完全有限前缀展开的行为等价过程树生成算法,用于将与过程树行为等价的过程模型转化为行为等价过程树.该方法首先利用完全有限前缀展开技术分析过程模型,抽取模型的活动关系;其次通过分析活动关系,进而对模型进行重构.最终通过活动关系判断和模型重构的不断迭代操作,构建行为等价过程树.在实验部分通过在测试模型上的实验,验证了该算法在行为等价过程树生成方面的正确性和可行性.

关键词: 过程模型;复杂结构;Petri 网;过程树;完全有限前缀展开

中图法分类号: TP311

中文引用格式: 朱锐,黄月,金芝,李彤,汤雅惠.基于完全有限前缀展开的行为等价过程树生成算法.软件学报. http://www.jos.org.cn/1000-9825/6162.htm

英文引用格式: Zhu R, Huang Y, Jin Z, Li T, Tang YH. Generating algorithm for the behavior equivalent process tree based on complete finite prefix. Ruan Jian Xue Bao/Journal of Software, 2016 (in Chinese). http://www.jos.org.cn/1000-9825/6162.htm

Generating algorithm for the behavior equivalent process tree based on complete finite prefix

ZHU Rui[1,2,3,4], HUANG Yue[1,4], JIN Zhi[2,3], LI Tong[4,5*], TANG Ya-Hui[1,4]

[1](School of Software, Yunnan University, Kunming 650091, China)

[2](School of Electronics Engineering and Computer Science, Peking University, Peking 100871, China)

[3](Key Laboratory of High Confidence Software Technologies, Ministry of Education (Peking University), Beijing 100871, China)

[4](Key Laboratory in Software Engineering of Yunnan Province, Kunming 650091, China)

[5](School of Big Data, Yunnan Agricultural University, Kunming 650201, China)

* 基金项目: 国家自然科学基金(62002310, 61662085, 61662065); 云南省重大科技专项(202002AD080002); 云南省自然科学基金基础研究面上资助项目(2019FB135);云南大学数据驱动的软件工程省科技创新团队资助项目(2017HC012);云南大学“东陆中青年骨干教师”培养计划资助项目(C176220200)

Foundation item: National Natural Science Foundation of China (62002310, 61662085, 61662065), Major Project of Science and Technology of Yunnan Province (202002AD080002), Yunnan Provincial Natural Science Foundation of China(2019FB135), Yunnan University Data-Driven Software Engineering Provincial Science and Technology Innovation Team Foundation of China(2017HC012), and the Yunnan University “Dong Lu Young-backbone Teacher” Training Program of China(C176220200).

收稿时间: 2020-04-22; 修改时间: 2020-07-11, 2020-08-24; 采用时间: 2020-09-12; jos 在线出版时间:2020-12-02

Abstract: The process tree has both the behavior and structure of process model, and it is significant on simplifying the complexity of the process model. Existing methods can only transform the block structured process model into process tree. However, it is difficult to transform process model with complex structure into process tree. To solve this problem, a novel generating algorithm for the behavior equivalent process tree based on complete finite prefix was proposed. The algorithm is used to transform the process tree in behavior equivalent's process model into behavior equivalent process tree. This algorithm analyses the process model based on an incomplete prefix unfolding technique and extracts the relationships between process model activities. After analyzing the activity relation, the algorithm reconstruct the process model. The behavior equivalent process tree is constructed through activity relation judgment and the iterative operation of model reconstruction. The validity and feasibility of the proposed algorithm in the generation of behavioral equivalent process tree are verified by experiments on the test model.

Key words: process model; complex structures; Petri net; process tree; complete finite prefix

伴随着“工业 4.0”和“中国制造 2025”而来的产品设计、生产装备、产品研发、感知设备等多个生产环节都与过程控制技术紧密相关.过程驱动的基本思想在当前工业 4.0 的生产开发中的作用日益凸显,然而在实践中业务过程模型却呈现海量特征、模型节点过多、行为难以分析等问题.例如,目前 SAP 参考模型大概有 600 多个;中国北车集团过程模型库中已累积约有 20 多万个;澳洲 Suncorp 银行约有 6000 多个过程模型[1].荷兰一家信息通信技术企业的实施软件变更管理系统的日志包含案例数多达 4 万个,其中事件数达到 37 万[2].面对如此海量的过程模型和活动数,模型往往会呈现出一种“意大利面式”的复杂结构[3].针对此类复杂的过程模型,如何有效的提高复杂模型的可理解性是亟待解决的问题.

目前,降低模型复杂性常用的方法有两种:模型频繁行为模式发现[4-6]和精简模型结构[7-9].前者强调从模型执行后的日志中去挖掘模型中的关键路径,去除噪声和非频繁日志所带来的低频次行为依赖,从而减少模型规模.这种方法能有效保留模型中发生次数较多的行为,但也可能去除掉非频繁的关键行为,理想状态下能够有较好的结果.后者精简模型结构主要强调从细粒度的过程模型中去发现高层次的粗粒度的模型结构,常用到的技术包括模型优化[7,10]、模型合并[9]以及事件聚类[11]等.综合上述文献可知,无论是频繁行为模式发现还是对模型结构进行精简,在一定程度上都会对模型的行为造成损失.这使得一些对模型质量要求较高的领域,无法容忍模型行为损失的关键过程(比如软件开发版本的重要分支过程),上述两种降低模型复杂性的方法并不适用.

当前过程模型最常用的形式化表达工具是 Petri 网[12,13],虽然其建模优势在于能够描述真并发、具有图形化描述、具有严格的执行语义等[14],但却时常会由于冗余结构增加了模型的复杂性.为此,本文将聚焦在基于 Petri 网建模的过程模型生成行为等价过程树的表达方法,其中行为等价即 π 演算中模型间轨迹等价的思想,具体定义可参见文献[15].通过结构相对清晰的过程树来表达过程模型,同时尽可能保存模型的行为,以此来降低“意大利面式”复杂结构所引起的模型不易理解问题.

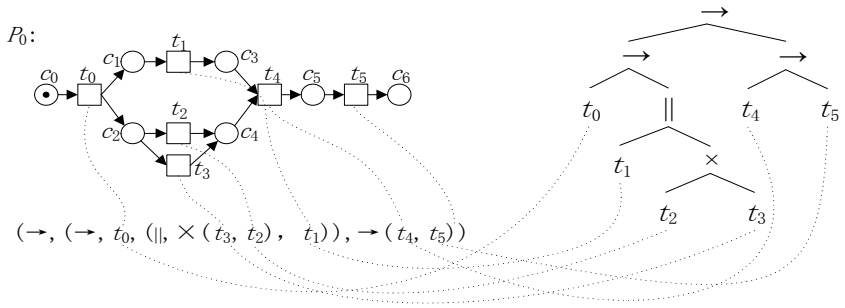


Fig.1 Petri-net onto process tree

图1 Petri 网到过程树的映射

为了在保留行为特征的同时,尽可能降低模型结构所带来的复杂性,我们引入过程树[16]来表示过程模型,这种过程树不仅可以避免这些不合理的构造,还可以兼具过程模型的行为,简化模型的复杂度,有利于直观地对模型进行分析.过程树是一种包含了过程模型信息的树形结构图,过程树不仅有对应的健全模型表示,也支持基本

的控制流模式[16].在过程树中的叶子节点代表活动,分支节点代表活动关系,过程树中任意两个活动(叶子节点)的关系即它们最近的公共父亲节点.例如图 1 中的模型 P_0 里的活动 t_1 和 t_2 之间是并发关系,因此将其转化为过程树后,过程树中的叶子节点 t_1 和 t_2 最近的公共父亲节点是并发.将基于 Petri 网描述的过程模型转化为行为等价过程树的意义在于:

(1) 过程树作为一种树形结构的表示,利用它来描述一个过程模型,结构清晰简单,对模型结构精简工作具有重要帮助.

(2) 过程树可以使用结构化的形式语言直接表达,它在过程模型的检索与存储的领域具有重要意义.当前过程模型通常是以图的形式进行存储,不利于直接对模型进行数据库的管理操作,通过结构化的过程树可以快速的对模型进行查找.

(3) 过程树不仅可以描述过程模型的重要行为特征,更能进一步描述模型的整体行为信息.这有利于理解模型的整体行为结构.

本文提出了一种基于完全有限前缀展开的行为等价过程树生成算法.该算法从过程模型活动的发生权的角度,讨论了如何将用 Petri 网描述的模型转化为行为等价过程树,并分析了不能转化的情况.本文第 1 节重点对行为等价过程树生成问题进行阐述.第 2 节对本文涉及到的基本概念进行表述.第 3 节详细论述了基于完全有限前缀展开的等价过程树生成算法及思路.第 4 节对本文所提算法进行相关实验.第 5 节将算法与其他相关工作进行对比分析.第 6 节对本文的主要工作进行总结并展望未来工作.

1 问题描述

尽管将 Petri 网表示的过程模型转化为过程树模型可简化模型的复杂性,有利于直观地对模型进行分析.但并不是所有的 Petri 网表示的过程模型都可以转化为过程树.当前,对于过程树转化的研究是针对基于块结构的过程模型[16-18](Block-Structured Process Model, BSPM,具体见定义 4),例如图 2 中的模型 P0 就是一个 BSPM 模型.现有的转化算法[19]通过识别活动间的基本块结构,能够将一个 BSPM 转化为一棵过程树,文献[18]提出可以通过补充基本块的方式增加过程树的表达能力,但是仍然只能将 BSPM 转化相应的过程树.例如图 2 中所示过程模型 P_1,相较于图 1 中的过程模型 P_0,P_1 是一个的非 BSPM,在结构上多了库所 c_7 和 c_8,无法通过上述方法产生行为等价的过程树.但通过识别 P_1 的活动间基本关系(发生权关系),能发现 P_1 可以用与 P_0 同的过程树进行表达:$(\rightarrow,(\rightarrow,t_0,(\|,\times(t_3,t_2),t_1)),\rightarrow(t_4,t_5)$.

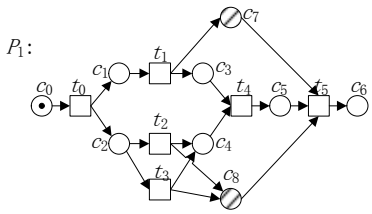


Fig.2 Non-BBSM onto process tree
图2 非 BSPM 转化为过程树

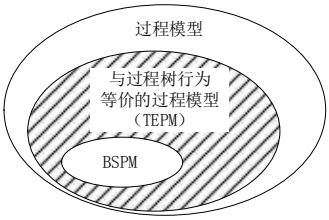


Fig.3 Classification of process models
图3 过程模型分类

根据过程模型是否能够等价转化为行为等价过程树,将过程模型进行了分类(如图 3 所示),其中与过程树的行为等价的过程模型(Process Tree in Behavior Equivalent's Process Model, TEPM,具体见定义 7)是可以转化为行为等价过程树的模型.针对如何将 TEPM 转化为行为等价过程树的问题,本文研究内容及贡献如下:

(1) 提出过程模型活动关系的抽取方法,该方法不仅可以支持基于基本块的过程模型 BSPM 中简单的活动关系抽取,还可以发现现有方法无法解决的具有复杂结构的过程模型活动关系抽取问题(例如图 2 中 t_2 和 t_3 之间的关系).

(2) 提出了模型重构方法,该方法首先定义了活动关系间是否存在且只存在一种基本关系的冲突判断条件,该条件可以用于判断过程模型是否为 TEPM.接着,定义了活动关系的优先级,该优先级能够解决

由于活动重构所导致的其他活动关系无法判断问题.最后,建立了过程重构算法,该算法在不影响过程模型其他结构的基础上,能够将多个活动重构为一个新活动.

(3) 提出了基于完全有限前缀展开的行为等价过程树生成算法,该算法通过活动关系判断和模型重构的不断循环操作,最终构建出行为等价过程树,能够有效的将具有复杂结构的 TEPM(图 3 中阴影部分)转化为过程树.

2 预备知识

为了更好的理解本文提出的方法,下面将给出与本文研究方法相关的一些预备知识.

2.1 过程模型和过程树

定义1(过程模型):一个四元组 $p=(C, A; F, M_0)$作为一个 Petri 网,其中 C 是条件集,$\forall c\in C$ 称作一个条件;A 是活动集,$\forall a\in A$ 称作一个活动,a 的发生称为 a 被执行或者 a 点火;F 是 Petri 网上的流关系,$F\subseteq C\times A\cup A\times C$;$M_0$(${M_0}\subseteq C$)为 p 的初始情态,一个 $d\in M_0$ 被称为托肯.

本文使用 Petri 网系统(EN 系统)来表示过程模型,Petri 网的其他相关概念,请参阅文献[20-22].

定义2(过程树):对于一个过程模型 $p=(C, A; F, M_0)$,A 为其活动集,则活动 $a\in A\cup\{\tau\}$为一个过程树;设 $M_1,M_2,\ldots M_n(n>0)$为 n 个过程树,过程树间关系符号记为◎,则◎(M_1,M_2)为过程树.

本文中使用顺序(→),选择(×),并发(||),迭代(∝)共四种关系符号,使用符号◎表示其中的一种.关于过程树的更多相关概念请参见文献[23,24].

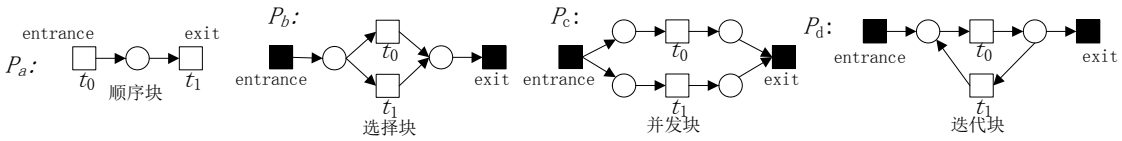


Fig.4 Four basic Block-Structured

图4 四种基本块结构

定义3(基本块):顺序块、选择块、并发块、迭代块被称为基本块,具体如图 4 所示,其中黑色变迁表示内部活动,它不会产生其他任何外部行为.一个五元组 $s=(C, A; F, a_e, a_x)$作为一个基本块,其中 C 是条件集,$\forall c\in C$ 称作一个条件;A 是活动集,$\forall a\in A$ 称作一个活动,a 的发生称为 a 被执行或者 a 点火;F 代表流关系,$F\subseteq C\times A\cup A\times C$;$a_e, a_x\in A$ 为 p 的入口活动和出口活动.

定义4(基于块结构的过程模型, Block-Structured Process Model, BSPM):BSPM 根据严格的活动结构来定义基于块结构,设过程片段 $s=(C, A; F, a_e, a_x)$是一个由基本块构成的过程片段(具体请参阅文献[16-18]),则基于块结构的过程模型 p 的获得方式为:

(1) 构造条件集 C',定义条件 c_e 和 c_x,使 $C'= C\cup\{c_e, c_x\}$.

(2) 构造流关系集 F',定义条件 c_e 和 c_x,使 $F'= F\cup\{(c_e, a_e), (a_x, c_x)\}$.

(3) 为条件 c_e 添加托肯,并设当前情态为初始情态 M_0.

(4) $p=(C', A; F', M_0)$.

定义5(复杂结构):设 s 为一个过程片段,无法使用基本块和有限次地活动重构(具体请参阅文献[16-18])操作得到 s,则将该过程片段 s 称为复杂结构.

定义6(过程模型的行为空间):设 $p=(C, A; F, M_0)$是一个过程模型,设 $\sigma\subseteq A^*$为过程模型 p 上的一个变迁序列,则 p 上所有可能发生的变迁序列的集合称为 p 行为空间,记为$\mathcal{P}(p)$,则 $\sigma\in\mathcal{P}(p)$。

定义7(与过程树行为等价的过程模型, Process Tree in Behavior Equivalent's Process Model, TEPM):一个四元组 $p=(C, A; F, M_0)$被称作一个 TEPM,当且仅当存在一个过程树模型 t,使得 p 与 t 之间满足:

(1) 不存在 σ 使得 $\sigma\in\mathcal{P}(p)\wedge\sigma\notin\mathcal{P}(t)$;

（2）不存在 σ 使得 $\sigma\in\mathcal{P}(t)\wedge\sigma\notin\mathcal{P}(p)$.

其中，σ 表示过程模型 p 或者过程树 t 的一个行为。p 和 t 行为等价从本质上而言就是过程模型 p 中所有出现的行为一定会在过程树 t 中出现，反之亦然。TEPM 包含所有的 BSPM 模型和一部分具有复杂结构的模型,这两部分模型都有对应的行为等价过程树,换而言之,有对应行为等价过程树的过程模型就是 TEPM.

2.2 完全前缀展开

对静态过程模型的分析,传统的方法主要使用可达树或者可达图的方法,这种方法面临的主要问题是“状态空间爆炸”.为避免该问题,1996 年 Esparza 提出了完全前缀展开技术,该技术通过过程模型的完全前缀展开,可以将过程模型展开为一个包含截断事件的分支进程 $\Pi=(o,h)$,其中 $o=(S,T;F')$是一个出现网,h 是一个映射函数,例如针对一个过程模型 $p=(C, A; F, M_0)$利用分支进程的可能扩展集,不断的扩展其对应的分支进程,直到扩展到截断条件时停止,最后得到一个包含截断事件的分支进程.

下面对完全前缀展开技术中一些基本概念进行说明,更多相关介绍详见文献[25].

定义8（出现网）：一个三元组 $o=(S,T;F')$作为一个出现网,其中：

（1）$(S,T;F')$是一个无环的网,$S\cup T\neq\Phi$;S 是一个库所（即条件）的有限集,$\forall s\in S$ 称作一个库所;T 是一个变迁（即事件）的有限集,$\forall t\in T$ 称作一个变迁;F 是出现网的边集,且 $F\subseteq S\times T\cup T\times S$;

（2）$\forall s\in S$,满足$|\cdot s|\leq 1$,即任意库所的输入都小于等于 1;

（3）$\forall x\in S\cup T$,满足 $\neg x\#x$,即任意元素都不能自冲突;

（4）$\forall x\in S\cup T$,满足$\{\forall y\in S\cup T|y<x\}$是有限的.

在出现网中对于$\forall x,y\in S\cup T$,$x\ \#\ y$ 表示 x 与 y 之间存在冲突关系,即在出现网中存在某公共库所 c,从 c 到 x 的路径与从 c 到 y 的路径不相交.$x<y$ 表示 x 与 y 之间存在因果关系,即在出现网中存在一条从 x 到 y 的路径.x co y 表示 x 与 y 之间存在 co 关系,即 x,y 满足 $\neg x<y\wedge\neg y<x\wedge\neg x\#y$.

定义9（分支进程）：对于一个过程模型 $p=(C, A; F, M_0)$,其分支进程是一个二元组 $\Pi=(o,h)$,其中 $o=(S,T;F')$是一个出现网,h 是一个从$(S,T;F')$到$(C,A;F)$的映射,其中：

（1）$h(S)\subseteq C$,$h(T)\subseteq A$,$h(F')\subseteq F$;

（2）$Min(o)$与 M_0 是双射关系,$Min(o)$表示出现网 o 中根据适当偏序关系得到的最小元素集合;

（3）$\forall t_1,t_2\in T$,且 t_1,t_2 满足 $\bullet t_1=\bullet t_2\wedge h(t_1)=h(t_2)$,则 $t_1=t_2$.

定义10　（分支进程的可能扩展集）：二元组 $\Pi=(o,h)$ 是过程模型 $p=(C, A; F, M_0)$的一个分支进程,其中 $o=(S,T;F')$是一个出现网,h 是一个映射.分支进程 Π的可能扩展集是(t,B)对的集合,记为 $PE(\Pi)$,其中 B 是 Π中条件集 S 的 co 关系集（co-set）,t 是过程模型 p 的一个变迁,其中：

（1）$h(B)=\bullet t$,B 映射到过程模型 p 中是 t 的前集;

（2）$h(e)=t$ 且$\cdot e=B$,则 B 不包含 e.

定义11　（配置）：一个出现网 $o=(P,T;G)$的配置 C 是一组变迁的集合,其中：

（1）$t\in C\Rightarrow\forall t'<\bullet t$：$t'\in C$,即配置 C 是由适当偏序组成的闭包;

（2）$\forall t,t'\in C$：$\neg(t\#t')$,即配置 C 中的任意元素是无冲突的.

对于一个出现网 $o=(P,T;G)$,$\forall t\in T$,变迁 t 的本地配置也是一组包含 t 本身,并且是由适当偏序组成的无冲突的变迁集合,记为$[t]$,满足 $t\in[t]$,且 $t_1\in[t]$,t_1 满足$[t_1]<\bullet[t]\wedge$任意 $t_2\in C$ 有 $\neg t_2\#t_1$.

定义12　（适当偏序关系）：一个出现网 $o=(P,T;G)$的上的适当偏序关系是本地配置之间的关系,用$<\bullet$表示适当偏序关系,其中：

（1））$<\bullet$ 是对$\subset$的改进,即$\forall t,t'\in T$ 有$[t]\subset[t']$,则$[t]<\bullet[t']$;

（2）$<\bullet$通过有限扩展得到保留,即$\forall t,t'\in T$ 有$[t]<\bullet[t']$和 $Mark(t)=Mark(t')$,则对于$[t]$的有穷扩展$[t]\oplus E$,存在一个同构变换 I,使$[t]\oplus E<\bullet[t']\oplus I(E)$.

定义13　（割集）：一个出现网 $o=(P,T;G)$的有限配置 C,其割集 $Cut(C)$,是一组只包含 co 关系的条件集,$Cut(C)$被定义为 $Cut(C)=(Min(o)\cup C\bullet)\setminus\bullet C$.

一个出现网 $o=(P,T;G)$的有限配置 C,$h(Cut(C))$的位置集是一个可达标记,表示条件集合 $Cut(C)$所能到达的事件,用 $Mark(C)$表示 $h(Cut(C))$.

定义14 **（截断事件）**：二元组 $\Pi=(o,h)$是过程模型 $p=(C, A; F, M_0)$的一个分支进程,其中 $o=(S,T;F')$是一个出现网,h 是一个映射.如果 Π中存在事件 e'使得另一个事件 e 是一个截断事件时,记为 $corr(e)=e'$,满足：

（1）$Mark([e])=Mark([e'])$,即通过 e 到达的情态与 e'的相等;

（2）$[e'] < \bullet [e]$,本地配置$[e']$和本地配置$[e]$是适当偏序关系.

将 e 到达的情态 $Cut([e])$称为截断条件 p,对于任意截断条件,其后面的元素都被截断了,即任意截断条件 $p\bullet=\varnothing$.

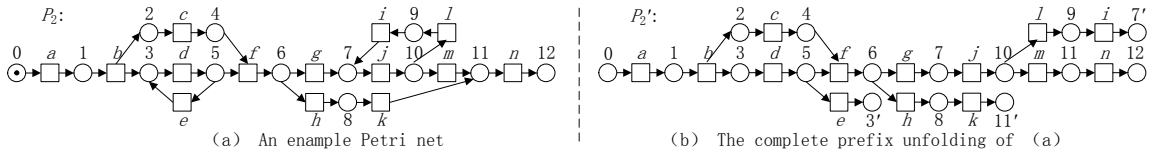


Fig.5 The complete prefix unfolding of case

图5 完全前缀展开案例

例图 5（a）中的 Petri 网,图 5（b）是其完全前缀展开,也是该网的一个分支进程,其中有配置[a]={a},[e]={a, b, d, e},[f]={a, b, c, d, f},而割集 $Cut([a])=\{1\}$, $Cut([b])=\{2, 3\}$.针对事件 b 与 e,具有 $Mark([b])=Mark([e])=\{3\}$和 $[b]< \bullet [e]$（[b]={a, b}, [e]={a, b, d, e}）,事件 e 为截断事件,其对应事件是 $corr(e)=b$.同样对于事件 g 与 i,有 $Mark([i])=Mark([j])=\{7\}$和$[g]< \bullet [i]$（[g]={a, b, c, d, f, g}, [i]={ a, b, c, d, f, g, j, l, i}）,因此有 $corr(i)=g$.

3 行为等价过程树生成

本节分为 3 部分:首先对行为等价过程树生成的总体流程进行了介绍;然后详细阐述总体流程中基于完全有限前缀展开的活动关系判断方法,为下文的模型重构提供了实现基础;最后阐述模型重构方法及模型重构的先决条件.

3.1 总体流程

基于完全有限前缀展开的行为等价过程树生成算法,该算法的主要步骤包括六步,具体如图 6 所示.

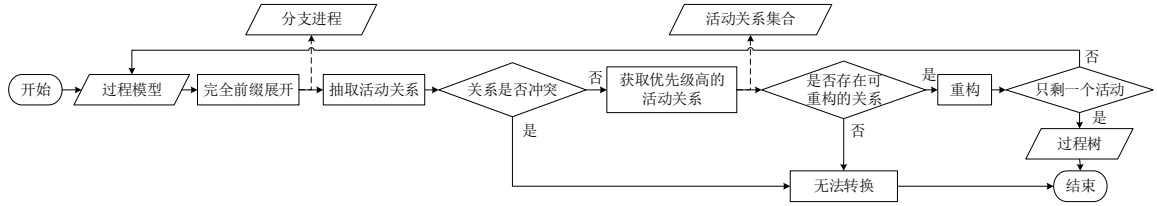


Fig.6 The overall process of generating algorithm for the behavior equivalent process tree

图6 行为等价过程树生成算法总体流程

第一步,完全前缀展开,将给定的过程模型进行完全前缀展开,得到为一个包含截断事件的分支进程.

第二步,抽取活动关系,根据模型及其分支进程,判断每组活动之间的关系,并将其保存到活动关系矩阵中.

第三步,判断关系间是否存在冲突,若活动关系间存在冲突,则该过程模型无法转化为行为等价的过程树,算法停止;若活动关系间无冲突,则进行下一步.

第四步,通过比较活动关系优先级,获得优先级高的活动关系集合.

第五步,过程模型重构,若当前存在可重构的活动关系,则依次进行重构,否则算法停止.

第六步,若存在多个活动,则跳转至第一步;否则输出给定过程模型所对应的行为等价过程树.

过程树生成算法的伪代码如算法 1 所示,它首先抽取活动关系,然后判断这些活动关系之间是否冲突,选择优先级最高的活动关系,将该组活动重构,生成新的过程模型.不停的将过程模型进行重构,当过程模型只剩下一个活动时,这个活动的名称就是过模型程 p 对应的过程树的前缀表达式.

算法1：过程树生成算法

输入：过程模型 $p=(C, A; F, M_0)$

输出：过程树

步骤：

```
processToPTree(p)
1.  flag = true                                  //变量 flag 用于判断每次循环时是否抽取出活动关系
2.  while flag and p.A.size()>1 then
3.      flag = false
4.      get a relationship matrix RM with IdentifiesRelation(p)   //计算过模型程 p 对应的活动关系矩阵 RM
5.      if isConflictRelation(RM) then          //判断过模型程 p 活动关系是否冲突,判断方法参考 3.3 节
6.          return ∅
7.      end if
8.      RL = getRelationList(RM)        //获取优先级最高的活动关系,方法参考 3.3 节
9.      for each ◎(a1, a2) ∈ RL then
10.         if {a1, a2}⊂ p.A then
11.             p = refactoringOfProcess(p,◎, a1, a2)     //重构过模型程 p 中的活动 a1, a2
12.             flag = true
13.         end if
14.     end
15. end while
16. if p.A.size==1 then
17.     return p.A
18. end if
19. return ∅
```

3.2 活动关系判断

在建立行为等价过程树时,判断过程模型的活动关系非常关键,因此需要对活动关系的判断条件进行准确定义.本文中使用了顺序（→）,选择（×）,并发（||）,迭代（∝）四种关系来描述模型行为.为了构建一棵完整的行为等价过程树,本文抽取出的活动关系需满足两个条件：(1)该活动关系属于四种关系中的一种;(2)所对应的两个活动合并后不改变其他任何行为的活动关系.下面给出了满足该条件的这四种关系的判断条件.

定义15　（顺序关系判断条件）：对于过程模型 $p=(C, A; F, M_0)$, Π=（o,h）为过程模型 p 对应的一个包含截断事件的分支进程,其中,$o=(S,T;F')$是一个出现网,h 是一个映射函数.$\forall t_1,t_2\in T$, $\forall a_1,a_2\in A$, $h(t_1)=a_1$, $h(t_2)=a_2$, 当 t_1,t_2 满足下述条件时,a_1 与 a_2 存在可重构的顺序关系,a_1 先发生,记为$\rightarrow(a_1, a_2)$.

（1）$[t_1]=[t_2]-\{t_2\}$,即 t_1 的配置与 t_2 的配置相比只相差 t_2;

（2）$h(t_2)\notin h([t_1])$,即本地配置$[t_1]$映射到过程模型 p 中的活动集合不包含活动 a_2.

（3）$\forall t\in T$ 且 $t\neq t_1\wedge t\neq t_2$,当 $h(t)=h(t_2)$时,必有 $h(\bullet\bullet t)=h(t_1)$,即只有 a_1 发生后 a_2 才可以发生.

（4）$\forall t\in T$ 且 $t\neq t_1\wedge t\neq t_2$,当 $corr(t)=t_1$ 时,必有 $h(t)=h(t_1)\wedge[t_2]=[t]-\{t\}$,即 a_1 与其他活动间不存在迭代关系.

（5）$\forall t\in T$ 且 $t\neq t_1\wedge t\neq t_2$,必有 $\{t_1, t_2\}\cap[t]=\emptyset$ 或 $\{t_1, t_2\}\cap[t]=\{t_1, t_2\}$.

定义16　（迭代关系判断条件）：对于过程模型 $p=(C, A; F, M_0)$, Π=（o,h）为过程模型 p 对应的一个包含截断事件的分支进程,其中,$o=(S,T;F')$是一个出现网,h 是一个映射函数.$\forall t_1,t_2\in T$, $\forall a_1,a_2\in A$, $h(t_1)=a_1$, $h(t_2)=a_2$,

当 t_1,t_2 满足下述条件时,a_1 与 a_2 存在可合并的迭代关系,其中 a_1 先发生,记为$\propto(a_1, a_2)$,a_2 先发生,记为$\propto(a_2, a_1)$.

(1) $[t_1]=[t_2]-\{t_2\}$,即 t_1 的配置和 t_2 的配置只相差 t_2;

(2) $[corr(t_2)]=[t_1]-\{t_1\}$,即 t_2 是一个截断事件,且它的对应事件 $corr(t_2)$的配置比$[t_1]$少一个 t_1;

(3) 若 $h(t_2)\neq h(corr(t_2))$,则 a_1 与 a_2 存在可合并的迭代关系,且 a_1 先发生;否则 a_2 与 a_1 存在可合并的迭代关系,且 a_2 先发生;

定义17 **(选择关系判断条件)**:对于过程模型 $p=(C, A; F, M_0)$, $\Pi=(o,h)$ 为过程模型 p 对应的一个包含截断事件的分支进程,其中,$o=(S,T;F')$是一个出现网,h 是一个映射函数.$\forall t_1,t_2\in T$, $\forall a_1,a_2\in A$, $h(t_1)=a_1$, $h(t_2)=a_2$, 当 t_1,t_2 满足下述条件时,a_1 与 a_2 存在可合并的选择关系,记为$\times(a_1, a_2)$或$\times(a_2, a_1)$.

(1) $\bullet t_1\cap \bullet t_2\neq\varnothing$,事件 t_1 与事件 t_2 的前驱是相交的;

(2) $[t_1]-\{t_1\}=[t_2]-\{t_2\}$,即 t_1 和 t_2 的发生条件是相同的;

(3) $corr(t_1)=t_2$ 或 $corr(t_2)=t_1$,即 t_1 和 t_2 中有一个是截断事件且对应事件为对方.

定义18 **(并发关系判断条件)**:对于过程模型 $p=(C, A; F, M_0)$, $\Pi=(o,h)$ 为过程模型 p 对应的一个包含截断事件的分支进程,其中,$o=(S,T;F')$是一个出现网,h 是一个映射函数.$\forall t_1,t_2\in T$, $\forall a_1,a_2\in A$, $h(t_1)=a_1$, $h(t_2)=a_2$, 当 t_1,t_2 满足下述条件时,a_1 与 a_2 存在可合并的并发关系,记为$||(a_1, a_2)$或$||(a_2, a_1)$.

(1) $\bullet t_1\cap \bullet t_2=\varnothing$,事件 t_1 与事件 t_2 的前驱不相交;

(2) $[t_1]-\{t_1\}=[t_2]-\{t_2\}$;

(3) 对于所有 $t\in T$ 且 $t\neq t_1\wedge t\neq t_2$,必有$\{t_1, t_2\}\cap[t]=\varnothing$ 或$\{t_1, t_2\}\cap[t]=\{t_1, t_2\}$.

Table 1 The event information for the case in figure 5

表1 图 5中案例的事件信息

事件	配置	前驱	后继	是否为截断事件	对应事件
a	{a}	{0}	{1}		
b	{a, b}	{1}	{2, 3}		
c	{a, b, c}	{2}	{4}		
d	{a, b, d}	{3}	{5}		
e	{a, b, d, e}	{5}	{3'}	Yes	b
f	{a, b, c, d, f}	{4, 5}	{6}		
g	{a, b, c, d, f, g}	{6}	{7}		
h	{a, b, c, d, f, h}	{6}	{8}		
i	{a, b, c, d, f, g, i, j, l}	{9}	{7'}	Yes	g
j	{a, b, c, d, f, g, j}	{7}	{10}		
k	{a, b, c, d, f, h, k}	{8}	{11'}	Yes	m
l	{a, b, c, d, f, g, j, l}	{10}	{9}		
m	{a, b, c, d, f, g, j, m}	{10}	{11}		
n	{a, b, c, d, f, g, j, m, n}	{11}	{12}		

例如表 1 是图 5 中完全前缀展开案例是事件信息,根据表 1 中的信息可知活动 d 和 e 之间存在迭代关系,a 和 b、h 和 k、l 和 i 之间存在顺序关系.

通过本节的四种关系的判断条件,可以构造出活动关系抽取算法 *IdentifiesRelation*,它的基本步骤如下.

第一步,将过程模型展开获得一个包含截断事件的分支进程 $\Pi=(o,\ h)$,其中 $o=(S,T;F')$是一个出现网,h 是一个映射函数.

第二步,构造一个二元矩阵 *RM* 以方便读取和保存活动间的关系,*RM* 的行数与列数都等于 p 中的活动数.

第三步,利用二次嵌套循环判断任意 $t_1,t_2\in T\wedge h(t_1)\neq h(t_2)$间的关系,具体判断方法可参考定义 15 至定义 18,当其存在某种关系时将其保存到二元矩阵 *RM* 中.

3.3 模型重构

冲突判断条件对应算法的总体流程中的第三步,判断关系之间是否存在冲突.如果关系之间存在冲突,则该过程模型就是非 TEPM,不能转换为行为等价的过程树,下面给出冲突的判断条件的定义.

定义19 **(冲突的判断条件)**:对于过程模型 $p=(C, A; F, M_0)$,***RM*** 是过程模型 p 的活动关系集合,当任意 $a_1,a_2\in A$,满足下述条件时,***RM*** 中的活动关系不冲突.

（1）a_1,a_2无关系或只存在一种关系;

（2）当→(a_1, a_2)存在时,a_1和 a_2与任意 $a\in p.A\wedge a\neq a_1\wedge a\neq a_2$间可以存在的关系只有→($a$, a_1)或→(a_2, a);

（3）当∝(a_1, a_2)存在时,a_1和 a_2与于任意 $a\in p.A\wedge a\neq a_1\wedge a\neq a_2$之间不能存在迭代关系;

顺序关系、迭代关系是有序的.顺序关系只要一次遍历,就可以抽取所有活动的顺序关系,但当多个活动重构后,新的活动汇集了旧活动的直接前驱,可能出现新的顺序关系.

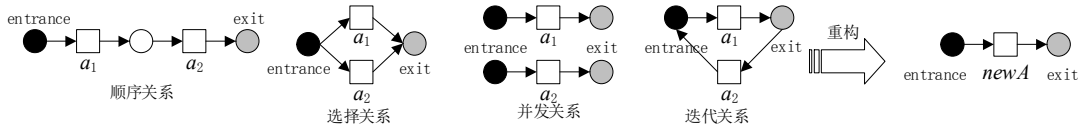


Fig.7 Activity relation reconstruction

图7 活动关系重构

接下来需要选择优先重构的关系,将两个活动重构为一个过程树后,会丢失一部分结构信息.活动重构后会缺少部分结构信息,再次抽取新的关系时并不会和其他原有关系产生冲突,甚至重构前活动和原有关系的活动的冲突会消失,因此需确定活动关系间优先级,以确保活动重构后不会消除被重构相对活动与其他活动关系.

定义20 （活动关系优先级定义）：对于本文中使用四种描述行为关系,其优先级为：顺序（→）>选择（×）>并发（||）>迭代（∝）.

首先,优先级最高的是顺序关系,从图 7 中可以看出顺序关系重构后,活动 t_1和活动 t_2之间的库所和流关系被消除,t_1和 t_2重构后与外部活动的关系并未改变,因此顺序关系的优先级最高.其次是选择关系,选择关系重构后,活动 t_1和活动 t_2与外部活动之间的流关系被合并,因此选择关系的优先级低于顺序关系.接下来是并行选择关系,并行关系重构后,不仅活动 t_1和活动 t_2与外部活动之间的流关系被合并,与活动 t_1和活动 t_2相邻的库所也被合并,因此并行关系的优先级低于选择关系.最后对于迭代关系,该关系会导致一些活动重复执行,重构迭代关系时会合并不同方向的流关系,因此将迭代关系的优先级最低.

过程模型重构思路如图 7 所示,它删除原过程模型中的某一组活动 a_1和 a_2,添加了重构后的新活动 *newA*,这个新活动 *newA* 用活动 a_1和 a_2的前缀表达式命名.然后根据更新后的活动集合,删除了一些不会影响活动间关系的条件.其次根据更新后的活动集合和条件集,来决定保留哪些流关系,注意到虽然新活动 *newA* 汇集了旧的活动 a_1和 a_2的所有流关系,但是这些流关系仍然需要判断是否需要保留.最后,返回新的过程模型.

过程模型重构算法 *reconstruction* 的伪代码如算法 2 所示,其主要步骤可分为五步,具体如下.

第一步,构造新活动 *newa*,构造需要删除的活动集 *delA*,将活动 a_1和 a_2加入 *delA*;

第二步,构造需删除的条件集 *delC*,加入其中的条件需要满足(1)该条件同时连接了 a_1和 a_2,且不与其他活动产生流关系;(2)该条件不能是初始情态.其中(1)是为了获得仅和 a_1和 a_2产生联系的条件;(2)针对一些特殊情况,例如在迭代关系中,c 可能作为一个初始情态,但同时又不与其他活动产生流关系仅连接 a_1和 a_2;

第三步,构造需删除的流关系集 *delF*,将与 *delA* 和 *delC* 相关的流关系加入 *delF* 中;

第四步,构造流关系集 *newF*,有选择的令 *newa* 继承与 a_1和 a_2相关的流关系,将其加入 *newF* 中.这一步分为两种情形(1)当前重构的关系是迭代关系时,只继承与 a_1相关的流关系,但要注意到有时会出现的 $a_1\bullet$属于 *delC* 情况,例如表 4 中的第 16 个案例,这时为保证 *newF* 存在后继,需新建 *newa* 到 $a_2\bullet$的流关系;(2)当前重构的关系不是迭代关系时同时继承与 a_1和 a_2相关的流关系.但在某些结构中 a_1和 a_2可能同时和某个条件产生不同向的流关系,都保留会使 *newa* 和该条件产生双向流关系,因此需排除这种情况.另外两种情形都要排除 *delC* 的影响.

第五步,在原过程模型中删除 *delA*、*delC* 和 *delF*,并增加 *newa* 和 *newF* 从而构建新的过程模型.

算法2：过程模型重构算法

输入：过程模型 $p=(C, A; F, M_0)$,a_1,a_2需要重构的活动,◎是 a_1和 a_2间的关系

输出：重构后的过程模型 p'

步骤：

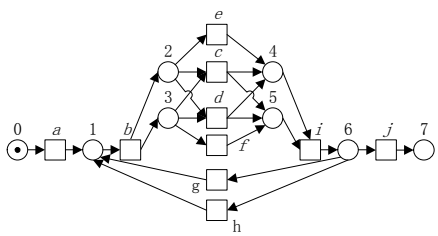
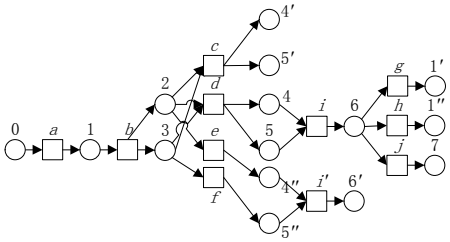
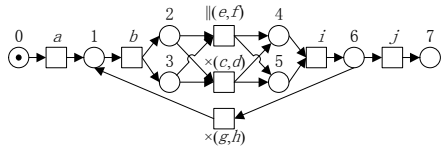
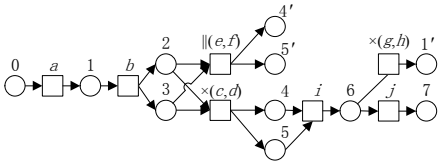
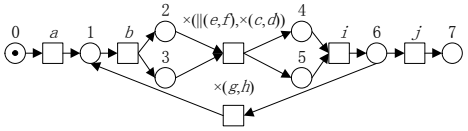
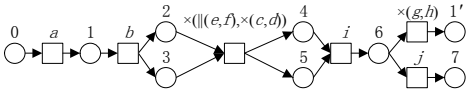
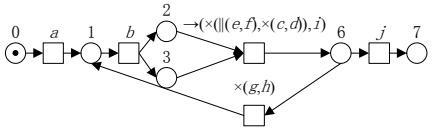
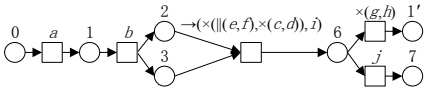
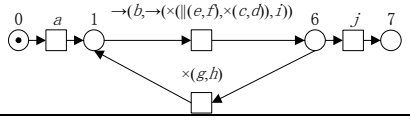
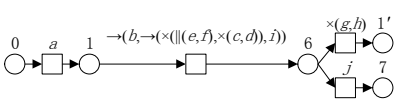
reconstruction $(p, ◎, a_1, a_2)$

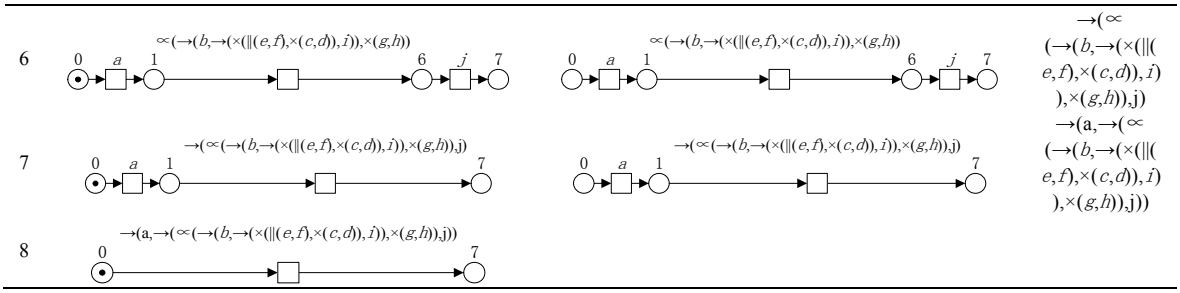
1. $newa = ◎(a_1, a_2)$ // *newa* 和 *newF* 分别代表新增的活动和流关系集
2. $delA = \{a_1, a_2\}$ // *delA*、*delC* 和 *delF* 分别代表需要删除的活动集、条件集和流关系集
3. $delC = \{ c \mid c \in p.C$ **and** $\bullet c \neq \varnothing \wedge c \bullet \neq \varnothing \wedge \bullet c \subseteq delA \wedge c \bullet \subseteq delA \wedge c \notin p.M_0\}$
4. $delF = \{ e_1 \times e_2 \mid e_1 \times e_2 \in p.F$ **and** $\{e_1, e_2\} \cap (delA \cup delC) \neq \varnothing \}$
5. **if** $◎ == \propto$ **then**
6. $newF = \{newa \times c \mid c \in ((a_1 \bullet \subseteq delC ? a_2 \bullet : a_1 \bullet) - delC) \} \cup \{c \times newa \mid c \in (\bullet a_1 - delC)\}$
7. **else then**
8. $newF = \{newa \times c \mid c \in ((a_1 \bullet + a_2 \bullet) - (\bullet a_1 + \bullet a_2) - delC)\} \cup \{c \times newa \mid c \in ((\bullet a_1 + \bullet a_2) - (a_1 \bullet + a_2 \bullet) - delC)\}$
9. **end if**
10. **return** $p' = (p.C - delC, p.A - delA + \{newa\}; p.F - delF + newF, p.M_0)$

表 2 中展示了一个 TEPM 转化为行为等价过程树生成的案例,同时该例也体现了算法对复杂结构的有效处理. 表 2 中第一列代表了当前的循环次数,第二列代表了该次循环中的模型,第三列是其对应的展开网,第四列是代表需要进行重构的活动.第四列中显示了需要进行重构的活动,并不等于该次循环只检测出了这些活动,例如在第 8 次迭代中,可以同时判断出活动$\propto(\rightarrow(b,\rightarrow(\times(\|(e,f),\times(c,d)),i)),\times(g,h))$与活动 j 之间的顺序关系，和活动 a 与活动$\propto(\rightarrow(b,\rightarrow(\times(\|(e,f),\times(c,d)),i)),\times(g,h))$之间的顺序关系,但是这两种关系显然只能选择一个进行重构.

Table 2 Process tree generation case

表2 过程树生成案例

循环	Petri 网	展开网	重构活动
1			$\|\|(e,f)$、$\times(c,d)$、$\times(g,h)$
2			$\times(\|\|(e,f),\times(c,d))$
3			$\rightarrow(\times(\|\|(e,f),\times(c,d)),i)$
4			$\rightarrow(b,\rightarrow(\times(\|\|(e,f),\times(c,d)),i))$
5			$\propto(\rightarrow(b,\rightarrow(\times(\|\|(e,f),\times(c,d)),i)),\times(g,h))$



4 实验分析

本节主要从两个方面来验证基于完全有限前缀展开的行为等价过程树生成算法：(1)通过多个测试用例来测试算法的可行性和有效性;(2)通过对算法的性能进行分析,以及和现有过程树生成算法的对比,体现文算法的优势.本节所有的实验都在 Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz 且具有 8G RAM 的 PC 平台上完成.

4.1 实验数据

实验采用过程日志生成器(Process Log Generator, PLG[26])生成的 100 个不同的过程模型,这些过程模型包括本文中提到的 4 种基本结构,例如图 8 就是通过 PLG 产生的一个例子.通过 PLG 产生的过程模型记为 RD 组;构造 20 个与过程树等价的具有复杂结构的 TEPM,记为 ED 组.这两组模型作为测试用例进行测试.RD 组模型以及本文算法的源代码已经上传至 Github[1].

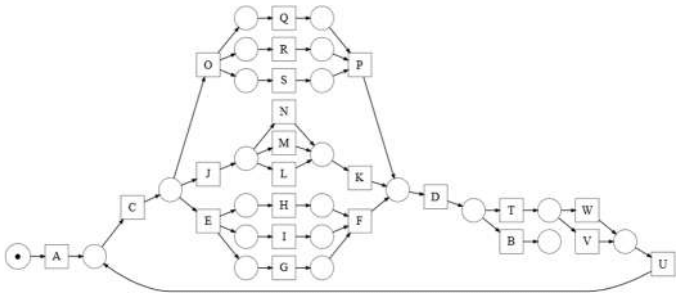


Fig.8 A sample of process model generated by PLG

图8 PLG 生成的过程模型例子

4.2 准确性分析

为了验证算法在处理复杂结构转化问题上的有效性,首先建立 20 个较为复杂的人工案例。具体案例如表 3 所示,其中（a）列是原模型,（b）列是与原模型行为等价的过程树(为了更直观的表示模型行为,本文将表 3 中（b）列的二叉树转换为多叉树).这 20 个人工案例的设计思路主要可以分为两种:(1)对一些经典案例进行修改,例如 Courier Protocol、AccidentColoured 和 Coloured Reader Writer 等,具体为编号 1-8 的案例;(2)四种行为关系随机叠加组合获得的一系列模型,并从中挑选的具有代表性的案例,具体为编号 9-20 的案例.可以发现将其转化为过程树后,可以很容易的观察到它的整体行为结构.

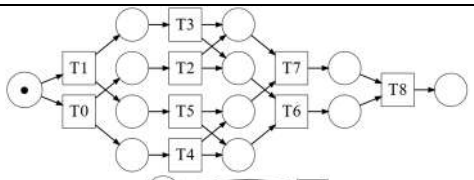
Table 3 Group ED experimental case

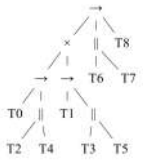
表3 ED 组实验案例

编号	（a）实验模型	（b）行为等价的过程树

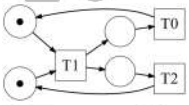
1 https://github.com/xgyxhy/ProcessModel

(1)

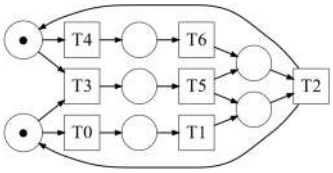


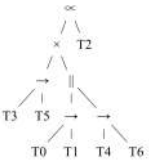


(2)

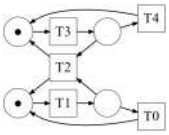


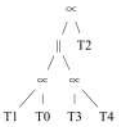
(3)



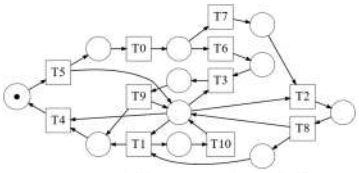


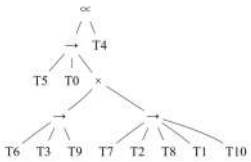
(4)



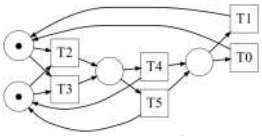


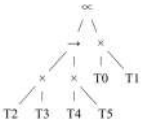
(5)



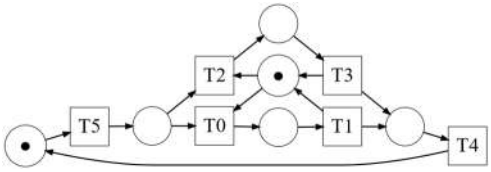


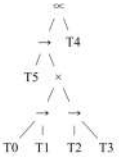
(6)



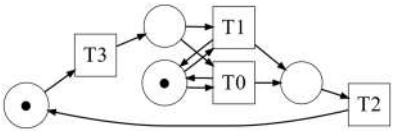


(7)

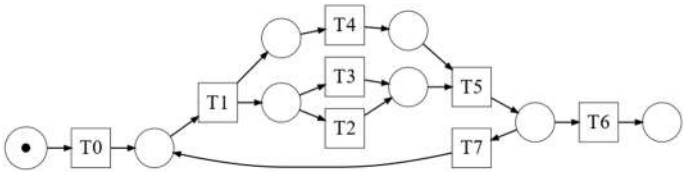


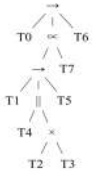


(8)

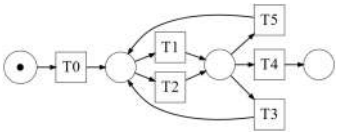


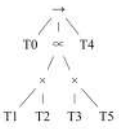
(9)



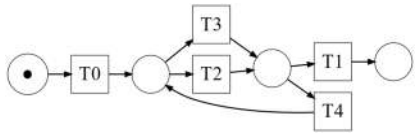


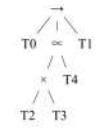
(10)



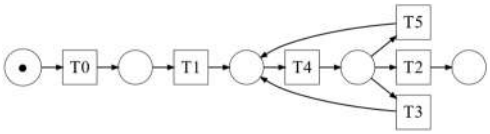


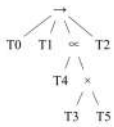
(11)



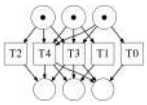


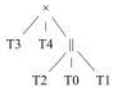
(12)



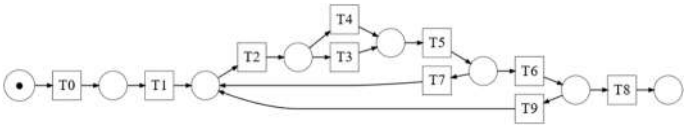


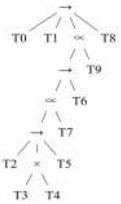
(13)



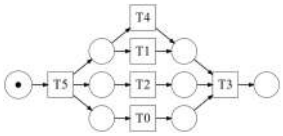


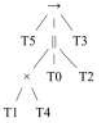
(14)



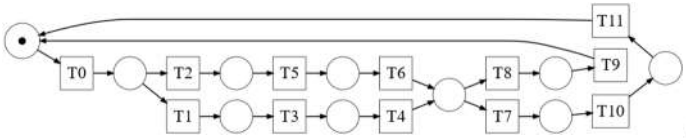


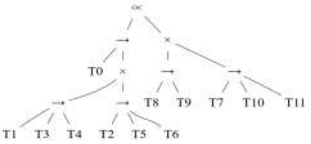
(15)



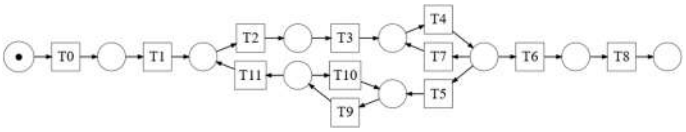


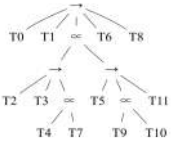
(16)



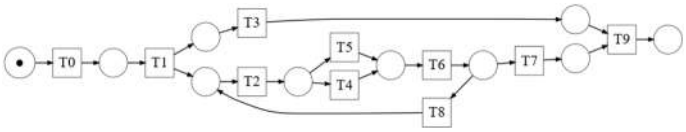


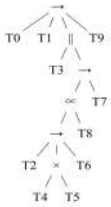
(17)



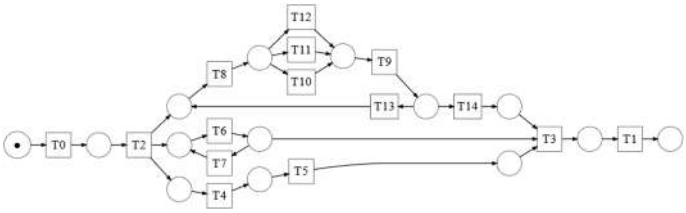


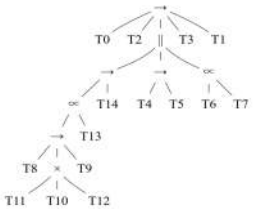
(18)



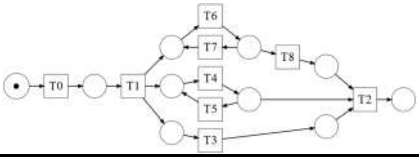


(19)





(20)



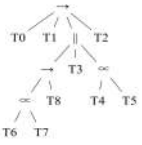
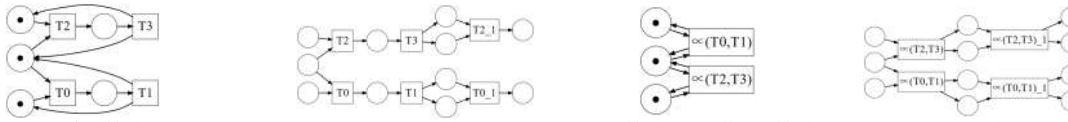


Table 4 Comparison of Group ED experimental case

表4 ED 组实验案例对比

	是否能转化				是否存在行为损失				是否将结构泛化为花模型				是否将结构泛化为块结构				是否能处理案例中的循环结构			
	本文	文献[19]	文献[23,24]	文献[27]	本文	文献[19]	文献[23,24]	文献[27]	本文	文献[19]	文献[23,24]	文献[27]	本文	文献[19]	文献[23,24]	文献[27]	本文	文献[19]	文献[23,24]	文献[27]
(1)	✓	×	✓	✓	✓	⊥	✓	×	×	⊥	✓	×	×	⊥	×	✓	∅	∅	∅	∅
(2)	✓	×	✓	×	✓	⊥	✓	⊥	×	⊥	✓	⊥	×	⊥	×	⊥	✓	⊥	✓	⊥
(3)	✓	×	✓	×	✓	⊥	✓	⊥	×	⊥	✓	⊥	×	⊥	×	⊥	✓	⊥	✓	⊥
(4)	✓	×	✓	×	✓	⊥	✓	⊥	×	⊥	✓	⊥	×	⊥	×	⊥	✓	⊥	✓	⊥
(5)	✓	×	✓	×	✓	⊥	✓	⊥	×	⊥	✓	⊥	×	⊥	×	⊥	✓	⊥	✓	⊥
(6)	✓	×	✓	×	✓	⊥	✓	⊥	×	⊥	✓	⊥	×	⊥	×	⊥	✓	⊥	✓	⊥
(7)	✓	×	✓	×	✓	⊥	✓	⊥	×	⊥	✓	⊥	×	⊥	×	⊥	✓	⊥	✓	⊥
(8)	✓	×	✓	×	✓	⊥	✓	⊥	×	⊥	✓	⊥	×	⊥	×	⊥	✓	⊥	✓	⊥
(9)	✓	✓	✓	×	✓	×	×	⊥	×	×	×	⊥	×	×	×	⊥	✓	✓	✓	⊥
(10)	✓	✓	✓	×	✓	×	×	⊥	×	×	×	⊥	×	×	×	⊥	✓	✓	✓	⊥
(11)	✓	✓	✓	×	✓	×	×	⊥	×	×	×	⊥	×	×	×	⊥	✓	✓	✓	⊥
(12)	✓	✓	✓	×	✓	×	×	⊥	×	×	×	⊥	×	×	×	⊥	✓	✓	✓	⊥
(13)	✓	×	✓	✓	✓	⊥	✓	×	×	⊥	✓	×	×	⊥	×	✓	∅	∅	∅	∅
(14)	✓	✓	✓	×	✓	×	×	⊥	×	×	×	⊥	×	×	×	⊥	✓	✓	✓	⊥
(15)	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×	×	×	✓	∅	∅	∅	∅
(16)	✓	✓	✓	×	✓	×	×	⊥	×	×	×	⊥	×	×	×	⊥	✓	✓	✓	⊥
(17)	✓	✓	✓	×	✓	×	×	⊥	×	×	×	⊥	×	×	×	⊥	✓	✓	✓	⊥
(18)	✓	✓	✓	×	✓	×	×	⊥	×	×	×	⊥	×	×	×	⊥	✓	✓	✓	⊥
(19)	✓	✓	✓	×	✓	×	×	⊥	×	×	×	⊥	×	×	×	⊥	✓	✓	✓	⊥
(20)	✓	✓	✓	×	✓	×	×	⊥	×	×	×	⊥	×	×	×	⊥	✓	✓	✓	⊥

表 3中（a）列模型都是 TEPM,使用现有的方法[19,23,24]无法将其都转化为行为等价过程树,针对这些模型采用本文的算法且可以从中抽取出行为等价过程树.表 4 比较了包括本文在内的四种过程树生成的方法对于表 3 中案例的处理.其中,文献[19]通过识别基本块可以将 BSPM 转化为行为等价过程树,但不能处理非 BSPM;文献[22,23]可以将 BSPM 和非 BSPM 都转化为过程树,但是其在处理非 BSPM 时会将其泛化为花模型;文献[27]并非生成过程树的算法,不过它可以将非 BSPM 转换为 BSPM,然后在通过识别基本块就可以得到相应的过程树,但是该方法不能针对循环结构.表 4 中“✓”和“×”分别代表是“是”和“否”.文献[19]和文献[17]不能转换某些模型进而无其它指标值,使用“⊥”表示.案例(1)、(13)和(15)是不包含循环结构的,使用“∅”符号表示.从表 4 中可以看出本文算法相较于其他几种方法在处理复杂结构和循环结构更有优势,能更好的保留模型行为.



（a）实验模型 （b）与（a）对应的展开网 （c）模型（a）重构后的模型 （d）与（c）对应的展开网

Fig.9 Examples of process models that cannot be transformed into process trees

图9 不能转化为过程树的过程模型案例

图 9 展示了一些非 TEPM 的例子，这些例子是不能转化为过程树的过程模型案例,其中（a）是原模型,（b）是原模型的一个展开网,（c）是重构后的模型,（d）是（c）的一个展开网.在案例中,对展开网（b）进行检测,可以发现事件 T3 和 T2_1、T1 和 T0_1 满足可重构的迭代关系判断条件,这时可抽取出∝(T0,T1)和∝(T2,T3).∝(T0,T1)和∝(T2,T3)并不冲突,将其重构后可得到图 14（c）,将其展开可得到（d）的展开网,此时展开网（d）上无法检测出新的关系,而图 14（c）上的活动尚未全部重构为一个因此图 9 中的案例二不能转化为过程树.

综合上述实验结果,基于完全有限前缀展开的等价过程树生成算法将过程模型进行展开,然后对模型进行活动关系判断,对符合可重构关系的活动进行重构,更新过程模型,以此迭代,直到所有可重构的关系重构完毕,该方法能将与过程树等价的过程模型转化为过程树.

4.3 性能分析

在验证本文算法性能之前，首先分析算法的时间复杂度。本文的行为等价过程树生成算法采用完全前缀展开分析模型,然后抽取模型的行为,在获得优先级高的活动关系集合,从而重构过程模型,通过不断重复这些操作最终获得行为等价过程树.本文算法的时间复杂度由算法 IR 的复杂度以及迭代次数（过程树深度）决定，其中算法 IR 由完全前缀展开算法和计算活动关系两部分构成.首先考虑最坏情况下算法的复杂度,完全前缀展开算法的复杂度最坏情况下为 $O(|A|\cdot R^{\zeta})$,其中$|A|$是模型中的活动数量,R 是展开网中非截断条件的数量,ζ是活动的出度和入度中的最大数.计算活动关系的复杂度是稳定的为 $O(|T|^3)$,$|T|$是展开网中的事件数.因此本文算法复杂度最坏为 $O(h(|T|^3+|A|\cdot R^{\zeta}))$,其中 h 为过程树的深度.但通常情况下完全前缀展开算法的复杂度为 $O(R)$,此时完全前缀展开算法的复杂度就远低于计算活动关系的复杂度 $O(|T|^3)$,所以本文算法的时间复杂度通常由迭代次数和计算活动关系的复杂度决定,具体为 $O(|T|^3\cdot h)$.相比同样可以处理 TEPM 结构的经典文献[27]达到指数级的时间复杂度 $O((|C|/n)^n)$（$|C|$是模型中的条件数量,n 是活动的入度中的最大数）,本文算法的时间复杂度在通常情况下是非常低的.

对于 RD 组实验数据,经过验证可以全部正确的转化为行为等价过程树。现将统计抽取过程树花费的时间,以及对应过程模型的活动数的统计图展示如下图 10.图 11 显示了 RD 组实验模型平均活动耗时统计,从该图可以看出文本的算法将 BSPM 转化为过程树的效率稳定,平均到每个活动的耗时趋于 0.5ms.

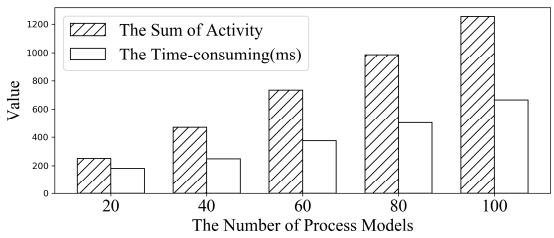


Fig.10 RD group experimental efficiency statistics
图10 RD 组实验效率统计

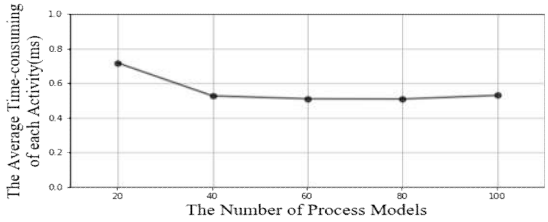


Fig.11 Statistics of average activity time of RD group experiment
图11 RD 组实验平均活动耗时统计

根据本文中的算法已经开发了本实验的原型系统[1],原型系统中的过程树生成功能就是利用基于完全有限前缀展开的等价过程树生成算法实现的.

5 相关算法对比

过程模型的分析研究已经成为现代组织用于管理复杂运作流程的重要手段,其中对于过程模型的行为研究是该领域极其重要的研究内容,文中的算法的核心也是在于如何分析模型行为,表 5 给出了在判断模型行为研究上取得一定成果的文献内容.

Table 5 Related Research on Model Behavior Representation
表5 模型行为表示的相关研究

来源	方法	描述整体模型行为	直接描述任意活动间行为	描述 TEPM	描述循环结构	图形描述	文字描述
文献[28]	广度优先拓扑序列 BFTS			✓	✓		✓
文献[29]	有代表性的 trace 集合			✓		✓	
文献[30]	任务紧邻关系集			✓		✓	✓
文献[31]	完整触发序列			✓	✓	✓	✓
文献[32]	BQL(Behavior Query Language)		✓	✓	✓		✓
文献[33]	The APQL grammar trees	✓			✓		✓
文献[34]	The log-tree representation	✓		✓		✓	

1 http://120.79.58.61:8080/ProcessModel/Jump?type=processModelling

文献[16]	基于遗传算法的过程树	✓	✓		✓	✓	✓
文献[19]	基于块结构的过程树	✓	✓		✓	✓	✓
文献[27]	Bpstruct	✓		✓		✓	
文献[35]	CASS	✓		✓	✓	✓	
本文	**基于完全有限前缀展开的过程树**	✓	✓	✓	✓	✓	✓

表 5 展示了模型行为研究一些代表性成果,其中,“描述整体模型行为”是指可以通过该方法直接观察整体活动的行为,“直接描述任意活动间行为”是指能直接获取模型中任意活动的关系,“描述 TEPM”特指是否能在不损失模型行为的情况下处理本文中定义的 TEPM 模型,“描述循环结构”是指 TEPM 中的出现的循环路径的模型,“图形描述”是指该模型行为表示方法可以通过图来描述模型,“文字描述”是指该模型行为表示方法可以通过文字来描述模型.

从表 5 中可以得出现有的模型行为表示方法对于描述整体模型行为和描述局部（模型中任意活动）行为难以兼得.本文使用过程树不仅能同时获得模型的整体模型行为和部分行为,还具有相应的图形和文字描述.文献[16]、文献[19]和本文一样使用过程树表示行为,但无法描述复杂行为.因此,本文提出了基于完全有限前缀展开的行为等价过程树生成算法,用于在不牺牲模型行为的条件下将 TEPM 模型转换为行为等价过程树.

6 结束语

本文设计一个基于完全有限前缀展开的行为等价过程树生成算法.该算法利用完全前缀展开算法对模型进行分析,判断出模型的行为.然后根据模型的行为判断该模型是否与过程树的行为等价,如果等价就选择优先级高的关系生成与模型是部分行为等价子过程树,并以此迭代直到生成行为等价过程树.一方面,该算法利用基于完全前缀展开对具有复杂结构的模型的活动关系进行判断,这种方法避免了“状态空间爆炸”问题.另一方面,该算法能够判断一些传统方法无法判断的模型行为,使其能够将具有复杂结构的过程模型转化为行为等价的过程树.同时该方法还可以通过对基本关系进行扩充,提高具有复杂结构的过程模型的转化范围.

在本文所提出的行为等价过程树生成算法的基础上,下一步将继续针对过程树进行深入探讨.其次随着模型规模的增加,过程树生成算法的效率需要继续进行优化.另外,将过程树应用在模型结构精简、避免模型复杂性、过程模型的检索与存储等工作中具有重要价值,因此这也是下一步需要考虑的.

致谢:

本文核心工作为朱锐博士在北京大学高可信软件技术教育部重点实验室访学期间,与北京大学金芝教授讨论完成.特向北京大学高可信软件技术教育部重点实验室的各位老师和同学表示感谢!

References:

[1] Jin Tao, Wen Lijie. Indexing technology for business process models. Computer Integrated Manufacturing Systems, 2011, 17(8): 1580-1586 (in Chinese with English abstract). [doi:10.4028/www.scientific.net/AMR.154-155.87]

[2] B. F. Van Dongen. BPI Challenge 2014: Change details, Rabobank Nederland, 2014. [doi: 10.4121/uuid:d5ccb355-ca67-480f-8739-289b9b593aaf]

[3] Wil Van Der Aalst. Process Mining: Data Science in Action. 2th ed. Heidelberg New York Dordrecht London: Springer, 2016. [doi: 10.1007/978-3-662-49851-4]

[4] Niek Tax, Natalia Sidorova, Reinder Haakma, Wil M.P.van der Aalst. Mining local process models. Journal of Innovation in Digital Ecosystems, 2016, 3(2): 183-196. [doi:10.1016/j.jides.2016.11.001]

[5] Claudia Diamantini, Laura Genga, Domenico Potena. Behavioral process mining for unstructured processes. Journal of Intelligent Information Systems, 2016, 47(1): 5-32. [doi:10.1007/s10844-016-0394-7]

[6] Xuewei Zhang, Wei Song, Jiacun Wang, Jianchun Xing, Qizhen Zhou. Measuring Business Process Consistency across Different Abstraction Levels. IEEE Transactions on Network and Service Management, 2019, 16(1): 294-307. [doi:10.1109/TNSM.2018.2883362]

[7] Diana Chinces, Ioan Salomie. Optimizing spaghetti process models. In: Proc. of the 2015 20th International Conference on Control Systems and Computer Science. Bucharest, Romania: IEEE, 2015. 506-511. [doi:10.1109/CSCS.2015.15]

[8] Niek Tax, Natalia Sidorova, Reinder Haakma, Wil M. P van der Aalst. Event Abstraction for Process Mining Using Supervised Learning Techniques. In: Bi Y, Kapoor S, Bhatia R, ed. Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016. Cham: Springer International Publishing, 2016. 251-269. [doi:10.1007/978-3-319-56994-9_18]

[9] Ornela Çela, Agnès Front, Dominique Rieu. Model Consolidation: A Process Modelling Method Combining Process Mining and Business Process Modelling. In: Gulden J, ed. BPMDS 2018, EMMSAD 2018: Enterprise, Business-Process and Information Systems Modeling. Cham: Springer International Publishing, 2018. 117-130. [doi:10.1007/978-3-319-91704-7_8]

[10] Maikel Leemans, Wil MP Van Der Aalst, Mark GJ Van Den Brand. Recursion aware modeling and discovery for hierarchical software event log analysis. In: Proc. of the 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER). Campobasso, Italy: IEEE, 2018. 185-196. [doi:10.1109/SANER.2018.8330208]

[11] Xixi Lu, Seyed Amin Tabatabaei, Mark Hoogendoorn, Hajo A. Reijers. Trace Clustering on Very Large Event Data in Healthcare Using Frequent Sequence Patterns. In: Hildebrandt T., van Dongen B., Röglinger M., Mendling J. BPM 2019: Business Process Management. Cham: Springer International Publishing, 2019. 198-215. [doi: 10.1007/978-3-030-26619-6_14]

[12] ZHU Rui, LI Tong, MO Qi, HE Zhen-Li, YU Qian, WANG Yi-Quan. Data-Driven Bilayer Software Process Mining. Journal of Software, 2018, 29(11): 221-249 (in Chinese with English abstract). [doi: 10.13328/j.cnki.jos.005304]

[13] Wil M. P. Van Der Aalst. Process discovery from event data: Relating models and logs through abstractions. Wiley Interdisciplinary Reviews Data Mining & Knowledge Discovery, 2018, (3): e1244. [doi: 10.1002/widm.1244]

[14] Wil MP van der Aalst. Everything You Always Wanted to Know About Petri Nets, but Were Afraid to Ask. In: Hildebrandt T., van Dongen B., Röglinger M., Mendling J. BPM 2019: Business Process Management. Cham: Springer International Publishing, 2019. 3-9. [doi: 10.1007/978-3-030-26619-6_1]

[15] RobinMilner, Lin Huimin. Communicating and mobile systems: π-calculus. 1th ed, Beijing: TSINGHUA UNIVERSITY PRESS, 2009 (in Chinese with English abstract). [ISBN: 978-7-302-20725-2]

[16] Wil van der Aalst, Joos Buijs, Boudewijn van Dongen. Towards Improving the Representational Bias of Process Mining. In: Aberer K., Damiani E., Dillon T. Data-Driven Process Discovery and Analysis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. 39-54. [doi: 10.1007/978-3-642-34044-4_3]

[17] J. Buijs, B. Van Dongen, W. Van Der Aalst. A genetic algorithm for discovering process trees. In: Proc. of 2012 IEEE Congress on Evolutionary Computation. Brisbane, QLD, Australia: IEEE, 2012. 1-8.

[18] Zhu Rui, Zhang Zhixing, Mo Qi, Li Tong, Ma Zifei, Li Bin. Hybrid process mining method supporting complex structures. Computer Integrated Manufacturing Systems, 2018, 24(7): 1653-1670 (in Chinese with English abstract). [doi: 10.13196/j.cims.2018.07.007]

[19] Mohd Anuaruddin Bin Ahmadon, Shingo Yamaguchi. Convertibility and Conversion Algorithm of Well-Structured Workflow Net to Process Tree. In: Proceedings of the 2013 First International Symposium on Computing and Networking. Matsuyama, Japan: IEEE, 2013. 122-127. [doi: 10.1109/CANDAR.2013.24]

[20] Chunfu Zhong, Wenlong He, Zhiwu Li, Naiqi Wu, Ting Qu. Deadlock analysis and control using Petri net decomposition techniques. Information Sciences: An International Journal, 2019, 482: 440-456. [doi: 10.1016/j.ins.2019.01.029]

[21] Cong Liu, Qingtian Zeng, Hua Duan, Lei Wang, Jie Tan, Chongguang Ren, Wangyang Yu. Petri Net Based Data-Flow Error Detection and Correction Strategy for Business Processes. IEEE Access, 2020, 8: 43265-43276. [doi: 10.1109/ACCESS.2020.2976124]

[22] Yuting Li, Li Yin, Yufeng Chen, Zhenhua Yu, Naiqi Wu. Optimal Petri net supervisor synthesis for forbidden state problems using marking mask. Information Sciences, 2019, 505:183-197. [doi: 10.1016/j.ins.2019.07.008]

[23] Sander J. J. Leemans, Dirk Fahland, Wil M. P. Van Der Aalst. Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In: Colom JM., Desel J. Proc. of PETRI NETS 2013: Application and Theory of Petri Nets and Concurrency. Berlin, Heidelberg: Springer, 2013. 311-329. [doi: 10.1007/978-3-642-38697-8_17]

[24] Sander J. J. Leemans, Dirk Fahland, Wil M. P. Van Der Aalst. Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In: Lohmann N., Song M., Wohed P. Proc. of BPM 2013: Business Process Management Workshops. Cham: Springer International Publishing, 2013. 66-78. [doi: 10.1007/978-3-319-06257-0_6]

[25] Javier Esparza, Stefan Römer, Walter Vogler. An Improvement of McMillan's Unfolding Algorithm. Lncs, 1996, 1099(3): 285-310. [doi: 10.1023/A:1014746130920]

[26] Andrea Burattin, Alessandro Sperduti. PLG: A Framework for the Generation of Business Process Models and Their Execution Logs. In: zur Muehlen M., Su J. Proc. of BPM 2010: Business Process Management Workshops. Berlin, Heidelberg: Springer, 2011. 214-219. [doi: 10.1007/978-3-642-20511-8_20]

[27] Artem Polyvyanyy, Luciano García-Bañuelos, Dirk Fahland, Mathias Weske. Maximal Structuring of Acyclic Process Models. 2014, 57(1): 12-37. [doi: 10.1093/comjnl/bxs126]

[28] Tan Wenan, Xie Na, Zhao Lu, Sun Yong, Huang Li. Retrieval of Business Process Models Based on Performance Constraints. Computer Integrated Manufacturing Systems, 2019, 25(04): 847-855 (in Chinese with English abstract). [doi: CNKI:SUN: JSJJ.0.2019-04-006]

[29] Sun Jinyong, GuTianlong, Wen Lijie, Qian Junyan. Similarity algorithmfor semantic workflows used in process-oriented case-based reasoning. Computer Integrated Manufacturing Systems, 2016, 22(2): 381-394 (in Chinese with English abstract). [doi: 10.13196/j.cims.2016.02.011]

[30] Sun Jinyong, GuTianlong, Wen Lijie, Qian Junyan, Meng Yu. Retrieval of Similar Semantic Workflows Based on Behavioral and Structural Characteristics.. Journal of Computer Research and Development, 2017, 54(9): 1880-1891 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2017.20160755]

[31] DONG Zi-He, WEN Li-Jie, HUANG Hao-Wei, WANG Jian-Min. Behavioral Similarity Algorithm for Process Models Based on Firing Sequence Collection. Journal of Software, 2015, 26(3): 449-459 (in Chinese with English abstract). [doi: 10.13328/ j.cnki.jos.004765]

[32] Jin Tao, Jianmin Wang, Lijie Wen. Querying Business Process Models Based on Semantics. In: Yu J.X., Kim M.H., Unland R. Proc. of DASFAA 2011: Database Systems for Advanced Applications. Berlin, Heidelberg: Springer, 2011. 164-178. [doi: 10.1007/978-3-642-20152-3_13]

[33] Artem PolyvyanyyMarcello La RosaArthur H. M. ter Hofstede. Indexing and Efficient Instance-Based Retrieval of Process Models Using Untanglings. In: Jarke M, ed. Proc. of CAiSE 2014: Advanced Information Systems Engineering. Cham: Springer International Publishing, 2014. 439-456. [doi: 10.1007/978-3-319-07881-6_30]

[34] Alessio Bottrighi, Luca Canensi, Giorgio Leonardi, Stefania Montani, Paolo Terenziani. Interactive mining and retrieval from process traces. Expert Systems with Applications, 2018, 110: 62-79. [doi: 10.1016/j.eswa.2018.05.041]

[35] Wei Song, Hans-arno Jacobsen, S.C. Cheung, Hongyu Liu. Workflow Refactoring for Maximizing Concurrency and Block-Structuredness. IEEE Transactions on Services Computing, 2018, [doi: 10.1109/TSC.2018.2867593]

附中文参考文献:

[1] 金涛,闻立杰.业务过程模型库索引技术.计算机集成制造系统,2011,17(8):1580-1586.[doi:10.4028/www.scientific.net/AMR.154-155.87]

[12] 朱锐,李彤,莫启,何臻力,于倩,王一荃.数据驱动的双层次软件过程挖掘方法.软件学报,2018,29(11):221-249.[doi:10.13328/ j.cnki.jos.005304]

[15] RobinMilner,林惠民.通信与移动系统:π演算.1thed,北京:清华大学出版社,2009.[ISBN:978-7-302-20725-2]

[18] 朱锐,张志幸,莫启,李彤,马自飞,黎彬.支持复杂结构的混成过程挖掘方法.计算机集成制造系统,2018,24(7):1653-1670. [doi:10.13196/j.cims.2018.07.007]

[28] 谭文安,谢娜,赵璐,孙勇,黄黎.基于性能约束的业务过程模型检索.计算机集成制造系统,2019,25(04):847-855. [doi:CNKI:SUN:JSJJ.0.2019-04-006]

[29] 孙晋永,古天龙,闻立杰,钱俊彦.用于面向过程的基于实例推理的语义工作流相似性算法.计算机集成制造系统,2016,22(2): 381-394. [doi:10.13196/j.cims.2016.02.011]

[30] 孙晋永,古天龙,闻立杰,钱俊彦,孟瑜.基于行为和结构特征的相似语义工作流检索.计算机研究与发展,2017,54(9):1880-1891. [doi:10.7544/issn1000-1239.2017.20160755]

[31] 董子禾,闻立杰,黄浩未,王建民.基于触发序列集合的过程模型行为相似性算法.软件学报,2015,26(3):449-459. [doi:10.13328/j.cnki.jos.004765]