软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(3):509−532 [doi: 10.13328/j.cnki.jos.004771]

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

一种基于 Token Log 的符合性检查方法*

李传艺[1,2,3], 葛季栋[1,2,3], 胡海洋[5], 胡 昊[1,4], 骆 斌[1,2,4]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210046)

[2](南京大学 软件学院,江苏 南京 210093)

[3](高维信息智能感知与系统教育部重点实验室(南京理工大学),江苏 南京 210094)

[4](南京大学 计算机科学与技术系,江苏 南京 210046)

[5](杭州电子科技大学 计算机学院,浙江 杭州 310018)

通讯作者: 葛季栋, E-mail: gjd@nju.edu.cn

摘 要: 使用事件日志进行符合性检查的主要方法是:使用过程模型模拟执行事件日志中的任务序列,通过统计可被模型再现的任务序列及模型运行中可能触发的非运行序列中的任务个数,判断模型与日志的符合程度.但这种判断方法并不完备:如果模型中包含大量选择结构,则即使日志是模型本身的日志,也会因为模拟执行较多任务时会触发当前序列外的其他任务,而误判日志与模型的符合性较低;或者,如果模型中只包含少数的并发结构和多数的顺序结构,则即使日志只包含顺序结构的内容且非该模型对应日志时,也会因为在模拟执行时只有个别任务会导致模型无法继续执行,而其他多数任务可以执行而误判日志与模型有较高的符合性.基于已有方法的弱点,提出了使用日志内容检查模型结构正确性与使用模型结构检查日志内容完整性的双向检查标准,并提出一种内容特征与模型结构特征一一对应的新型日志——Token Log,用于过程模型与系统日志的符合性检查,使得检查和判断过程更加清晰简洁,结果更加准确.

关键词: 符合性检查;过程挖掘;Petri 网;工作流网;Token Log;*T*-不变量;*S*-不变量;ProM

中图法分类号: TP311

中文引用格式: 李传艺,葛季栋,胡海洋,胡昊,骆斌.一种基于 Token Log 的符合性检查方法.软件学报,2015,26(3):509−532. http://www.jos.org.cn/1000-9825/4771.htm

英文引用格式: Li CY, Ge JD, Hu HY, Hu H, Luo B. Method for conformance checking based on Token Log. Ruan Jian Xue Bao/Journal of Software, 2015,26(3):509−532 (in Chinese). http://www.jos.org.cn/1000-9825/4771.htm

Method for Conformance Checking Based on Token Log

LI Chuan-Yi[1,2,3], GE Ji-Dong[1,2,3], HU Hai-Yang[5], HU Hao[1,4], LUO Bin[1,2,4]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210046, China)

[2](School of Software, Nanjing University, Nanjing 210093, China)

[3](Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information, Ministry of Education (Nanjing University of Science and Technology), Nanjing 210094, China)

[4](Department of Computer Science & Technology, Nanjing University, Nanjing 210046, China)

[5](School of Computer, Hangzhou Dianzi University, Hangzhou 310018, China)

* 基金项目: 国家自然科学基金(61100039, 61321491, 91318301, 61272188, 61202002); 国家高技术研究发展计划(863) (2013AA01A213); 国家重点基础研究发展计划(973)(2015CB352202); 中央高校基本科研业务费; 江苏省自然科学基金(BK20131277); 浙江省自然科学基金(LY12F02005); 高维信息智能感知与系统教育部重点实验室(南京理工大学)基金(30920130122005); 南京大学计算机软件新技术国家重点实验室开放基金(KFKT2014B15); 浙江省哲学社会科学重点研究基地(信息化与经济社会发展研究中心)课题(14JDXX04YB)

收稿时间: 2014-07-01; 修改时间: 2014-09-30; 定稿时间: 2014-11-21

Abstract: Logs used in conformance checking with process models are often the event logs. Conformity between the model and the log is often measured by counting the traces which could be reconstructed and the tasks which would be evoked but were not in the running trace through rerunning the model according to the task traces in the log. However the method is not sufficiently comprehensive. While checking the model consisting of many selections with its Event Log, the conformity will be very low due to the large number of evoked tasks that are not in the running task trace. Moreover, while checking the model mainly composed by parallel branches with the log only containing sequential task traces and sharing the same task set with the model, the conformity will be very high due to the fact that only a few tasks can't be executed normally while monitoring the real behavior. To overcome the weakness of the original method, a bidirectional checking method made up of checking the accuracy of the model and checking the completeness of the log, and a new kind of log named Token Log which can describe the property of its corresponding model, are proposed in this paper. With the Token Log, the new method for conformance checking is clearer, more concise and more accurate.

Key words: conformance checking; process mining; Petri-net; Token Log; *T*-invariant; *S*-invariant; ProM

符合性检查(conformance checking)[1,2]是过程挖掘(process mining)[2−4]技术研究的重要内容.在符合性检查过程中,将已知的过程模型与已有的系统日志进行对比,用来检查系统日志记载的实际行为与设计的过程行为是否相符.符合性检查是过程模型仓库(process model repository)[5]的重要功能之一,特别是过程模型查询[6]相关的功能.符合性检查方法的输入为一个过程模型和一个系统日志,输出是该模型与日志符合程度的度量结果.现有算法中,过程模型均使用 Petri 网(Petri-nets)[7]表示,系统日志均为事件日志(Event Log)[8].如果需要检验的系统模型没有使用 Petri 网表示,则需要转化为 Petri 网形式;同样,系统日志也需要首先转化为事件日志.例如,在 Web Service 领域进行符合性检查时,首先将其系统架构(abstract BPEL)转化为 Petri 网,将其 SOAP Message 转化为 Event Log,再使用符合性检查方法[9].

使用事件日志进行符合性检查时存在误判的情况,例如,当模型中包含大量选择结构时(非 Overfitting[10]结构),即使日志是模型本身的日志,也会因为在模拟执行较多任务时会触发当前执行序列外的其他任务,而误判日志与模型的符合性较低.关于使用事件日志进行符合性检查可能存在的问题的描述见第 2.3 节和第 5 节.

基于事件日志可能带来的问题,本文提出了一种基于 Token Log 的符合性检查的方法.Token Log 是一种由工作流相关系统产生的、用以记录系统运行情况的新型日志.不同于事件日志,Token Log 关注系统中各种资源的生产、流动与消耗情况,并以此记录系统任务间的关系(详见第 2.1 节).在符合性检查研究中,使用 Token Log 替代事件日志的优点有(详见第 5 节):

(1) 全面检查模型结构正确性与日志行为的完整性,使得检查思路更清晰合理;
(2) 化动态模拟执行为静态结构分析,简化检查过程;
(3) 简化根据计算结果判定是否符合的过程;
(4) 可根据输入模型与日志的特点定制度量参数与判定标准.

为了更加详细、清晰地介绍使用 Token Log 参与过程模型与系统日志符合性检查研究的方法和优点,本文第 1 节介绍过程挖掘、Petri 网、工作流网、事件日志和符合性检查等相关背景知识.第 2 节说明 Token Log 的来源与定义,并详细分析 Token 与结构间的映射关系.第 3 节详细介绍模型检查与日志检查的方法.第 4 节描述判定模型与日志符合程度的过程.第 5 节详细分析使用 Token Log 的优点.第 6 节分析使用 Token Log 对模型增强的作用.相关实验工具、算法和实现过程在第 7 节介绍.第 8 节总结全文的工作并提出对未来工作的设想.

1 背景知识

1.1 过程挖掘与符合性检查

过程挖掘技术指通过分析信息系统的日志信息,发现、模拟和增强业务过程的技术,其主要包括过程发现(discovery)、符合性检查(conformance checking)和模型增强(enhancement)这 3 个研究方向[11]:

- 过程发现指通过分析信息系统的日志信息再现原有过程模型;
- 符合性检查指通过某种计算方法,比较过程模型结构与系统日志行为的匹配程度;

- 模型增强指通过分析系统日志发现过程模型实际运行中出现的错误、意外和瓶颈等情况,并针对这些情况对原过程模型作出相应增强修改.

过程挖掘技术还包括其他一些基础的研究方向,如过程模型的存储[12]、过程模型的形式化描述[5]和过程模型相似度检验[13]等.

符合性检查又称合规性检验,是判断过程模型与系统日志符合程度的过程挖掘技术.文献[1]提出了过程模型与系统日志的匹配程度需要通过 *Fitness* 和 *Appropriateness* 两种指标衡量,其含义如下:

(1) *Fitness* 用以衡量日志记录的系统行为可被过程模型再现的程度;

(2) *Appropriateness* 用以衡量过程模型再现日志行为时的准确程度与模型本身的简洁程度.

Fitness 与 *Appropriateness* 通过分析过程模型模拟执行日志时的状态变化算得:

- 计算 *Fitness* 时,需要统计每个任务执行时,消耗令牌、新产生令牌、未使用令牌和缺失令牌的数目;
- 计算 *Appropriateness* 时,需要统计每个任务执行时,所有可能被触发的任务数目.

符合性检查过程中对两种指标的分析与统计,不仅是为了判断过程模型与日志的匹配程度,同时也为增强过程模型提供了潜在的途径.

1.2 建模工具Petri网与工作流网

Petri 网是一种常用于描述并发过程模型的建模工具,例如,文献[14−16]均使用 Petri 网为过程模型建模.图 1 是使用 Petri 网表示的医院门诊管理系统的简化过程模型.本文使用该例子是为了说明本文描述的检查方法适用于包含各种特殊结构的模型.

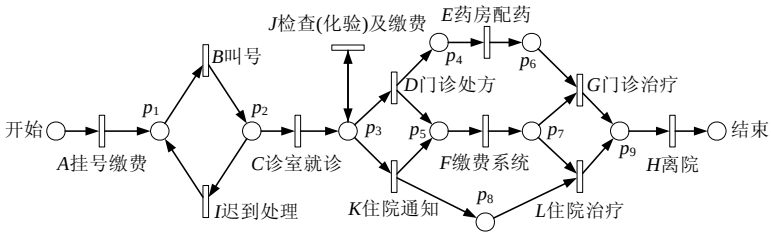


Fig.1 Simplified process model of hospital for outpatient

图 1 医院门诊流程简化模型

Petri 网是由变迁(transition)、库所(place)及其之间弧线(arc)组成的动态模型系统.本文 Petri 网使用表达式 $PN=(P,T,F)$表示,P 是库所集合,T 是变迁集合,F 是弧线集合.更多关于 Petri 网的介绍见文献[17].图 1 的 Petri 网中存在一些特殊的结构:

- 变迁 *B* 和变迁 *I* 组成了一个循环结构,而变迁 *J* 自身形成一个循环结构;
- 变迁 *F* 是一个非自由的选择结构,即:如果 *F* 由变迁 *D* 触发,则变迁 *G* 被执行;如果 *F* 由 *K* 触发,则变迁 *L* 被选择执行.

图 1 同时是一个合理的(sound)工作流网(workflow net,简称 WF-net)[18].工作流网是给工作流过程模型建模的 Petri 网子类.

一个 Petri 网可以称为一个工作流网,必须满足:

(1) 有唯一的开始库所;

(2) 有唯一的结束库所;

(3) 每一个节点都必须在某一个从开始库所到结束库所的路径上.

而一个工作流网被称为一个合理的工作流网,必须满足:

(1) 所有任务都可以潜在被执行(所有任务都在某一个可以执行的路径上),即,活性(live);

(2) 每次从开始库所填入一个令牌开始执行,以结束库所中遗留一个令牌结束执行,即,有界性(bounded).

工作流网的合理性要求过程模型执行时既没有死锁(dead lock)也没有活锁(live lock).本文使用的所有过程模型均为合理的工作流网.

1.3 事件日志

系统日志是过程挖掘研究的起点,目前,所有被使用的系统日志均为事件日志(event log).事件日志由事件组成,并满足 3 个假设条件:

(1) 每个事件(event)关联一个任务(task);

(2) 每个事件关联一个运行实例(case)[8]的 ID;

(3) 所有事件按照执行的先后顺序排列.

在过程挖掘相关研究中,首先从事件日志中获取每个运行实例的任务串(task trace).例如,表 1 是图 1 某事件日志的实例任务串(最后一列为实例出现的次数,即 case numbers).

Table 1 An example of event log of the process model in Fig.1

表 1 图 1 所示过程模型的事件日志实例列表

Case ID	Task trace	Case numbers
1	*A B C D E F G H*	N_1
2	*A B C D F E G H*	N_2
3	*A B C K F L H*	N_3
4	*A B C J J K F L H*	N_4
5	*A B I B C D E F G H*	N_5
6	*A B I B C D F E G H*	N_6
7	*A B I B I B C J K F L H*	N_7

表 1 共有 7 个运行实例,每个实例出现的频率不同,且列出的运行实例并不包含过程模型执行时所有可能的任务序列.最后一列(case numbers)被用来计算实例占整个日志的权重,每个实例按此权值参与计算日志整体与过程模型的符合性.

2 新型日志 Token Log

本节主要介绍本文所使用的 Token Log 的相关内容.首先阐述 Token 的含义,介绍 Token Log 的内容,并说明如何产生 Token Log;然后阐述使用 Token Log 替换事件日志的核心动机;最后详细说明 Token Log 的特性.

2.1 Token的含义及内容

文献[19]描述了一种 Artifact-Centric 工作流过程模型,它以某种实际存在的(concrete)、编号的(identifiable)和自描述的(self-describing)信息块为中心,如业务实体(business entity)和业务记录(business record),这些一直存在于模型中的信息块被称为 Artifact.文献[20]中有关于 Artifac-Centric 工作流过程模型的实例与应用描述.在 Artifact-Centric 过程模型中,根据 Artifact 状态与内容的改变,可推理出各个任务间的执行关系.但多数工作流中并不存在完整生命周期定义的 Artifact 实体,难以通过该实体的状态与内容的改变推理出任务间关系.一般工作流过程模型中,每两个存在依赖关系的任务间会存在资源和执行权限等信息的流动,这些信息就是任务执行的前置与后置条件.将这些信息看作更细粒度的类似 Artifact 的实体,通过记录这些实体便可获得任务间的依赖关系.在 Petri 网中,令牌被用来代表任务的前置与后置条件,借鉴此概念,本文使用令牌(token)代表工作流过程中两个任务间流动的信息,并通过记录这些信息的生产者与使用者获取任务间的关系.Token 的详细描述如下:

oken 代表任务间流动的信息,不同 Token 代表不同种类的信息.Token 由一个任务生产并传递给另一个任务.产生 Token 的任务称为生产任务(produce task,简称 PT),接收 Token 的任务称为消费任务(consume task,简称 CT).每一个 Token 都记录了自己的生产任务与消费任务.

一个任务执行后可能产生多个 Token;同样,一个任务执行时可能需要多个输入 Token.为便于区分这些同时被消费与同时被生产的 Token,为每个 Token 标记生产与消费时间.Token 的生产时间与消费时间即为生产任务与消费任务的执行时间,使用执行 ID(execute ID,简称 EID)标记,定义如下:

定义 1(执行 ID). 执行 ID(execute ID,简称 EID)是每个运行实例中,每个任务执行时分配的唯一执行标记.

不同实例中的任务执行可能会有相同的 EID.Token Log 中也包含过程执行的不同实例(case),使用 Case ID (CID)标记.每一个任务的每一次执行都可以通过组合(CID,EID)唯一确定.表 2 是图 1 过程模型最终可以参与计算的 Token Log 示例(添加 EID),有两个执行实例,对应于表 1 中的 Case 4,Case 5 和 Case 6.在事件日志中两个用于表示并发结构的不同实例(Case 5 和 Case 6)在 Token Log 中被相同的 Token 内容表示.表 1 中的 Case 1 与 Case 2 在 Token Log 中也是这样.这说明,使用 Token Log 替代事件日志将减少参与计算的实例数目.

Table 2 Example of Token Logs for Case 4, Case 5 and Case 6 shown in Table 1

表 2 表 1 中图 1 结构的 Case 4,Case 5 和 Case 6 的对应 Token Log 示例

CID	PT	CT	PEID	CEID	Case number	CID	PT	CT	PEID	CEID	Case number
4	*A*	*B*	1	2	N_4	5(6)	*A*	*B*	1	2	N_5+N_6
	B	*C*	2	3			*B*	*I*	2	3	
	C	*J*	3	4			*I*	*B*	3	4	
	J	*J*	4	5			*B*	*C*	4	5	
	J	*K*	5	6			*C*	*D*	5	6	
	K	*F*	6	7			*D*	*E*	6	7	
	K	*L*	6	8			*D*	*F*	6	8	
	F	*L*	7	8			*E*	*G*	7	9	
	L	*H*	8	9			*F*	*G*	8	9	
							G	*H*	9	10	

2.2 Token Log的产生

根据上述 Token 含义,不仅在 YAWL(yet another workflow language)[21]工作流引擎上实现打印 Token Log,还在 Petri 网建模工具 PIPE 上开发了模拟执行工作流模型,并产生 Token Log 的插件.

在 YAWL 引擎中,YIdentifier 作为标记业务过程状态的实体在工作流网中流动,作用类似于 Petri 网中的令牌[22];YCondition 作为存放 YIdentifier 的容器,作用类似于 Petri 网中的库所[22].任务(YTask)执行前需要判断其输入 YCondition 中是否包含足够的 YIdentifier;任务执行时需要删除对应输入 YCondition 中对应的 YIdentifier;任务执行完毕需要向输出 YCondition 中填入对应的 YIdentifier.实验中,首先在源代码中修改 YIdentifier 定义,添加生产任务与消费任务属性.任务执行时,设定输入 YIdentifier 的消费任务,并打印 YIdentifier 信息;任务执行后,设定输出 YIdentifier 的生产任务.但由打印结果发现:YIdentifier 在工作流网中为引用传递,YCondition 中只持有 YIdentifier 的引用.任务改变某个 YIdentifier 属性时,会对所有持有该 YIdentifier 引用的 YCondition 产生影响.

最终,实验选择在 YAWL 中添加新 Token 对象:

- 任务执行完时新建 Token 并设定生产任务,然后传递给输出 YCondition;
- 任务执行开始时设定输入 Token 的消费任务并打印 Token 信息,随即回收该 Token.

实验中,使用 YAWL 引擎生成 Token Log 的核心时序图如图 2 所示(YEngine 是管理所有运行过程实例的对象,YNetRunner 是独立执行一个过程实例的对象).

其中,方法 *consume*(·),*print*(·)和 *produce*(·)为添加的核心方法:

- *consume*(·)主要工作为获取任务输入库所中的 Token 并设定当前任务为消费任务;
- *print*(·)主要工作为打印所有消费的 Token 信息生成 Token Log;
- *produce*(·)主要工作为新建 Token 并输入到输出库所中.

本文描述的研究工作中,同时完成了在 Petri 网建模工具 PIPE 中开发模拟执行工作流过程模型并产生 Token Log 的插件,并在本文实验过程中使用了该插件生成的 Token Log.该插件的执行原理与 YAWL 中生成 Token Log 的原理相同.

关于该插件的详细介绍与使用说明见第 7.1 节.

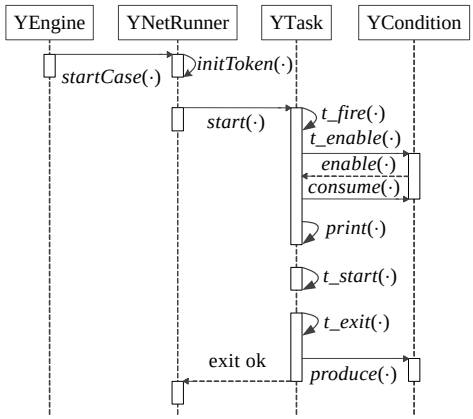


Fig.2 Process of making Token Log in YAWL

图 2 YAWL 引擎中打印 Token Log 原理图

2.3 使用Token Log的动机

除误判情况外,在符合性检查中使用事件日志,更难以避免过程模型 Overfitting 与 Underfitting 时对检查结果的干扰[10].过程模型 Overfitting 指过程模型可以生成参与检查的事件日志之外的很多其他日志内容,过程模型 Underfitting 指过程模型通过包含重复任务恰好能够生成参与比较的事件日志的内容[4].图 3 展示了与表 1 中 Event Log 对应的 Overfitting 与部分 Underfitting 过程模型.

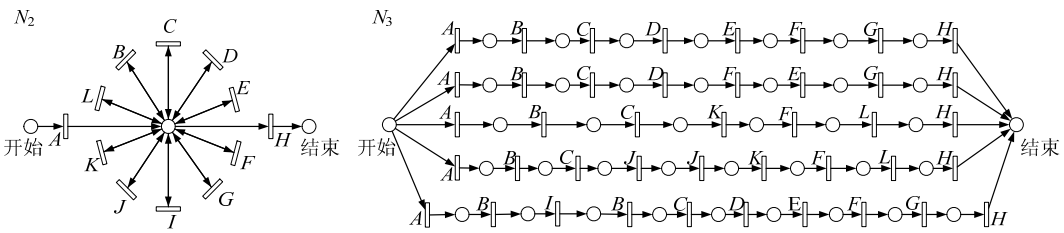


Fig.3 Overfitting and Underfitting processes of the log shown in Table 1[1]

图 3 表 1 事件日志的 Overfitting 与 Underfitting 模型[1]

使用事件日志进行符合性检查会受到 Overfitting 与 Underfitting 结构影响的原因是:没有直接检查日志对模型所有可能产生的执行路径的覆盖程度,而是通过统计可能触发多少非可执行路径进行间接检查.然而,本文提出了直接统计日志内容是否覆盖过程模型所有可能执行路径(即,日志完整性)的方法.但如果直接描述过程模型允许的所有可能执行路径,则会出现状态空间爆炸.因此,本文通过统计模型中特殊结构结点允许的所有行为是否被日志内容覆盖,以判断整个模型是否被覆盖.由于使用事件日志难以描述过程模型中的特殊结构,所以本文选择使用可以唯一确定过程模型特殊结构结点的 Token Log.使用 Token Log 进行符合性检查的主要工作包括:

(1) 基于日志内容判断过程模型结构的正确性(详见第 3.1 节);

(2) 基于过程模型结构判断 Token Log 内容的完整性(详见第 3.2 节).

在使用 Token Log 时,不再需要动态模拟执行过程模型,而只要根据 Token 间匹配关系与模型特征结构的对应关系(详见第 2.3 节),静态分析并计算日志与模型的符合程度,即使用 Token Log 将简化检查过程.新的检验方法将通过唯一的计算结果判断匹配程度,且可以根据日志与模型的实际情况自定义相关计算参量及判断标准.使用 Token Log 的优点详见第 5 节.

2.4 Token与结构的映射关系

使用 Token Log 优化符合性检查的基础是,一组特定关系的 Token 与过程模型特征结构之间的映射关系.本节首先介绍过程模型特征结构与 Token 间匹配关系,然后给出 Token 匹配关系与特征结构间的映射关系.

2.4.1 过程模型的特征结构

任何复杂的工作流过程模型均有几种简单的局部结构[7]组成.这些简单的局部结构包括:

(1) 顺序结构,如图 4(a)和图 4(b)所示;

(2) 并发(and-split)和同步(and-join)结构:任务有多个输出和任务有多个输入,如图 4(d)和图 4(e)所示;

(3) 选择(or-split)和或同步(or-join):库所有多个输出和库所有多个输入,如图 4(f)和图 4(g)所示;

(4) 单步循环(one loop)[8]:一个任务即是一个库所的输入也是该库所的输出,如图 4(h)所示;

(5) 特殊的隐式库所(Implicit place)[8]:两个任务由多个库所直接连接,如图 4(c)所示.

这些简单的局部结构在本文提出的符合性检查中作为过程模型的特征结构,用于判断 Token Log 是否覆盖这些特征结构允许的所有行为.

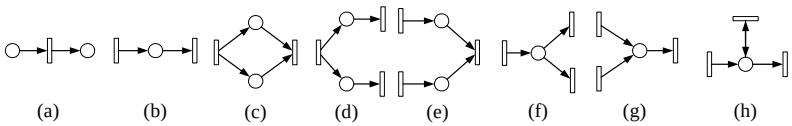


Fig.4 Substructures in process models

图 4 过程模型中的局部结构

2.4.2 Token 间匹配关系

在第 2.1 节中提到,最终参与计算的 Token Log 中所有内容相同的运行实例都会被归并,并统计其出现次数,所以 Token 中的 Case ID 信息指代了一类运行实例.Token 的完整定义如下:

定义 2(Token). Token 是 Token Log 中的条目,使用 5 元组 $token=(CID,PT,CT,PEID,CEID)$表示,其中,$CID$ 表示 Token 所在实例的 Case ID,PT 和 CT 分别表示 Token 的生产任务和消费任务,$PEID$ 和 $CEID$ 分别表示生产任务与消费任务在生产与消费 Token 时的 EID.

本文 Token 之间的关系被称为匹配(match)关系.由 Token 定义可知,Token $t_1=(CID_1,PT_1,CT_1,PEID_1,CEID_1)$由 CT_1 在 $CEID_1$ 执行中被消耗,而 CT_1 在此执行之后必然生产另一个 Token $t_2=(CID_1,PT_2,CT_2,PEID_2,CEID_2)$,其中,$PT_2=CT_1$ 且 $PEID_2=CEID_1$.此时,t_1 和 t_2 被称为顺序匹配(sequential match).但是,有一类特殊的 Token,它们由同一个任务生产和消费,但是在两次不同的执行中.这一类 Token 将不与其他 Token 进行关系匹配,即:在发现匹配关系时,发现这些特殊 Token 的优先级更高.这一类 Token 称为单步循环标记(one loop marking):

定义 3(单步循环标记). Token $t=(CID,PT,CT,PEID,CEID)$是一个单步循环标记当且仅当 $PT=CT$.所有的单步循环标记 Token 组成的集合为 $OLM=\{(CID,PT,CT,PEID,CEID)|PT=CT\}$.

类似顺序匹配关系,Token 间还有其他各种匹配关系.在定义匹配关系时,首先排除单步循环标记的 Token.所有匹配关系的定义如下:

定义 4(顺序匹配). Token $t_1=(CID_1,PT_1,CT_1,PEID_1,CEID_1)$与 $t_2=(CID_1,PT_2,CT_2,PEID_2,\ CEID_2)$称为顺序匹配关系当且仅当:$t_1\notin OLM,t_2\notin OLM,CID_1=CID_2,CT_1=PT_2,CEID_1=PEID_2$.

定义 5(并发匹配,and-split match). Token $t_1=(CID_1,PT_1,CT_1,PEID_1,CEID_1)$与 $t_2=(CID_1,PT_2,CT_2,PEID_2,CEID_2)$称为并发匹配关系当且仅当:$t_1\notin OLM,t_2\notin OLM,CID_1=CID_2,PT_1=PT_2,PEID_1=PEID_2$.

定义 6(同步匹配,and-join match). Token $t_1=(CID_1,PT_1,CT_1,PEID_1,CEID_1)$与 $t_2=(CID_1,PT_2,CT_2,PEID_2,CEID_2)$称为同步匹配关系当且仅当:$t_1\notin OLM,t_2\notin OLM,CID_1=CID_2,CT_1=CT_2,CEID_1=CEID_2$.

定义 7(多路匹配,multi-path match). Token $t_1=(CID_1,PT_1,CT_1,PEID_1,CEID_1)$与 $t_2=(CID_1,PT_2,CT_2,PEID_2,CEID_2)$称为多路匹配关系当且仅当:

$t_1 \notin OLM, t_2 \notin OLM, CID_1=CID_2, PT_1=PT_2, PEID_1=PEID_2, CT_1=CT_2, CEID_1=CEID_2$.

如果两个 Token 只是在同一个运行实例中而没有任务执行 ID 相同,即相同的执行任务必定在同一个运行实例中执行了多次,这个结构中存在循环结构,因此,Token 间的这种关系称为循环匹配,具体定义如下:

定义 8(循环匹配,loop match). Token $t_1=(CID_1,PT_1,CT_1,PEID_1,CEID_1)$与 $t_2=(CID_1,PT_2,CT_2,PEID_2,CEID_2)$称为循环匹配当且仅当:$CID_1=CID_2, PEID_1 \neq PEID_2, CEID_1 \neq CEID_2$,且满足以下任意条件:

(1) $PT_1=PT_2, CT_1=CT_2$;

(2) $PT_1 \neq PT_2, CT_1=CT_2$;

(3) $PT_1=PT_2, CT_1 \neq CT_2$.

2.4.3 Token 匹配关系与特征结构间的映射

首先通过 Token 间的匹配关系定义可知:只有部分匹配关系可以唯一确定某些特征结构,其他关系则可能与多种特征结构对应.表 3 给出了映射关系 *f*(Token 匹配关系→特征结构),其中,特征结构为图 4 中的结构.

从特征结构到 Token 间匹配关系的映射指:当过程模型中存在某种特征结构时,其对应的完整 Token Log 必定包含满足哪种匹配关系的 Token.例如,当过程模型结构中有单步循环时(如图 4(h)所示),Token Log 中肯定包含 Token *t* 满足 *t* 是一个单步循环标记.在根据特征结构确定 Token 间的匹配关系时,主要确定特征结构库所中填入的 Token 间匹配关系.表 4 给出了完整的映射关系 *f*(特征结构→Token 匹配关系)及各条件的编号(便于下文引用).

Table 3 Mapping from matches between Tokens to substructures shown in Fig.4

表 3 Token 匹配关系到特征结构(图 4)的映射关系

Token 匹配关系		特征结构
单步循环标记		图 4(h)
顺序匹配		图 4(a)
并发匹配		图 4(d)
同步匹配		图 4(e)
多路匹配		图 4(c)
循环匹配	仅消费者相同	图 4(e)、图 4(g)、图 4(h)
	仅生产者相同	图 4(d)、图 4(f)、图 4(h)
	生产者消费者同相同	图 4(b)、图 4(c)、图 4(h)

Table 4 Mapping from substructures in models to matches between Tokens

表 4 特征结构到 Token 匹配关系的映射

特征结构		对应库所中 Token 匹配关系的条件		条件编号
图 4(b)	循环主路径	库所中存在 t_1,t_2 满足循环匹配生产消费者都相同		1
	非循环主路径	库所中存在 Token		2
图 4(c)		t_1 与 t_2 满足多路匹配	两个库所中分别存在 t_1,t_2	3
图 4(d)		t_1 与 t_2 满足并发匹配		4
图 4(e)		t_1 与 t_2 满足同步匹配		5
图 4(f)	循环选择	t_1 与 t_2 满足循环匹配仅生产者相同	库所中存在两个 Token:t_1,t_2	6
	非循环选择	*CID* 不同,生产者相同		7
图 4(g)	循环或同步	t_1 与 t_2 满足循环匹配仅消费者相同		8
	非循环或同步	*CID* 不同,消费者相同		9
图 4(h)		库所中存在单步循环标记 Token		10

3 度量方法

使用 Token Log 的符合性检查方法分为两个部分:使用日志内容检查模型结构的正确性;使用模型特征结构检查日志内容的完整性.第 3.1 节主要介绍衡量模型结构正确性的方法,第 3.2 节详细介绍衡量日志内容完整性的方法.

3.1 检查模型结构的正确性

模型结构的正确性通过将日志 Token 正确填入模型的程度衡量,此处过程模型与 Token 的关系分为能够填入与不能填入:能够填入指 Token 在日志中拥有的所有匹配关系都能够正确填入过程模型;不能填入指 Token 拥有的匹配关系中存在不能正确填入模型的部分.为了判断过程模型是否可以正确填入 Token,首先需要将日志中的所有 Token 间的匹配关系找到,这些关系包括表 3 中的所有匹配关系.例如,表 5 是一个可能是图 1 对应 Token Log 中实例的 Token 列表.

Table 5 A running case in the token log of the process in Fig.1

表 5 图 1 对应 Token Log 中的一个运行实例

CID	Token	*PT*	*CT*	*PEID*	*CEID*	Token	*PT*	*CT*	*PEID*	*CEID*	Case number
3	t_1	*A*	*B*	1	2	t_7	*J*	*D*	6	7	100
	t_2	*B*	*C*	2	3	t_8	*D*	*E*	7	8	
	t_3	*C*	*M*	3	4	t_9	*D*	*F*	7	9	
	t_4	*C*	*J*	3	5	t_{10}	*E*	*G*	8	10	
	t_5	*J*	*J*	5	6	t_{11}	*F*	*G*	9	10	
	t_6	*M*	*D*	4	7	t_{12}	*G*	*H*	10	11	

表 6 展示了表 5 中 Token 间所有匹配关系,其中没有多路匹配关系和循环匹配关系.

Table 6 Matches between tokens in Table 5

表 6 表 5 中 Token 间的匹配关系

匹配关系名称	Token 对	*ap* 权重	个数
单步循环标记	t_5	0.1	1
顺序匹配	(t_1,t_2), (t_2,t_3), (t_2,t_4), (t_3,t_6), (t_4,t_5), (t_5,t_7), (t_6,t_8), (t_6,t_9), (t_7,t_8), (t_7,t_9), (t_8,t_{10}), (t_9,t_{11}), (t_{10},t_{12}), (t_{11},t_{12})	0.5	14
并发匹配	(t_3,t_4),(t_8,t_9)	0.2	2
同步匹配	(t_6,t_7),(t_{10},t_{11})	0.2	2

过程模型正确填入单个匹配关系的程度使用变量 *dap*(degree of accepting pair)表示,且 $dap\in[0,1]$,即,*dap* 的取值不是非 0 即 1.如图 5 中,将顺序匹配关系(t_6,t_8)填入模型时,t_6 不能填入而 t_8 能够填入,则可以将该关系的 *dap* 值设为 0.5 或其他界于 0,1 之间的某个数.在确定单个匹配关系的 *dap* 值时,可以根据对结果精确程度的要求标记匹配关系被正确填入的程度.根据每一个匹配关系的 *dap* 值,可以确定整个实例正确填入过程模型的程度,该值称为 *dac*(degree of accepting case),计算方法如下:

公式 1. 令 l 是日志中的匹配关系种类,m_i 是日志中某一实例中某种匹配关系的个数,$w_i\in(0,1)$是这种匹配关系在符合性检查中的重要程度,*dac* 表示该实例正确填入过程模型的程度,则:

$$dac=\sum_{i=0}^{l}\left(w_i\times\frac{\sum_{j=0}^{m_i}dap_j}{m_i}\right),\text{其中,}\sum_{i=0}^{l}w_i=1.$$

衡量匹配关系正确填入的程度时,不同匹配关系的标准不同.例如,并发匹配关系没有顺序匹配关系不被填入时,对整个实例影响比较大(如果顺序匹配不被填入,则可能是整个运行序列存在问题).所以在衡量 *dac* 时,使用 w_i 表示每一种匹配关系的重要程度,为方便计算,令 $w_i\in(0,1)$.每一个运行实例的 *dac* 值决定了整个日志能够正确填入过程模型的程度,使用 *dal*(degree of accepting log)表示,其计算方法如下.

公式 2. 令 n 是日志中的实例个数,c_k 是每个实例在整个日志中出现的次数,dac_k 是每个实例正确填入过程模型的程度,*dal* 表示整个日志能够正确填入过程模型的程度,则:

$$dal=\frac{\sum_{k=0}^{n}(c_k\times dac_k)}{\sum_{k=0}^{n}c_k}.$$

dal 表示所有匹配关系能正确填入过程模型的平均程度,每一个匹配关系都按照其重要程度参与计算.最终,*dal*∈[0,1],0 表示所有的匹配关系都不能正确填入过程模型,1 表示所有的匹配关系都能够正确填入.例如,将表 6 中的匹配关系填入图 1 模型中得到结果如图 5 所示.

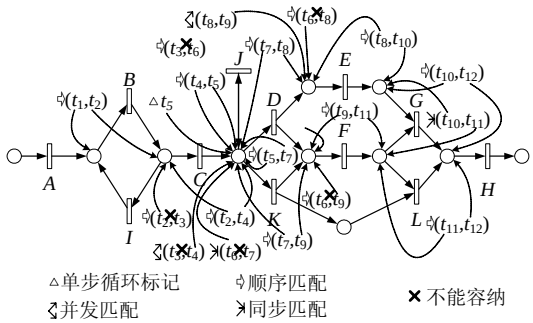


Fig.5 Result of putting tokens in Table 6 into the model in Fig.1

图 5 表 6 匹配关系填入图 1 模型中的结果

如图 6 所示:假设每个匹配关系中有一个 Token 可填入则其 *dap* 是 0.5,两个都填入 *dap* 是 1,且 $w_{\text{单步循环标记}}$=0.1,$w_{\text{顺序匹配}}$=0.5,$w_{\text{并发匹配}}$=0.2,$w_{\text{同步匹配}}$=0.2,则 *dac*=0.8107.*dac*=0.8107 表示这个运行实例中每个匹配关系平均有 81.07%的内容能够被正确填入过程模型.此处计算的只是一个运行实例,而不是整个日志.假设日志中还包含表 2 中 *CID*=4 的运行实例(*dac*=1)且出现 10 次,则最终 *dal*=(0.8107×100+1×10)/(100+10)=0.8279.

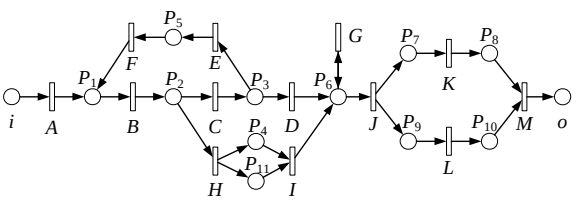


Fig.6 Example model containing all typical substructures

图 6 包含各种特征结构的过程模型实例

3.2 检查日志内容的完整性

如第 2.3 节所说,计算日志覆盖过程模型特征结构的能力是为了避免 Overfitting,Underfitting 及其类似结构对判断的误导,但其前提是已经通过过程模型对于日志的行为正确性初步判断日志与模型可能匹配.

日志内容的完整性通过日志中 Token 匹配关系是否能满足过程模型中每个特征结构的要求衡量.匹配关系满足特征结构要求指,匹配关系中包含了这个特征结构可能产生的所有匹配关系.本文讨论的过程模型中特征结构及其对匹配关系的要求见表 4.

3.2.1 识别过程模型特征结构

获取过程模型的特征结构,关键在于获取这些结构中的库所.循环节点中就包含了单步循环库所、多步循环起始与结束库所和循环中在主路径[23]上的非起始、结束库所.以图 6 为例,P_6 是单步循环库所,P_1 和 P_3 为多步循环的起始与结束库所,P_2 是循环路径在主路径上的普通库所.

在符合性检查中,需要让计算机通过计算获取这些库所信息.本文根据 Petri 网中 *T*-不变量(*T*-Invariant),*S*-不变量(*S*-Invariant)[7]理论对过程模型进行分解,获取特征结构的库所.*T*-不变量,*S*-不变量的定义如下:

定义 9(**关联矩阵、*T*-不变量、*S*-不变量**)[23]:

(1) 所有 Petri-net *PN*=(*P*,*T*,*F*)均可以表示为一个集合 *P* 与集合 *T* 基于集合 *F* 的关联矩阵 *A*;

(2) *T*-不变量是方程 $A\cdot X=0$ 的整数解,其解的本质是模型中所有保持点火变迁数目稳定的循环点火序列,其中,非 0 标记的变迁表示在点火序列中;

(3) *S*-不变量是方程 $A^T\cdot X=0$ 的整数解,其解的本质是模型中所有保持 Token 数目稳定的循环点火序列,其中,非 0 标记的库所表示在点火序列中;

(4) *T*-不变量有解当且仅当过程模型是合理的,*S*-不变量有解当且仅当过程模型是有界的.

本文使用的过程模型均为合理 WF-Net,因此,*T*-不变量、*S*-不变量一定有解.*T*-不变量结果拆分所有的选择结构;*S*-不变量结果拆分所有并发结构.为拆分模型中所有选择与并发结构,在开始库所与结束库所上添加变迁 *Q* 形成循环结构.例如,在计算图 6 不变量时,首先添加任务 *Q* 使得整个模型成为一个大的循环结构,如图 7 所示.

T-不变量、*S*-不变量方程的解都不唯一,但其有一组由 0,1 组成的最小半正解,即文献[24]中提出的 LMST-不变量(legal minimal semi-positive *T*-invariant).图 7 模型最终的 *T*-不变量、*S*-不变量的最小半正解见表 7、表 8.

根据表 7 可以获得循环和选择结构,具体识别过程如下(“条件”指表 4 中条件):

(1) 根据不包含 *Q* 的解确定内部循环,包括{*G*}和{*B*,*C*,*E*,*F*},可确定单步循环库所 P_6,需要满足条件 10;

(2) 根据包含 *Q* 的解确定主路径{*A*,*B*,*C*,*D*,*J*,*K*,*L*,*M*}和{*A*,*B*,*J*,*K*,*L*,*M*,*H*,*I*},根据多步循环与主路径的交叉任务 *B* 和 *C*,确定循环结点库所 P_1 和 P_3,分别满足条件 6 和条件 8.并确定循环主路径库所 P_2,满足条件 1;

(3) 根据两条主路径确定选择分支{*C*,*D*}和{*H*,*I*},并确定选择库所 P_2 和 P_6,分别满足条件 7 和条件 9.

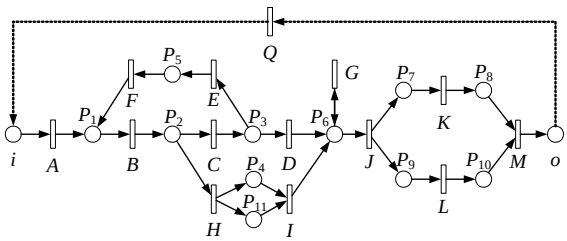


Fig.7 Circular model of Fig.6 after adding *Q*

图 7 图 6 结构添加任务 *Q* 形成整体循环结构

Table 7 Legal minimal semi-positive *T*-invariant of the model in Fig.7

表 7 图 7 模型 *T*-不变量最小半正解

A	*B*	*C*	*D*	*G*	*J*	*K*	*L*	*M*	*H*	*I*	*E*	*F*	*Q*
1	1	1	1	0	1	1	1	1	0	0	0	0	1
1	1	0	0	0	1	1	1	1	1	1	0	0	1
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	1	1	0

Table 8 Legal minimal semi-positive *S*-invariant of the model in Fig.7

表 8 图 7 模型 *S*-不变量最小半正解

i	P_1	P_2	P_3	P_6	P_7	P_9	P_8	P_{10}	*o*	P_4	P_5	P_{11}
1	1	1	1	1	1	0	1	0	1	1	1	0
1	1	1	1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	1	0	1	1	1	1	0
1	1	1	1	1	0	1	0	1	1	0	1	1

在表 8 结果中,标记为 0 的所有库所都是在某些点火路径上不被使用的.根据 *S*-不变量定义可知:它们是并发或同步结构中的库所,并可根据原过程模型确定其具体类别,过程如下:

(1) 确定所有标记为 0 的库所,它们是可能处于同一个并发结构的库所组合,将这样的组合构成的集合称为潜在并发库所组合集(set of potential parallel places’ combination,简称 PC):

$$PC=\{\{P_9,P_{10},P_{11}\},\{P_9,P_{10},P_4\},\{P_7,P_8,P_4\},\{P_7,P_8,P_{11}\}\}.$$

定义 10(潜在并发库所组合集). 令合理 WF-net $PN=(P,T,F)$的初始与结束库所分别为 i,o,任务 Q 以 o 为输入、以 i 为输出,集合 $PC=\{PC_1,PC_2,\ldots,PC_n\}$称为 PN 的潜在并发库所组合集当且仅当:$PC_i\in PC$ 且 PC_i 为 $PN'=(P,T\cup\{Q\},F\cup\{(o,Q),(Q,i)\})$的 S-不变量最小半正解的某个解中所有标记为 0 的库所组合.

定义 11(并发分支库所集,set of places in parallel branch,简称 PPB). 令 $PN=(P,T,F)$是一个合理 WF-net,PN 中一个并发结构的分支中所有的库所组成的集合称为 PN 的一个并发分支库所集,缩写为 PPB.

定理 1. 令 $PC=\{PC_1,PC_2,\ldots,PC_n\}$是 $PN=(P,T,F)$的潜在并发库所组合集,且$\forall p\in P'=\{P_1,P_2,\ldots,P_n\}$满足 $p\in P$. 令 $P_i\in P',PC_j\in PC$,若:$(P_i\in PC_j\wedge\forall P_k\in P(k\neq i)\Rightarrow P_k\in PC_j)\wedge(P_i\notin PC_j\wedge\forall P_k\in P(k\neq i)\Rightarrow P_k\notin PC_j)$,则 P'可能是 PN 的一个 PPB.如果原模型中包含一个包含且仅包含 P'中所有元素的顺序结构,则 P'是一个 PPB.

证明:在定理中,$P_i\in PC_j\wedge\forall P_k\in P(k\neq i)\Rightarrow P_k\in PC_j$ 指:如果 P_i 在某一个潜在并发库所组合中,则 P'中其他元素都一定在这个组合中.$P_i\notin PC_j\wedge\forall P_k\in P(k\neq i)\Rightarrow P_k\notin PC_j$ 指:如果 P_i 不在某个组合中,则 P'中其他任意的元素都不在这个组合中.潜在并发库所组合就是:所有可能在同一个并发分支上的所有库所,但是可能由多个并发结构的并发分支组成,但是每一个并发分支的所有库所都作为一个整体存在,作为一个整体存在于潜在并发库所组合中的、还有共享并发分支的多个并发结构中非共享并发分支的其他所有并发分支的库所组合.例如,图 1 中从 p_3 到 p_9 的结构为两个共享并发分支的并发结构,其 S-不变量中包含两个解:$\{p_4,p_6,p_8\}$和$\{p_5,p_7\}$,其中,$\{p_4,p_6,p_8\}$是多个并发分支非共享分支的组合.此时,$P'=\{p_4,p_6,p_8\}$就不是 PPB,即原模型中不存在包含且仅包含$\{p_4,p_6,p_8\}$的顺序结构. □

定理 2. 令集合 P 与 P'是合理 WF-net PN 的两个 PPB,则 P 与 P'属于 PN 的同一个并发结构当且仅当:

$$\forall PC_j\in PC,若\exists P_i\in P\wedge P_i\in PC_j\Rightarrow\nexists P_k\in PC_j\wedge P_k\in P'.$$

证明:条件$\exists P_i\in P\wedge P_i\in PC_j\Rightarrow\nexists P_k\in PC_j\wedge P_k\in P'$指:$PPB$ P 与 P'不能同时出现在一个潜在并发库所组合中.根据潜在并发库所组合(即 S-不变量)的定义可知:两个 PPB 出现在同一个潜在并发库所组合中,则必然不是同一个并发结构中的不同分支;同时,如果两个 PPB 不属于同一个并发结构,则一定会同时出现在某个潜在并发库所组合中.因此,根据前两个结论可知,如果两个 PPB 总是不出现在同一个潜在并发库所组合中,则必定是同一个并发结构的不同分支. □

(2) 根据定理 1,从上述 PC 中获取的并发分支库所集有$\{P_9,P_{10}\}$,$\{P_7,P_8\}$,$\{P_4\}$和$\{P_{11}\}$;根据定理 2,属于同一个并发结构的并发分支库所集组合为$\{P_9,P_{10}\}$与$\{P_7,P_8\}$,$\{P_4\}$与$\{P_{11}\}$.

(3) 最后在并发分支的库所中,根据原过程模型找到所有的并发(P_7,P_9)和同步(P_8,P_{10})库所对,它们应满足条件 4 和条件 5.其中,只包含一个库所的并发分支库所集均为特殊隐式库索(P_4,P_{11}),它们应满足条件 3.

(4) 模型中其他没有在上述库所中且不是开始与结束库所的库所,均为主路径上普通库所,其满足条件 2.

3.2.2 度量日志覆盖能力

使用日志平均覆盖特征结构行为的能力衡量日志完整性.特征结构中,库所被体现的程度使用 dcp(degree of covering place)表示,$dcp\in[0,1]$.例如,日志需要满足的条件只能达到一半,则令 $dcp=0.5$.根据每个库所的 dcp 值确定该类特征结构中所有库所被体现的平均程度.每种特征结构的重要程度不同,使用 w 表示其权重,$w\in[0,1]$. dcm 定义如下:

公式 3(degree of covering model). 令 PN 是合理 WF-net,TL 是 Token Log,PN 有 n 种特征结构,m_i 是每种特征结构中对应库所或库所对的个数,w_i 是每种特征结构对符合性检查的重要程度,dcp 是每种结构中每种库所被 TL 体现的程度,则 TL 体现 PN 的程度 dcm 为

$$dcm=\sum_{i=0}^{n}\left(w_i\times\frac{\sum_{j=0}^{m_i}dcp_j}{m_i}\right),其中,\sum_{i=0}^{n}w_i=1.$$

例如,计算表 5 实例对图 1 模型覆盖程度时(假设表 5 内容即为 Token Log 的内容),首先要通过第 3.2.1 节

的方法获取所有特征结构中的库所,结果见表 9.

Table 9 Places of special substructures of the model shown in Fig.1

表 9 图 1 模型中特征结构对应的特殊库所统计

特征结构		库所	条件	个数(m)	权重(w)
单步循环		p_3	10	1	0.1
多步循环	选择库所 或同步库所	p_2 p_1	6 8	2	0.1
选择结构	选择库所 或同步库所	p_3,p_7 p_5,p_9	7 9	4	0.4
并发结构	并发库所 同步库所	(p_4,p_5), (p_5,p_8) (p_6,p_7), (p_7,p_8)	4 5	4	0.4

由表 9 可知,该计算中非循环的选择结构与并发结构对判断结果更为重要.在计算 *dcm* 前,首先将所有 Token 关联到对应的库所中(如图 8 所示),再进行计算.

根据填入库所中的 Token 判断日志是否满足库所的要求并确定其 *dcp*:

(1) p_3 中 t_5 是单步循环标记,满足条件,*dcp*=1;

(2) p_2 只有 t_2,不满足条件,*dcp*=0;p_1 中只有 t_1,同样不满足条件,*dcp*=0;

(3) p_3 只包含 t_4,*dcp*=0.5;p_7 只包含 t_{11},*dcp*=0.5;p_5 与 p_9 也是 *dcp*=0.5;

(4) (p_4,p_5)与(p_6,p_7)中都包含了匹配的 Token,*dcp*=1;(p_5,p_8)与(p_7,p_8)的 p_8 没有 Token,不满足要求,*dcp*=0.

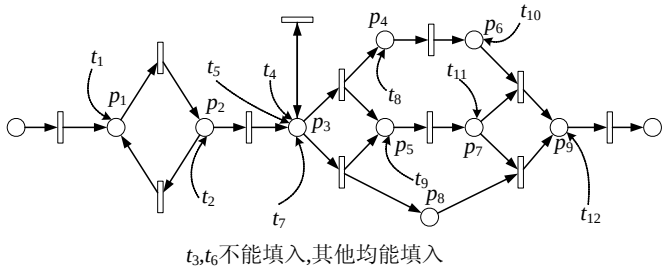


Fig.8 Result of putting tokens in Table 5 into the model in Fig.1

图 8 将表 5 中 Token 填入图 1 模型中

确定各种特征结构的平均 *dcp* 后,算得 *dcm*=0.5,表示日志能够平均覆盖模型中每种特征结构的一半行为,日志很不完整.但表 5 的日志仅含一个实例,若添加表 2 的 *CID*=4 的实例,则过程中被执行的路径将如图 9 所示.

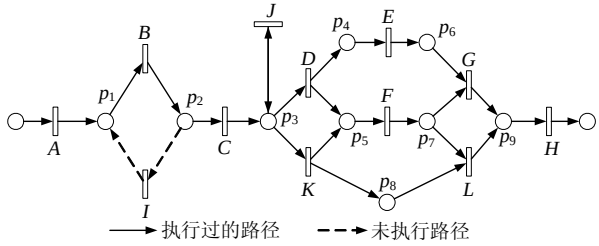


Fig.9 Running paths of Fig.1 after adding the case *CID*=4 of Table 2 into the log in Table 5

图 9 表 5 日志中添加表 2 中 *CID*=4 的实例后,图 1 模型的运行路径

此时,除了循环结构的平均 *dcp* 为 0,其他均为 1,所以 *dcm*=0.9.如果再添加表 2 中 *CID*=5 的实例,则模型中所有路径都会被标记执行,即库所中包含满足所有条件的 Token,其 *dcm*=1.该结果表示日志包含了所有模型能够生成的日志内容.

如果模型是 Overfitting 结构,则日志覆盖其允许行为的程度会很低,因此可以判定符合性较低.但如果模型是 Underfitting 结构,此时日志恰好包含其能够覆盖其行为(*dcm*=1),且模型能够生成所有日志内容(*dal*=1).此时,该如何排除对 Underfitting 结构的误判呢?排除对 Underfitting 结构误判的关键在于计算 *T*-不变量、*S*-不变量时,计算的并不是 Underfitting 结构的本身,而是实际录入的模型.如果结构是 Underfitting 结构,则会包含很多重复的任务,如图 3 中的 N_3.但是在本文的方法中,录入变迁与库所的关联矩阵时,不会区分重复的任务,即所有重复任务都最终对应一个任务,如图 3 中的 N_3 录入的是图 10 中的结构.

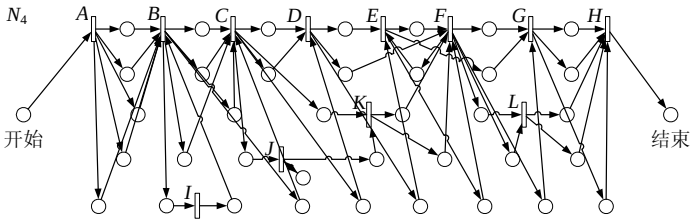


Fig.10 Model the checking method recognized by reading N_3 in Fig.3

图 10 图 3 中 N_3 实际录入的关联矩阵对应的模型,即去除重复任务的结构

通过去除重复任务的关联矩阵计算出的 *S*-不变量将表明:模型中包含很多的特殊隐式库所结构,则要求日志中包含满足这些隐式库所要求的 Token 匹配关系.但日志中并没有这些 Token 匹配关系,则最终的 *dcm* 会比较低,由此排除对 Underfitting 结构的误判.

4 综合 *dal* 与 *dcm* 判断符合性

dal 表示过程模型能够生成日志内容的能力,*dcm* 表示日志覆盖模型所有允许行为的能力.过程模型与日志匹配的充分必要条件为:(1) 模型要能够正确填入日志中的所有内容(*dal*=1);(2) 日志要包含模型能够生成的所有内容(*dcm*=1).

(1) 仅有 *dal*(模型正确性)是不够的.过程模型如果与日志内容不匹配,则可能只能正确填入日志中很少的内容,或者是可以正确填入日志中所有的内容,但同时还可以填入很多日志中没有的内容(即 Overfitting 结构).因此,*dal* 值大于标准值,只是模型与日志匹配的必要条件而不是充分条件;

(2) 仅有 *dcm*(日志完整性)是不够的.日志与模型不匹配,则可能日志中包含的模型能够生成的日志内容很少,或者日志包含了所有模型能够生成的日志内容,但是同时包含了很多模型不能生成的日志内容.因此,*dcm* 值大于标准值,只是日志匹配模型的必要条件而不是充分条件.

但即使模型能够生成日志所有内容,日志也恰好包含模型能够生成的所有内容,如果模型是 Underfitting 结构,也会影响对符合性的判断.由于合理的工作流网中有重复的任务,因此,本文在计算 *dcm* 时删除模型中所有的重复任务(具体解释见第 3.2.2 节)以排除 Underfitting 结构对判断的影响.

理论上,只有当 *dal*=1 且 *dcm*=1 时才判定过程模型与日志完全匹配.但实际比较过程中,由于某些因素,即使是完全匹配,其 *dal* 和 *dcm* 也达不到 1.表 10 列出了这些影响因素.

Table 10 Factors of the model and log affecting their match and their impacts

表 10 模型与日志中影响完全匹配的因素及其表现

影响因素	日志不完整	日志不纯净	隐藏的任务	不可见任务	重复任务
影响表现	*dcm* 降低	*dal* 降低	*dal* 降低	*dcm* 降低	*dcm* 降低

日志不完整指参与检查的日志不是某过程模型的完整日志,没有包含所有运行实例或者某些运行实例中有内容缺失.因为内容缺失,日志对模型特征结构行为的覆盖程度也会降低,即 *dcm* 降低.日志不纯净指日志中包含了其他过程模型中的运行实例或某些运行实例中包含错误的任务信息,导致模型不能完全正确填入日志内容,即 *dal* 降低.模型中存在隐藏任务指,这些任务在日志中会被记录,但是在模型中没有体现,这样会产生日志不

纯净的效果,即 *dal* 降低.不可见任务指存在于模型中却没有被日志记录下的任务,这样的任务会导致日志无法覆盖模型的所有特征结构的行为,从而降低 *dcm* 值.在计算 *dcm* 时,为了排除 Underfitting 结构的影响,将模型中所有的重复任务看作是同一个任务,所以如果模型中真实存在重复任务而又不是 Underfitting 结构,则会导致日志无法完全覆盖模型特征结构,即 *dcm* 被降低.

根据 *dcm* 的影响因素确定其能容忍的最小值,根据 *dal* 的影响因素确定其能容忍的最小值.这些影响因素与日志和模型的具体情况密切相关,不同的日志与模型受这些因素的影响程度不同,最终确定 *dal* 与 *dcm* 需要参考具体的日志与模型.

5 使用 Token Log 的优点分析

第 1.1 节简单介绍了基于事件日志的符合性检查方法,该方法存在误判的情况.例如:过程中有较多选择结构时,与其对应的日志进行符合性检查时会产生如下情况:(1) 模拟执行时,所有任务序列均可被再现,即 *Fitness* 达到最大;(2) 选择结构的输入任务执行后,接着只能执行一个选择分支,则必然有当前执行的任务序列外的其他任务处于可能被触发状态;又由于该结构中的选择结构较多,则所有任务串中平均可能被触发的其他任务会比较多,即 *Appropriateness* 较低;(3) 预测结果应该是该模型与日志完全符合,但最终结果会受到 *Appropriateness* 的影响,产生误判情况;或者,使用由多个并发结构组成的模型与只包含顺序结构的事件日志(二者有相同的任务集合)进行检查时,会有如下情况:(1) 在模拟执行每一个任务序列时,都会有极少数的并发结构的任务无法执行,但是多数任务都能够顺利执行,即 *Fitness* 值会接近最大值;(2) 执行所有非并发结构任务时都不会触发该任务序列以外的其他任务,即 *Appropriateness* 值也会接近最大值;(3) 根据前两者判断,该模型与日志有较高的符合性,但实际上该模型与日志的结构存在巨大差异,不可能有较高的符合性.

结合误判问题及第 2.3 节介绍的使用 Token Log 的动机,本节将详细介绍引言中提出的在符合性检查中使用 Token Log 的 4 个优点.

5.1 全面的检查方案

在符合性检查中,日志代表了模型实际运行的行为,而模型代表了产生日志行为的原结构.图 11 表示了使用两种日志进行符合性检查的不同过程.实线代表使用 Token Log,虚线代表使用事件日志.使用事件日志进行检查时,以日志行为为基础判断日志对结构描述的完整性,将判断的依据局限在已有日志行为中.如果参与检查的日志不会导致模型产生非当前任务序列的行为,而模型真实存在日志内容外的其他行为(图 11 中的模型允许的其他日志内容),则会误判日志已经完全反映了模型允许的行为.

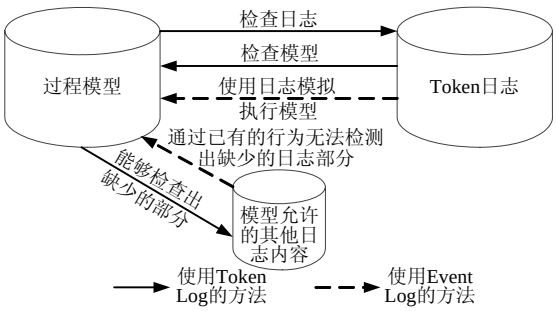


Fig.11 Differences between conformance checking with Token Log and Event Log

图 11 使用 Token Log 与使用 Event Log 进行符合性检查的方法示意图

使用 Token Log 时,基于日志与模型符合的充分必要条件,将符合性检查拆分为两部分:以实际行为为准检查模型结构的正确性;以模型结构为准检查日志行为的完整性.检查中不会有参考行为的遗漏(图 11 中 More Log 部分).

5.2 化动态模拟为静态分析

使用事件日志进行符合性检查时,要动态模拟执行参与检查的过程模型,即参照已有日志内容一步步推进模型的执行.然而,真实的过程模型执行是一个自然且难以宏观控制的过程,任务的并行由真实的多线程或者多进程,甚至是多处理器完成.实现该检查过程是再现过程模型的所有可能状态并且在每个状态上执行统计与分析的过程,其空间与时间开销非常大.

然而,使用 Token Log 进行符合性检查时不会遇到这些困难.整个过程中只有静态分析而没有动态执行.使用 Token Log 的过程中,可预见的实现难点只有计算 Token 匹配关系和分析模型特征结构.判断 Token 匹配关系时只需要判断同一个实例中 Token 的匹配关系,而且日志中 Token 个数与参与执行的任务个数呈线性相关,因此花费的空间与时间代价不会达到再现过程模型状态空间.因此,使用 Token Log 的符合性检查过程相对简单.

5.3 简化判定过程

使用事件日志进行符合性检查的方法中,模型与日志的匹配程度通过包含 5 个度量参数的组合结果衡量.但是根据这些变量的定义,无法直接通过有一个模型与一个日志计算出来的这些结果判定其是否匹配.只能通过比较多组不同模型与日志计算出来的值,确定哪些模型与日志更可能匹配.在多组数据比较的时候,需要权衡各变量的影响权重,不可简单地看哪组数据中较大的变量多就更加匹配.而且,如果参与比较的模型与日志比较多,就更难从中选出匹配的组合.

使用 Token Log 时,最终衡量符合性的变量只有两个:*dal* 与 *dcm*.可先计算一个衡量参量初步判定模型与日志是否匹配,再决定是否需要计算另一个衡量参数.

5.4 定制判定标准

参与符合性检查的日志与模型可能都存在一定的问题,如日志不完整、模型中有不可见任务或者重复任务等,在计算符合性的时候需要将这些因素纳入考虑,要在一定程度上根据日志与模型的实际情况,灵活调整对判定标准.使用 Token Log 检查日志与模型的符合性,就可以满足这种要求.在检查中,不仅可以根据对符合性要求的不同严格程度确定不同的 *dap*,*dcp* 及两种 w_i 的取值,还可以根据日志与模型的实际情况确定对 *dal* 与 *dcm* 的容忍程度.使用事件日志时,计算衡量参量的方法是唯一的,没有将模型与日志的特殊情况纳入考虑.如果一个不完整的日志与一个完整的日志同时与一个模型进行符合性检查,使用事件日志得到的结果是只有完整的日志是符合的,而使用 Token Log 得到的结果可能是两个日志都是符合的,更加符合实际情况.

6 对模型增强的促进

判定日志与模型相互符合后,可以根据日志记录的过程行为,分析与发现模型中存在的问题,对过程模型改进.在本文使用的符合性检查方法中,分析与发现模型中可能存在的问题包括:

(1) 日志中信息不完整的 Token,指没有生产者或者没有消耗者的 Token.这类 Token 表明模型中存在行为不可见的任务(invisible task),指原结构中存在这些任务,但是没有在日志中记录下来.这种问题可能出现在实现软件系统的过程中,发现与修正这些问题有助于增强过程行为的可视化,便于发现与解决问题;

(2) 日志中 Token 因模型中没有对应任务而无法填入,说明任务在行为中有记录,但是在模型中没有出现.这种情况说明过程模型在实现阶段发生了变化,但没有在建模工作中更新.在系统管理与维护过程中,系统模型与现实系统保持一致非常重要;

(3) 日志中 Token 因没有对应库所而无法填入模型,说明任务间的顺序关系没有在模型中体现.日志中 Token 间匹配关系无法在模型中找到对应的结构,也说明了同样的问题;

(4) 模型中没有填入 Token 的库所可能表示该库所所在的执行路径没有被执行或者有异常情况.在选择结构的库所中,还可以统计流向不同分支的 Token 比例,可能对应于不同业务流程被选择的不同概率.可以针对这些统计信息,对不同业务的条件作修改,以达到业务平衡或突出某种业务的执行.

例如,使用只包含表 5 实例的 Token Log 与图 1 的过程模型进行符合性检查,得到 *dal*=0.8107,*dcm*=0.5,考虑到日志的极其不完整性和模型中不可见任务存在的可能性,判定这样的结果即表示其二者匹配.然后,按照日志修改模型,并将日志中所有的 Token(黑色点圆)填入模型中,即可得到图 12.

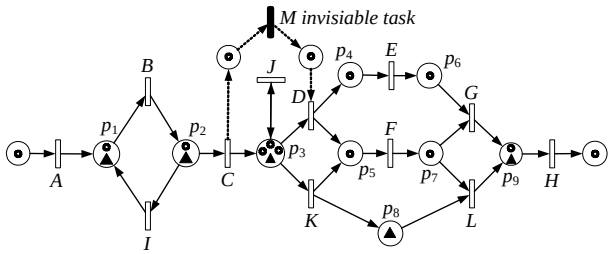


Fig.12 Putting all tokens in Table 5 into the modified model of Fig.1

图 12 将表 5 中所有 Token 填入修改后的图 1 过程模型中

填入 Token 的过程中,发现 t_3 与 t_6 无法找到一个相同的相关任务,若这个任务存在,两者间的顺序关系便成立,这个任务很可能就是一个不可见任务.添加了潜在的不可见任务 *M* 后,t_3 与 t_6 均可以填入模型中.图中黑色实心三角形表示库所中还应该填有的其他 Token,这些 Token 集中在选择结构的库所中,表示在已有的日志行为中,某些业务路径还没有被执行.例如,任务 *B* 到任务 *I* 的选择、任务 *C* 到任务 *K* 的选择.根据这些信息,可以调整过程中相关业务逻辑,促使某些没有被执行的业务被执行.

使用 Token Log 进行符合性检查的过程中,还有很多其他的统计信息对模型的增强有着重要的指导作用.其关键是针对 Token 填入模型时发生的各种情况具体分析,发现模型中对应的问题,从而给出解决方案.

7 工具与实验

本节主要介绍基于开源工具 PIPE 与 ProM 开发的实验工具、算法及实验过程,主要包括在 Petri 网建模与分析工具 PIPE 中开发的 Token Log 生成插件和在过程挖掘实验工具 ProM 中开发的 Token Log 实例去重插件与符合性检查插件.

7.1 生成Token Log

PIPE 是开源的 Petri 网建模与分析工具,支持用户自定义插件.为验证过程模型执行中能够产生并记录符合 Token Log 格式的日志内容,结合 PIPE 工具开发了模拟执行过程模型并生成 Token Log 的插件.

7.1.1 模拟执行算法

该插件的目标是:使用单线程模拟执行过程模型的并发、选择和循环等结构,同时实现一个过程模型的多个实例同时执行.生成的日志是多个运行实例内容交叉的结果.为满足该需求,本文提出了模拟执行算法.本文将描述算法的核心内容,使用到的相关数据结构及方法仅简单描述,内容如下:

(1) 过程模型通过库所(place)列表(PlaceList)和任务(task)列表(TaskList)表示,库所记录它所有的输入任务(InputTasks)和输出任务(OutputTasks),任务记录它所有的输入库所(InputPlaces)和输出库所(OutputPlaces);

(2) 每个任务执行条件为其输入库所中均填有 Token,该 Token 与第 2.1 节定义相同;

(3) 在任务执行后,所有填入 Token 的库所使用非空库所队列(NonEmptyPlaceQueue)记录,如果进入非空库所队列的库所并没有可执行的输出任务,则移入库所等待队列(PlaceWaitingQueue);

(4) 获取库所可执行输出任务的方法 *getReadyTask*(·),判断标准是该任务的所有输入库所中均含 Token;如果该库所有多个输出任务且都已经至少执行过 1 次则随机选择一个.

(5) 模拟执行任务的方法 *executeTask*(·),其效果是从任务所有输入库所中移除一个 Token,并向所有输出库所中填入新 Token;其输出所有被使用的 Token;

(6) 打印被使用的 Token 信息的方法 *printTokens*(·),只有在 Token 被消耗时才打印;

(7) 初始化过程模型的输入库所 *initModelInput*(·)方法,其工作是在模型输入库所中填入新 Token,并将输入库所添加到 NonEmptyPlaceQueue 中.在执行过程中,随机执行该方法则可实现多个实例同时运行的效果.如果其执行的次数达到约定的实例个数,则不再执行它;

(8) 非空库所队列的 *removeEmpty*(·)用来移除不含 Token 的库所,*addNewNonEmpty*(·)用来添加任务执行后被填入新 Token 的库所;库所等待队列也有 *removeEmpty*(·)方法,作用与非空库所队列的相同,它的 *getReady*(·)方法则用来获取队列中包含可执行输出任务的库所.

根据以上数据结构与方法定义,模拟执行过程模型的算法见算法 1.

算法 1. 模拟执行过程模型的算法.

```
Input: PlaceList; TaskList; numCase;
Global: NonEmptyPlaceQueue; PlaceWaitingQueue;
Related Methods: getReadyTask(·); executeTask(·); initModelInput(·); printToken(·);
Output:Token Log
Begin:
  initModelInput(·);
  while ((Place p=NonEmptyPlaceQueue.getHead(·))!=null){
    Task t=p.getReadTask(·);
    if (t!=null){
      UsedTokens tokens=t.executeTask(·);
      printToken(tokens);
      NonEmptyPlaceQueue.addNewNonEmpty(·);
      NonEmptyPlaceQueue.removeEmpty(·);
      PlaceWaitingQueue.removeEmpty(·);
      if (!PlaceWaitingQueue.isEmpty(·)){
        NonEmptyPlaceQueue.add(PlaceWaitingQueue.getReady(·));
      }
      if (numCase>0){
        if initModelInput(·) by random is OK then numCase=numCase−1;
      }
    }else{
        PlaceWaitingQueue.add(p);
      }
    }
End
```

算法核心过程是不停地从非空库所队列中取出库所,获取可执行任务并模拟执行,其关键是维护非空库所队列.非空库所队列的改变均发生在任务执行后.首先,任务执行可能需要非当前选中库所中的其他 Token,则要将新的空库所移除队列.任务执行后会向某些库所中添加新的 Token,则要将新非空库所加入队列.任务执行后,可能导致某些非空库所没有可执行的输出任务,则要将该非空库所加入到库所等待队列.库所等待队列用来保证非空库所在确定可以执行后再添加到非空库所队列中,避免直接加入后导致的下次遇到时还是没有可以执行的输出任务.因此在任务执行后,还需要将库所等待队列中有可执行输出任务的库所添加到非空库所队列中.

为实现模拟多个实例同时运行的效果,在执行过程中添加了随机启动新实例的过程.如果启动的实例个数达到预设值,则停止启动新实例.为实现交叉打印 Token Log 信息,所有的运行实例都使用同一个非空库所队列.

通过 Token 携带的 CaseID 信息,有效管理不同实例的运行状态,同时为每一个运行实例维护一个时钟信息.

7.1.2 实验插件

使用上述模拟执行算法在 PIPE 中开发了模拟执行过程模型且生成 Token Log 的插件,图 13 为使用该插件模拟执行 100 个病人去医院看病的过程所打印的 Token Log(为了使生成的 Token Log 格式整齐,在导入图 1 模型后,将任务名称修改成两个字),且每列均使用虚线框标记了其内容名称.

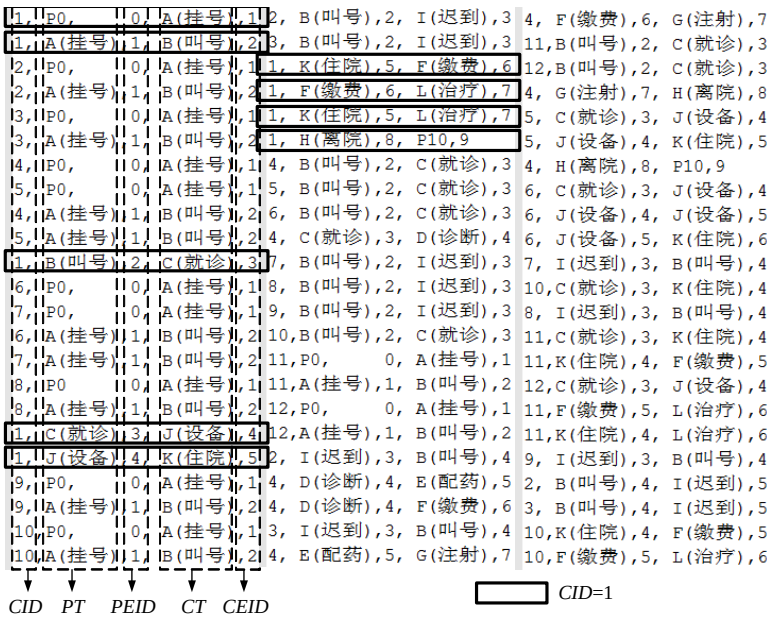


Fig.13 Part of the Token Log of the model in Fig.1

图 13 图 1 模型生成的 Token Log 部分内容示例

如图 13 所示,日志中不仅包含了每个 Token 的 5 个信息,同时,所有的 Token 也按照出现的顺序排列:

- 首先,同一个实例的 Token 按照在系统的产生的顺序排列.例如,所有 *CID* 为 1 的 Token 均按照产生顺序出现(黑色实线框标记),该顺序正是其 *PEID* 增长的顺序;
- 再者,该系统是一个医院的门诊系统,每一个实例代表一个病人看病的过程.病人看病都需要排队,每个病人都必须在所有先于其挂号的病人完成叫号之后参与叫号.如果病人迟到,则重新参与叫号,需要等所有在其被迟到处理前参与叫号与挂号的病人被叫号后才能再次被叫号.

该日志将被用于与图 1 模型的符合性检查中.

7.2 Token Log预处理

上述生成的日志中包含 100 个实例,但很多实例是相同的(看病流程相同).因此在用于符合性检查前,需要对日志进行预处理,将所有相同的运行实例合并,并统计其出现次数.本节首先描述日志预处理算法,然后描述基于该算法在 ProM 中开发的预处理插件.

7.2.1 Token Log 预处理算法

该算法用于统计 Token Log 中各交叉出现实例出现的次数.为快速统计出相同实例,首先遍历所有的日志内容,将同一个实例的内容添加到同一个链表中.向已有链表中插入 Token 的过程见算法 2.

算法 2. 向实例的 Token 链表中插入 Token 的算法.

Input: Token *token*; LinkedList⟨Token⟩ *tokenList*;

Output: The last token in of *tokenList*.

```
Begin:
  lastToken=tokenList.getLast(·);
  if (lastToken.caseID==token.caseID){
    tokenList.getFirstFromTail (where tmp.CT==token.PT and tmp.CEID==token.PEID);
    tokenList.insertAfterTmp(tmp,token);
    if (token.isTail(·))
      lastToken=token;
  }
  return lastToken;
End
```

按上述方法添加 Token 时,链表里的 Token 会按照执行顺序排列,如果两个实例内容相同,则对应链表内容必然相同;如果两个实例相同,则其链表最后一个 Token 的 *PEID* 和链表长度必然相同.首先,将所有实例的链表按照其长度与最后一个 *PEID* 分配到不同的小组中,则只有在同一个小组中的实例才可能内容相同.这样不仅减少了实例间比较的次数,同时可以使用多线程并行处理各小组实例内容.在数据量非常大的情况下,还可以使用多台机器分别处理多个小组以充分提高计算效率.整个预处理过程的算法表示见算法 3.

算法 3. Token Log 的预处理算法.

```
Input: Token Log;
Output: Numbers of cases.
Begin:
  for (Token t in Token Log){
    if (LinkedList for t doesn't exist){
      Create LinkedList for the case of t;
    }
    tmpToken=Insert t into the LinkedList for the case of t;
    if (tmpToken is the last one of the case){
      Split the list by tmpToken's PEID and lengh into different Groups;
    }
  }
  Handle different groups separately with multi-threads, multi-processes, or multi-computers);
End
```

本文提出的预处理算法,通过:

(1) 整理实例中 Token 执行顺序;

(2) 将实例内容分组和

(3) 使用节点内容不断将链表分组

的方法,有效避免了直接比对各实例内容是否相同的低效做法,提高了日志预处理效率.

7.2.2 ProM 中预处理插件

图 14 展示了根据上述预处理算法在 ProM 中开发的预处理插件处理图 14 日志的结果.结果表明:实验中使用的日志文件中共有 27 个不同的实例,出现次数如柱状图所示.

Fig.14 Statistics of merging same cases in the log

图 14 日志查重的统计结果

7.3 计算*dal*和*dcm*

计算匹配程度分两个步骤:计算模型正确填入日志的程度(*dal*);计算日志覆盖模型所有可能行为的程度(*dcm*).根据第 3 节描述的方法,在 ProM 中开发了对应的插件,输入是过程模型和预处理之后的 Token Log,输出包括 *DAL* 结果(如图 15 所示)和 *DCM* 结果(如图 16 所示).

图 15 展示了过程模型匹配所有 Token 间关系的结果.左边的表格统计了所有实例的出现次数,包含的 Token 间关系类型和数量,能被模型再现的数量及相关 *DAP* 值.图 16 的右边展示了日志包含各种特征结构的程度(*DCP*)值与由此计算出的日志包含整个模型的程度(*DCM*)值.

实验中,使用医院流程模型的 Token Log 进行符合性检查.为避免日志与模型完全一致,事先将 Token Log 部分信息删除,结果得出 *DAL*=0.995,*DCM*=0.857.根据该结果可以判定,日志与模型相符合.判定结果与事实相符.实验结果表明了本文提出的基于 Token Log 的符合性检验方法的可行性与正确性.在实验中,每种匹配关系被模型再现的程度值是预先设定的,其影响每个实例 *DAC* 值的比重也是按照其出现的次数占所有匹配关系出现次数比重计算的,因此,在结果界面上没有调整相关匹配程度定义与比重定义的按钮.同样,在计算 *DCM* 的过程中,所有的计算参数都是预先设定的,没有给出调整的操作区域,在后续的研究中将进一步完善实验工具.

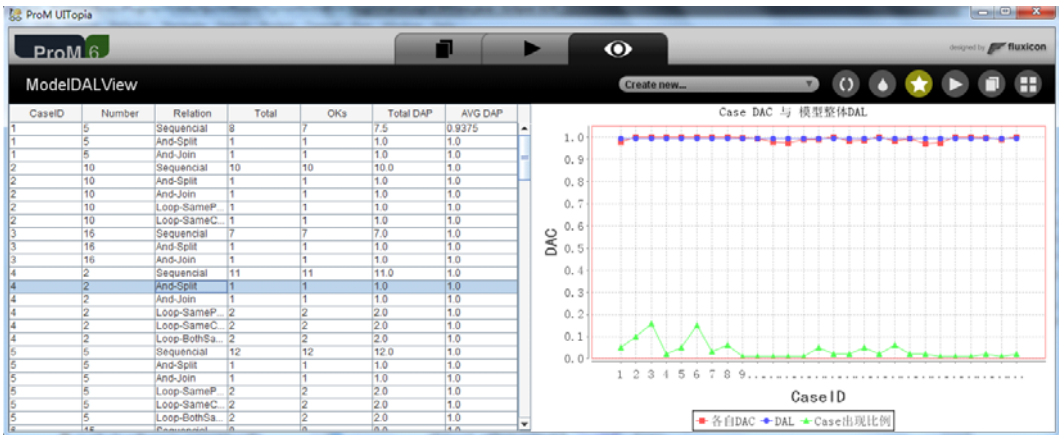


Fig.15 Statistics of the degree of the model in constructing the token log

图 15 过程模型生成 Token 匹配关系的能力统计

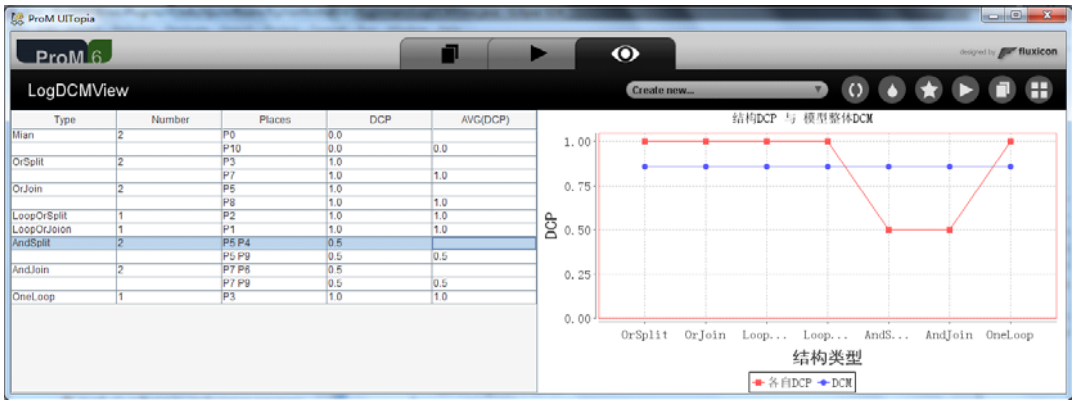


Fig.16 Statistics for the degree of the log covering the model

图 16 日志覆盖模型的 DCM 计算结果

8 总结与展望

本文提出了一种基于新型日志的符合性检查技术.符合性检查技术用于检查过程模型与系统日志的符合程度.在已有的符合性检查方法中,使用的系统日志为事件日志,并通过按照事件日志内容模拟执行过程模型的方法计算符合程度.该方法的主要缺点是:将模型结构的检查范围局限在已有的事件日志范围内,并没有判断日志行为的完整性,由此产生各种误判情况.例如,如果模型中包含大量选择结构,则即使日志是模型本身的日志,也会因为模拟执行较多任务时都会触发当前序列外的其他任务而误判日志与模型的符合性较低.或者,如果模型中只包含少数的并发结构和多数的顺序结构,则即使日志只包含顺序结构的内容且非该模型对应日志时,也会因为在模拟执行时只有个别任务会导致模型无法继续执行,但其他多数任务可以执行而误判日志与模型有较高的符合性.

本文提出的基于 Token Log 的符合性检查技术,将检查过程分为使用日志内容检查模型结构的正确性和使用模型特征结构检查日志内容的完整性两个部分,并使用统计结果 *dal* 表示模型结构匹配日志内容的程度和统计结果 *dcm* 表示日志内容覆盖模型行为的程度,使得检查过程更加全面和清晰,检查结果更加准确.同时,本文提出的新技术可以通过计算结果直接判断模型与日志是否符合,简化了已有技术通过权衡多个计算结果以判断是否符合的过程.本文不仅结合实例详细阐述了使用新型日志进行符合性检查的过程,同时通过实验验证了该方法的可行性与准确性.在实验中,提出了模拟执行过程模型以生成 Token Log 的算法和统计 Token Log 中相同实例个数的 Token Log 预处理算法.

在本文的研究工作基础上,还可以进一步探究在真实工作流系统中实现获取 Token Log 和存储 Token Log 的方法,为大规模应用本文提出的符合性检查技术提供数据基础.在日志数量非常庞大时,如何更加高效地实现日志中相同实例的统计也值得深入研究.同时,在本文提出的符合性检查技术基础上,可以进一步细化计算过程,添加根据模型与日志特点确定的自定义判断标准,以进一步灵活运用该技术.

致谢 感谢审稿专家提出的宝贵意见和建议.

References:

[1] Rozinat A, van der Aalst WMP. Conformance checking of process based on monitoring real behavior. Information Systems, 2008, 33(1):64−95. [doi: 10.1016/j.is.2007.07.001]

[2] Van der Aalst WMP, Adriansyah A, van Dongen B. Replaying history on process models for conformance checking and performance analysis. WIREs Data Mining and Knowledge Discovery, 2012,2(2):182−192. [doi: 10.1002/WIDM.1045]

[3] Van der Aalst WMP, Weijters AJMM. Process mining: A research agenda. Computers in Industry, 2004,53(3):231−244. [doi: 10.1016/j.compind.2003.10.001]

[4] Van der Aalst WMP. Process discovery: Capturing the invisible. Computational Intelligence Magazine, 2010,5(1):28−41. [doi: 10.1109/MCI.2009.935307]

[5] La Rosa M, Reijers HA, van der Aalst WMP, Dijkman RM, Mendling J, Dumas M, Garcia-Banuelos L. APROMORE: An advanced process model repository. Expert Systems with Applications, 2011,38(6):7029−7040. [doi: 10.1016/j.eswa.2010.12.012]

[6] Yan Z, Dijkman R, Grefen P. Fast business process similarity search with feature-based similarity estimation. In: Robert M, Tharam D, Pilar H, eds. Proc. of the Move to Meaningful Internet Systems: OTM 2010. Berlin, Heidelberg: Springer-Verlag, 2010. 60−77. [doi: 10.1007/978-3-642-16934-2_8]

[7] Murata T. Petri nets: Properties, analysis and applications. Proc. of the IEEE, 1989,77(4):541−580. [doi: 10.1109/5.24143]

[8] Van der Aalst WMP, Weijters AJMM, Maruster L. Workflow mining: Discovering process models from Event Logs. IEEE Trans. on Knowledge and Data Engineering, 2004,16(9):1128−1142. [doi: 10.1109/TKDE.2004.47]

[9] Van der Aalst WMP, Dumas M, Ouyang C, Rozinat A, Verbeek E. Conformance checking of service behavior. ACM Trans. on Internet Technology, 2008,8(3). [doi: 10.1145/1361186.1361189]

[10] Van der Aalst WMP, Rubin V, Verbeek HMW, van Dongen BF, Kindler E, Günther CW. Process mining: A two-step approach to balance between underfitting and overfitting. Software & Systems Modeling, 2010,9(1):87−111. [doi: 10.1007/s10270-008-0106-z]

[11] Wang JM, Wen LJ. Discovering process knowledge from Event Logs. Communications of the CCF, 2012,8(6):63−68 (in Chinese with English abstract).

[12] Li J, Wen LJ, Wang JM. Process model storage mechanism based on Petri net edit distance. Computer Integrated Manufacturing Systems, 2013,19(8):1832−1841 (in Chinese with English abstract).

[13] Wang SH, Wen LJ, Wei DS, Wang JM, Yan ZQ. SSDT matrix-based behavioral similarity algorithm for process models. Computer Integrated Manufacturing Systems, 2013,19(8):1822−1831 (in Chinese with English abstract).

[14] Fan GS, Yu HQ, Chen LQ, Liu DM. Fault diagnosis and handling for service composition based on Petri nets. Ruan Jian Xue Bao/ Journal of Software, 2010,21(2):231−247 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3790.htm [doi:10. 3724/SP.J.1001.2010.03790]

[15] Song M, Wei ZX, Yin GS. Evolution analysis of data flow oriented internet ware service. Ruan Jian Xue Bao/Journal of Software, 2013,24(12):2797−2813 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4396.htm [doi:10.3724/SP.J.1001. 2013.04396]

[16] Wu D, Feng DG, Lian YF, Chen K. Efficiency evaluation model of system security measures in the given vulnerabilities set. Ruan Jian Xue Bao/Journal of Software, 2012,23(7):1880−1898 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/ 4112.htm [doi:10.3724/SP.J.1001.2012.04112]

[17] Yuan CY. Principals and Application of Petri Net. Beijing: Publishing House of Electronics Industry, 2004 (in Chinese).

[18] Van der Aalst WMP, Weijters T, Maruster L. The application of Petri nets to workflow management. Journal of Circuits, Systems and Computers, 1998,8:21−66. [doi: 10.1142/S0218126698000043]

[19] Hull R. Artifact-Centric business process models: Brief survey of research results and challenges. In: Meersman R, Tari Z, eds. Proc. of the Move to Meaningful Internet Systems: OTM 2008. LNCS, Berlin, Heidelberg: Springer-Verlag, 2008. 1152−1163. [doi: 10.1007/978-3-540-88873-4_17]

[20] Xu W, Su JW, Yan ZM, Yang J, Zhang L. An artifact-centric approach to dynamic modification of workflow execution. In: Herrero P, Kumar A, Reichert M, Qing L, Ooi BC, Damiani E, Schmidt DC, White J, Hauswirth M, Hitzler P, Mohania M, eds. Proc. of the Move to Meaningful Internet Systems: OTM 2011. LNCS, Berlin, Heidelberg: Springer-Verlag, 2011. 256−273. [doi: 10.1007/978-3-642-25109-2_17]

[21] Van der Aalst WMP, ter Hofstede AHM. YAWL: Yet another workflow language. Information System, 2005,30(4):245−275. [doi: 10.1016/j.is.2004.02.002]

[22] YAWL foundation. YAWL-Technical Manual (version 2.1). 2010.

[23] Ge JD, Hu H, Lu J. A transformation approach from workflow net to PERT diagram based on invariants. Acta Electronica Sinica, 2008,36(5):893−898 (in Chinese with English abstract).

[24] Ge JD, Hu HY, Zhou Y, Hu H, Wang DY, Guo XB. A decomposition approach with invariant analysis for workflow coordination. Chinese Journal of Computers, 2012,35(10):2169−2181 (in Chinese with English abstract).

附中文参考文献:

[11] 王建民,闻立杰.从日志数据中挖掘过程知识.中国计算机学会通讯,2012,8(6):63−68.

[12] 李婕,闻立杰,王建民.基于 Petri 网编辑距离相似性的过程模型存储机制.计算机集成制造系统,2013,19(8):1832−1841.

[13] 汪抒浩,闻立杰,魏代森,王建民,闫志强.基于任务最短跟随距离矩阵的流程模型行为相似性算法.计算机集成制造系统,2013,19(8):1822−1831.

[14] 范贵生,虞慧群,陈丽琼,刘冬梅.基于 Petri 网的服务组合故障诊断与处理.软件学报,2010,21(2):231−247. http://www.jos.org.cn/1000-9825/3790.htm [doi:10.3724/SP.J.1001.2010.03790]

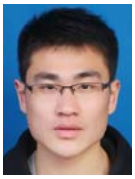
[15] 宋敏,韦正现,印桂生.面向数据流的网构软件服务动态演化分析.软件学报,2013,24(12):2797−2813. http://www.jos.org.cn/1000-9825/4396.htm [doi:10.3724/SP.J.1001.2013.04396]

[16] 吴迪,冯登国,连一峰,陈恺.一种给定脆弱性环境下的安全措施效用评估模型.软件学报,2012,23(7):1880−1898. http://www.jos.org.cn/1000-9825/4112.htm [doi:10.3724/SP.J.1001.2012.04112]

[17] 袁崇义.Petri 网原理与应用.北京:电子工业出版社,2004.

[23] 葛季栋,胡昊,吕建.一种基于不变量的从工作流网到 PERT 图的转换方法.电子学报,2008,36(5):893−898.

[24] 葛季栋,胡海洋,周宇,胡昊,王栋毅,过小波.一种基于不变量的工作流协同模型分解方法.计算机学报,2012,35(10):2169−2181.



李传艺(1991−),男,江苏淮安人,博士生,主要研究领域为工作流技术,过程挖掘.

葛季栋(1978−),男,博士,副教授,CCF 高级会员,主要研究领域为工作流技术,过程挖掘,软件协同,软件过程技术.

胡海洋(1977−),男,博士,教授,主要研究领域为工作流技术,过程挖掘,软件协同,软件体系结构.

胡昊(1975−),男,博士,副教授,主要研究领域为工作流技术,软件协同,软件体系结构,软件过程技术.

骆斌(1967−),男,博士,教授,博士生导师,主要研究领域为软件工程,工作流技术,过程挖掘.