清　华　大　学

攻 读 工 程 硕 士 学 位

研究生选题报告

论文题目　基于完全前缀展开的无序事件日志修复研究

研 究 生　肖永博　学 号　2014213491

院（系、所）软件学院　专　业　软件工程领域

联 系 电 话　15201519152　电 子 邮 件　xiaoyb2010@gmail.com

指 导 教 师　闻立杰　专业技术职务　副教授

联 系 电 话　13681026629　电 子 邮 件　wenlj@tsinghua.edu.cn

联合指导教师　　专业技术职务

联 系 电 话　　电 子 邮 件

入 学 日 期　2014 年 9 月

2015 年 9 月 30 日

目录

1 选题背景

在高科技信息化时代，伴随着信息技术的飞速发展和市场变化的日新月异，过程感知的信息系统（即 Process Aware Information System，简称 PAIS）已经被广泛的应用于各行业，为企业提供业务流程建模与实施能力。商业流程随着其经营内容的发展而变得复杂，这一变化促进了企业对流程模型的研究和使用。大量以分析流程模型为核心功能的系统（如：业务流程管理系统，即 Business Process Management 系统，简称 BPM 系统；业务活动监测系统，即 Business Activity Monitoring 系统，简称 BAM 系统；以及业务流程智能系统，即 Business Process Intelligence 系统，简称 BPI 系统）的涌现印证了流程模型对管理层的重要性。这些业务流程管理系统为企业提供了对流程模型和业务事件日志相配套的管理。在这些应用环境中，模型和日志已不单单是用作记录流程的工具，它们已经成为企业业务流程优化、业务流程监控等环节的重要参与者和有效载体[1]。业务流程模型，简称流程模型，近年来已经成为现代企业不可缺少的重要组成部分，是企业宝贵的知识资产。企业为业务流程建立模型，这些模型可以帮助人们记录保存商业流程、理解分析业务逻辑。结构良好的流程模型已经被广泛应用于办公自动化、企业信息化、电子政务和电子商务等领域。常用模型的类型包括 Imperative 模型（如 EPC、Petri 网、BPMN2.0 和 YAWL）、Declarative 模型（如 ConDec）、Hybrid 模型等。系统日志是过程分析、过程挖掘与过程改进研究的起点。目前，所有的系统日志均可转化为事件日志（即 Event Log）。事件日志用于记录业务流程的实际执行情况，其在企业信息系统中的作用体现在两个方面：1. 记录企业的内部业务流程执行情况；2. 反映企业各部门的运营状况。

业务流程的不断丰富变化使其结构也变得越来越复杂，企业的运营管理也越来越依赖于业务流程方面的支持。然而，实际的业务操作经常性地偏离于企业的业务模型，例如业务模型中的某些活动在事件日志中被遗漏而没有被记录下来、事件日志中记录了业务模型中没有描述到的一些活动、事件日志中某些活动的执行顺序不符合业务模型中的描述等。造成上述业务操作同业务模型发生偏离的原因可能来自两方面：1. 业务流程本身发生变化造成了业务模型的过时，即业务模型已经无法刻画实际的业务需求；2. 在业务的实际操作过程中存在人为或系统的记录错误。流程管理为了减小模型和日志之间的偏离程度，首先需要对它们进行符合性检测来获知它们的偏离程度，然后需要采取一定的技术手段对模型或者日志进行相应的修复。

业务流程分析技术往往依赖于流程模型，并且认定实际执行的业务过程应该符合于流程模型。因此对于流程管理而言，检测实际的业务流程在多大程度上符合流程模型的规范是非常重要的。符合性检测是流程挖掘中用来检查系统日志记载的实际行为与设计的流程模型行为是否相符的常用办法，为检测实际行为同模型规范之间的偏离程度提供了方法（见图 1-1）。符合性检测的主要方法是：在流程模型上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，进而判断模型与日志的符合程度[19]。因此，事件日志的正确性对符合性检测，以及后续的流程分析、挖掘与改进来说是非常重要的。

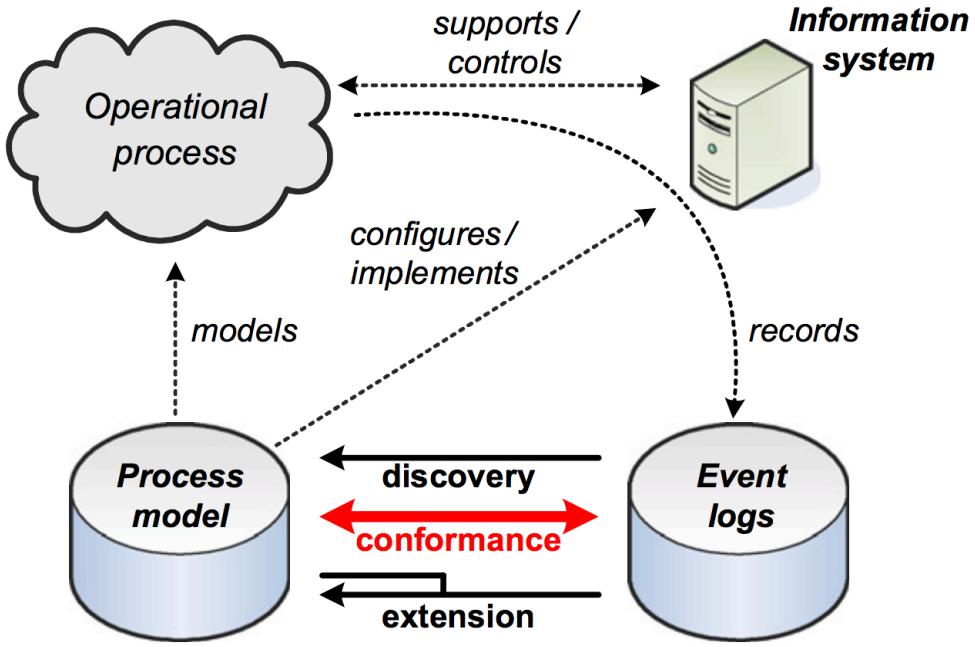


图 1-1 符合性检测同流程发现、流程增加之间的关系

目前已经有很多学者对流程日志与模型的修复问题展开了研究。针对修复对象而言，已有的技术可以分为对模型控制流进行修复、对日志进行修复、对模型任务标签进行修复。对模型控制流、模型任务标签进行修复可以让模型重新变得可靠，使其更高效、准确地刻画业务流程。对事件日志进行修复，即将发生偏离的日志修正为符合模型要求的日志，使这部分日志中的有效信息得以保留、错误信息加以纠正、缺失信息加以补充。针对日志修复维度而言，修复方法又可以分为针对控制流进行的修复和对包含数据、资源、时间等信息的控制流进行修复。

现有的日志模型修复方法或是基于已有日志和模型进行修复或是基于日志中的结构化信息进行修复，然而当日志中的可用信息非常少（例如活动顺序全部丢失，即日志中事件的发生时间缺失），传统的日志修复方法就无法快速有效地对日志进行修复。

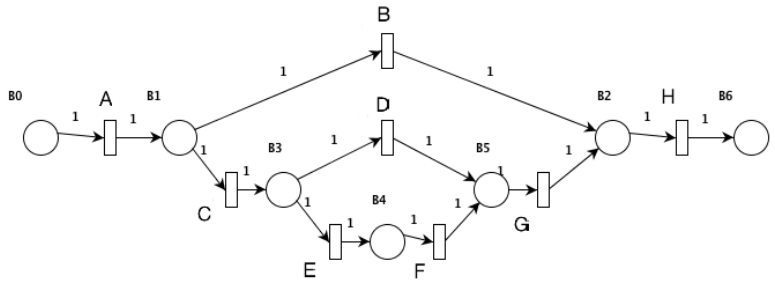


图 1-2 Petri 网模型 I

给定流程模型如图 1-2 所示，现有的事件日志仅记录了活动名称：{A, A, H}，活动的执行顺序等其他信息全部缺失。在这种情况下对日志进行修复可形成有效的轨迹有<A, B, H>、<A, C, D, G, H>和<A, C, E, F, G, H>。假定对活动多集中的

任意活动而言，将其插入当前轨迹的代价是 1，对只在给定模型中出现但不出现在活动多集中的活动而言，将其插入当前轨迹的代价是 2，那么从原日志中得到第一条轨迹的代价是 4，得到第二条和第三条轨迹的代价分别是 8 和 10。因此，只有第一条轨迹是经过最小修复代价而得到的。

本课题旨在针对解决上述类型的日志修复问题进行形式化定义、问题的复杂性分析，提出具体的算法解决方案并验证算法的效果和效率。

2 国内外研究现状

在流程挖掘领域里，针对不同模型（例如 ConDec、Petri 网、BPMN 2.0 等）进行符合性监测的研究越来越得到人们的重视。展开技术的发展（例如完全有限前缀等）大大简化了分析复杂模型的工作。同时，与符合性监测相关的日志与模型修复方面的研究也有很多新的成果。下面，本文将从流程模型、符合性检测、展开技术和日志与模型修复四个方面介绍国内外的研究现状和相关工作。

2.1 流程模型

2.1.1 Declarative 模型

Declarative 模型是一种非结构化的模型[2]。并不像 Petri 网那样规定某个事件一定要在另一个事件之前发生，Declarative 模型往往声明什么应该被执行而不是如何被执行[3]。Declarative 模型只规定一些约束规则，即哪些操作是不被允许的或者哪些操作是一定要被满足的。图 2-1 一个典型的 Declarative 模型，可以看到，其中的两个活动“Create Questionnaire”和“Send Questionnaire”是用约束“response”连接起来，这表明，“Create Questionnaire”这个活动执行之后会要求执行“Send Questionnaire”，但是并不是说前者执行之后，后者需要立即执行。这个约束仅仅规定了“Send Questionnaire”需要执行，但没有规定何时执行、执行几次。因此，在“Send Questionnaire”之前，可以执行很多其他的活动。Declarative 模型挖掘旨在针对松散流程产生的日志，挖掘出其中的约束规则，并用一个简单易懂的方式展现给用户。

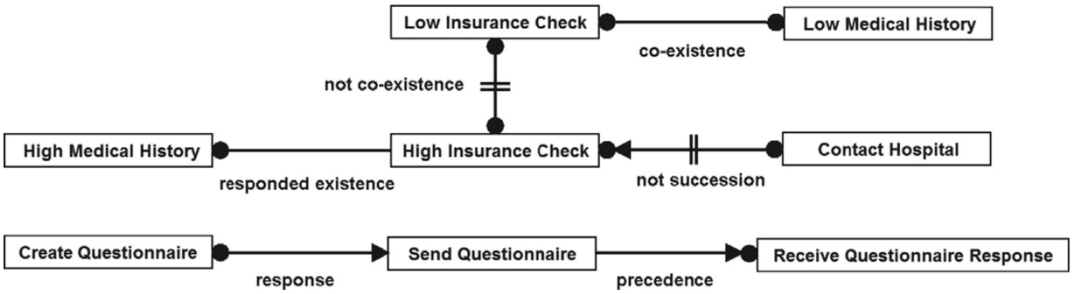


图 2-1 Declarative 模型示例

van der Aalst 提出的 DECLARE 系统是针对工作流管理系统（Workflow

Management System，简称 WFMS）的一个补充，使其能够支持 Declarative 模型[2]。目前最常用的 Declarative 模型建模语言是由 Pesic 提出的 ConDec[3] 。对于 Declarative 模型而言，最常见的形式化方法就是 LTL 逻辑。图 2-2 展示了一部分 Declarative 模板的图形化表示和对应的 LTL 形式化表示方法。使用这样的语法可以将业务流程表示为 Declarative 模型。

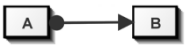
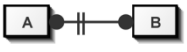
Constraint	LTL semantics	Graphical representation
responded existence	$\Diamond A \rightarrow \Diamond B$	
co-existence	$\Diamond A \leftrightarrow \Diamond B$	
response	$\Box(A \rightarrow \Diamond B)$	
precedence	$\neg B \mathcal{W} A$	
alternate response	$\Box(A \rightarrow \bigcirc(\neg A \sqcup B))$	
alternate precedence	$(\neg B \mathcal{W} A) \wedge \Box(B \rightarrow \bigcirc(\neg B \mathcal{W} A))$	
chain response	$\Box(A \rightarrow \bigcirc B)$	
chain precedence	$\Box(\bigcirc B \rightarrow A)$	
not co-existence	$\Diamond A \rightarrow \neg \Diamond B$	
not succession	$\Box(A \rightarrow \neg \Diamond B)$	
not chain succession	$\Box(A \rightarrow \neg \bigcirc B)$	

图 2-2 Declarative 模型的形式化展示及其对应的 LTL 形式化表示方法

2.1.2 Imperative 模型

与 Declarative 类型相对的建模语言是 Imperative 类型，包括 EPC、Petri 网、BPMN 2.0 及 YAWL 在内的诸多工作流建模语言都是 Imperative 类型的，它们严格描述流程如何执行。这样的工作流系统允许用户在执行流程时改变模型以使得工作更加符合动态的流程管理[4]。图 2-3 展示了两种类型建模语言的差异，ConDec 模型通过使用规则约束（图 2-3（a）中的阴影区域）来划分哪些行为是禁止的、哪些行为是允许的，而 Imperative 模型则会明确描述流程发生的过程（图 2-3（a）中的箭头）。图 2-3 (b)和图 2-3 (c)分别展现了 ConDec 模型和 Imperative 模型对于“事件 A 和事件 B 不能同时发生”的表达。图 2-3 (b)中，ConDec 仅仅通过一个约束来要求事件 A 和事件 B 不能同时发生，而在图 2-3 (c)中，Imperative 模型则需要通过一个决定活动 X（Xor-split）来表达这一含义。

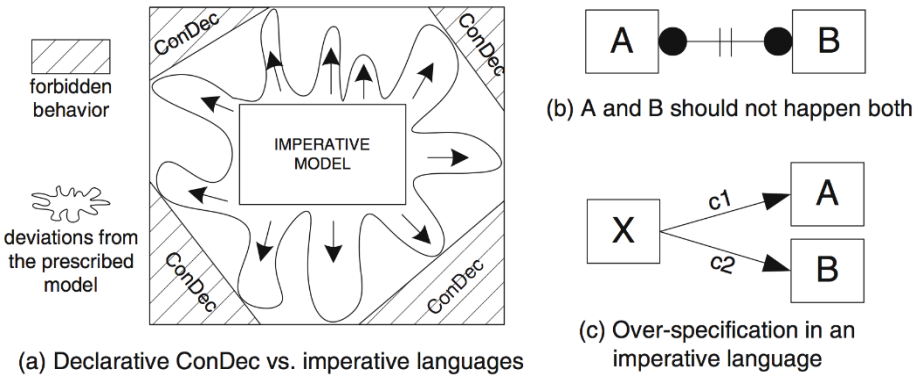


图 2-3 Imperative 模型与 Declarative 模型比较

Petri 网是 20 世纪 60 年代由 Carl Adam Petri 发明的，适合于描述异步的、并发的计算机系统模型。Petri 网既有严格的数学表示方式，也有直观的图形表达方式。由于 Petri 网能够表达并发事件，因此它被认为是自动机理论的一种。研究领域趋向认为 Petri 网是所有流程定义语言之母。其具体定义如下：

定义（Petri 网）：Petri 网是一个三元组（P, T, F），其中 P 和 T 分别是库所(place)和变迁(transition)的集合，且 P 交 T 为空集，F 是一个函数：$(P\times T)\cup(T\times P)\rightarrow\{0,1\}$。库所和变迁统称结点，如果 F(x, y)=1，则 x 到 y 有一条有向弧。任意变迁 t 的输入库所集合记为 $\bullet t$，输出库所集合记为 $t\bullet$；同理任意库所 p 的输入变迁集合记为 $\bullet p$，输出变迁集合记为 $p\bullet$。

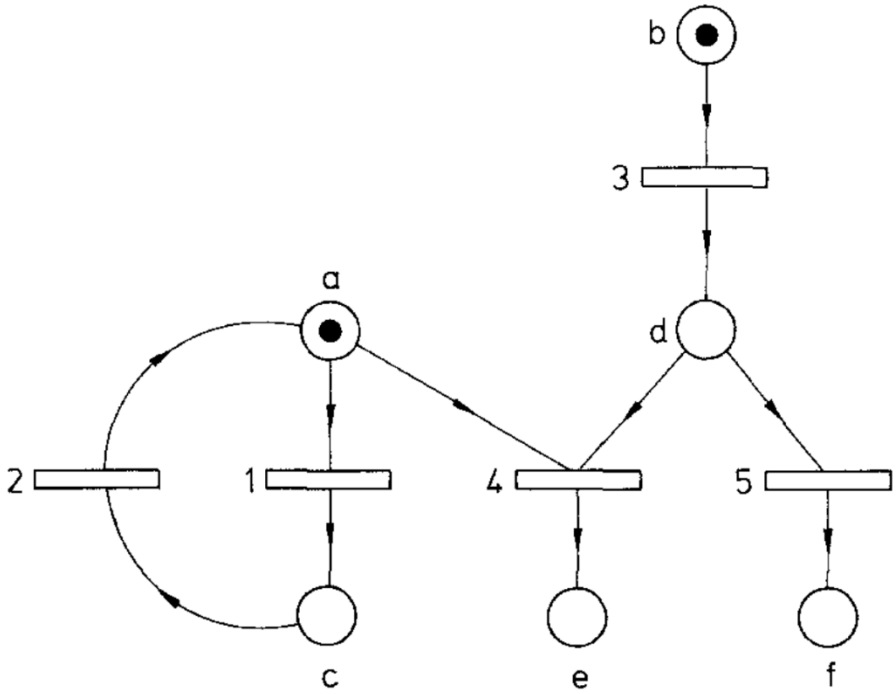


图 2-4 Petri 网模型 II

如图 2-4 中的 Petri 网模型所示，它的库所集合 P 为{a, b, c, d, e, f}，它的变迁集合 T 为{1, 2, 3, 4, 5}，变迁 4 的输入库所集合为{a, d}，变迁 4 的输出库所集

合为{e}。

目前已有很多学者对 Petri 网进行了深入研究，并取得了显著进展。在理论方面，学术界已经为 Petri 网的研究建立了一系列分析技术，包括代数分析技术[6]、图分析技术[7]、约简分析技术、展开分析技术等。

代数分析技术最先是由 Peterson 在[6]中提出的，之后 Murata 对该工作做了更详细的扩展[8]。代数分析技术的主要方法是用关联矩阵的形式对一个网系统的结构进行刻画，然后建立状态可达的线性系统关系。它的优点在于可以利用线性代数简洁地展现 Petri 网的一些性质。而这种方法的缺点则是它难于很好地刻画 Petri 网的动态特征。

图分析技术类似于状态机的分析，是以一个有限的有向树直接地展现一个网系统的运行机制。这一思想最先由 Karp 和 Miller 在[7]中提出。图分析技术的优点是能够反映一个网系统的动态行为和特征。虽然可达树可以很好地对 Petri 网进行分析，但其状态的复杂性通常是令人难以接受的。为此，人们提出可达树化简等方法来克服这一困难，在一定程度上起到了作用，但还有待进一步研究。

约简分析技术是针对 Petri 网的状态复杂性而提出的。一个规模不大的系统，可能会出现状态组合爆炸的危险，从而给分析带来困难。对此，人们提出化简和分解的思想。化简是在保留一些性质不变的基础上，将一个较复杂的 Petri 网简化成一个比较简单的 Petri 网的同态变换过程，这个过程减小了可达状态空间，为分析原网提供充分的信息[9]。

展开技术是针对 Petri 网的空间搜索问题而提出的研究方法。当业务流程模型中包含循环结构的时候，业务流程模型的行为空间会剧烈增长，产生爆炸[10]。1993 年，McMillan 提出了一种新的 Petri 网的空间搜索方法[11]，他通过构造 Petri 网的完全有限前缀（Complete Finite Prefix, 简称 CFP）来压缩业务流程模型，使其仍能表达原有模型所能表达的全部行为状态。Esparza 等人则在此基础上对完全有限前缀技术进行了改进，使得在最坏情况下依然能够产生一个有限的前缀，进一步缩小了完全有限前缀展开后的大小[12]。

建立在通用网论基础之上的并发行为的特性研究是 Petri 网理论研究的另一主要方向。在最为著名的并发公理系统[13]基础上，许多研究者开展了并发语义的刻画系统以及并发系统的构造等研究，并且取得了很多成果。

2.1.3 Hybrid 模型

轻结构而重多样性的模型适合用 Declarative 语言来表示，而重结构且稳定的模型则适合用 Imperative 语言来表示。在很多场景中，一个流程可以既包含适合用 Declarative 语言表示的部分也包含适合用 Imperative 语言表示的部分。为了能够有效表示这类模型，Maggi 等人提出了 Hybrid 模型[5]，这类模型的每个子流程都是 Imperative 模型或者 Declarative 模型。

图 2-5 是一个 Hybrid 模型示例。该类模型将 Declarative 模型与 Imperative 模型用子流程表示并连接各个子流程从而描述混杂类型的业务流程。其中，P1.1、P1.2、P1.3、P1.4、P2.1、P2.2 是 Petri 网模型，D1.1、D2.1 是 ConDec 模型。

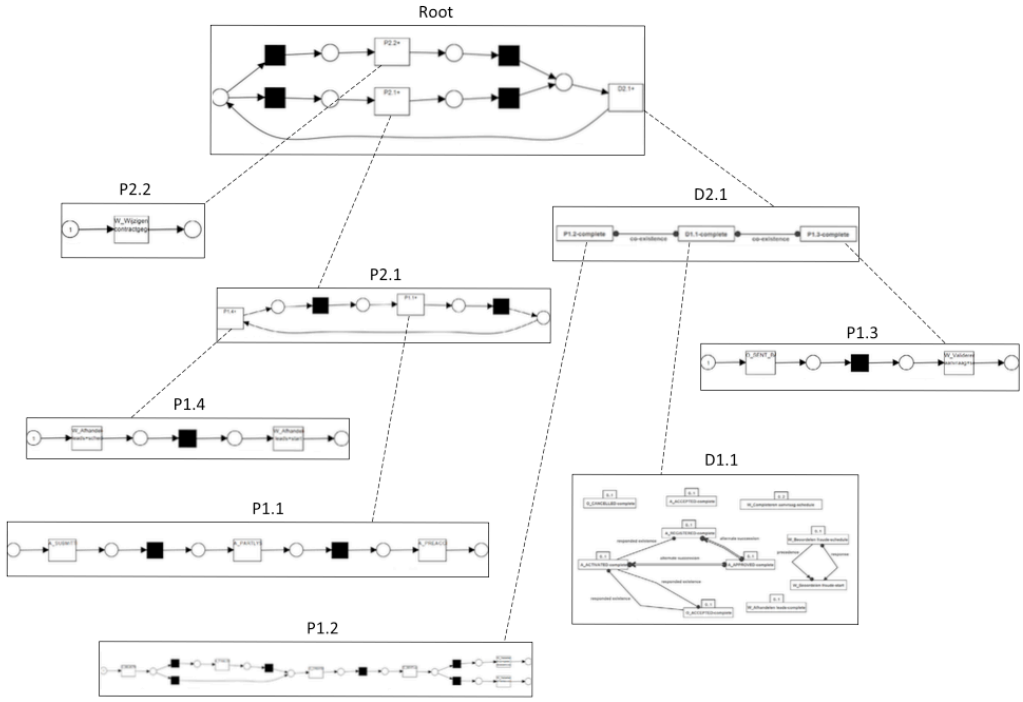


图 2-5 Hybrid 模型

2.2 符合性检测

流程发现（Process Discovery）致力于自动地从事件日志数据中抽象出流程模型，而符合性检测（Conformance Checking）则关注于已有流程模型同其对应的事件日志之间的关系。

符合性检测，也被称为符合性分析，是用来检测流程模型同其相对应的事件日志之间的一致性，并由此来评价流程模型同流程日志之间的关系[15]（见图 1-1）。符合性检测既可以用来比较不同流程发现算法挖掘出来的流程模型之间的优劣，也可以用来评价已有流程模型能否支持实际的业务操作。流程模型在企业业务管理过程中有着重要作用，然而流程模型经常性地偏离于实际业务。因此对于流程管理而言，检测实际的业务流程在多大程度上符合业务流程模型的规范是非常重要的。

2.2.1 度量维度

Rozinat 等人在[16]中提出了流程模型与事件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流程模型再现的程度；

(2) Appropriateness 用以衡量流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 可以通过分析流程模型、模拟执行日志时的状态变化而得到：

(1) 计算 Fitness 时，需要统计每个任务在被执行时，消耗令牌 (Token)、产生令牌、残留令牌和缺失令牌的数目；

(2) 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。

上述提到的 Appropriateness 可以被扩展成 Precision、Generalization 和 Structureness 这三个指标来分别进行衡量[21]：

(1) Precision 表达了模型刻画日志的精确程度。图 2-6(d)中的模型可以允许活动 A-I 以任意的顺序执行，因此这个模型有很好的 Fitness 但是它的 Precision 却很低。

(2) Generalization 表达了模型的平凡程度。如果一个模型仅仅能够表达日志中发现的行为而不能表达的更多，那么这个模型的 Generalization 就会较低。

(3) Structureness 依据建模语言来描述流程模型结构复杂度。如果一个模型有很多重复任务或者有很多不必要的 and/xor 结构，同时这个模型可以在一定程度上得到简化得到一个结构更小的模型，那么这个模型的 Structureness 就会较低。

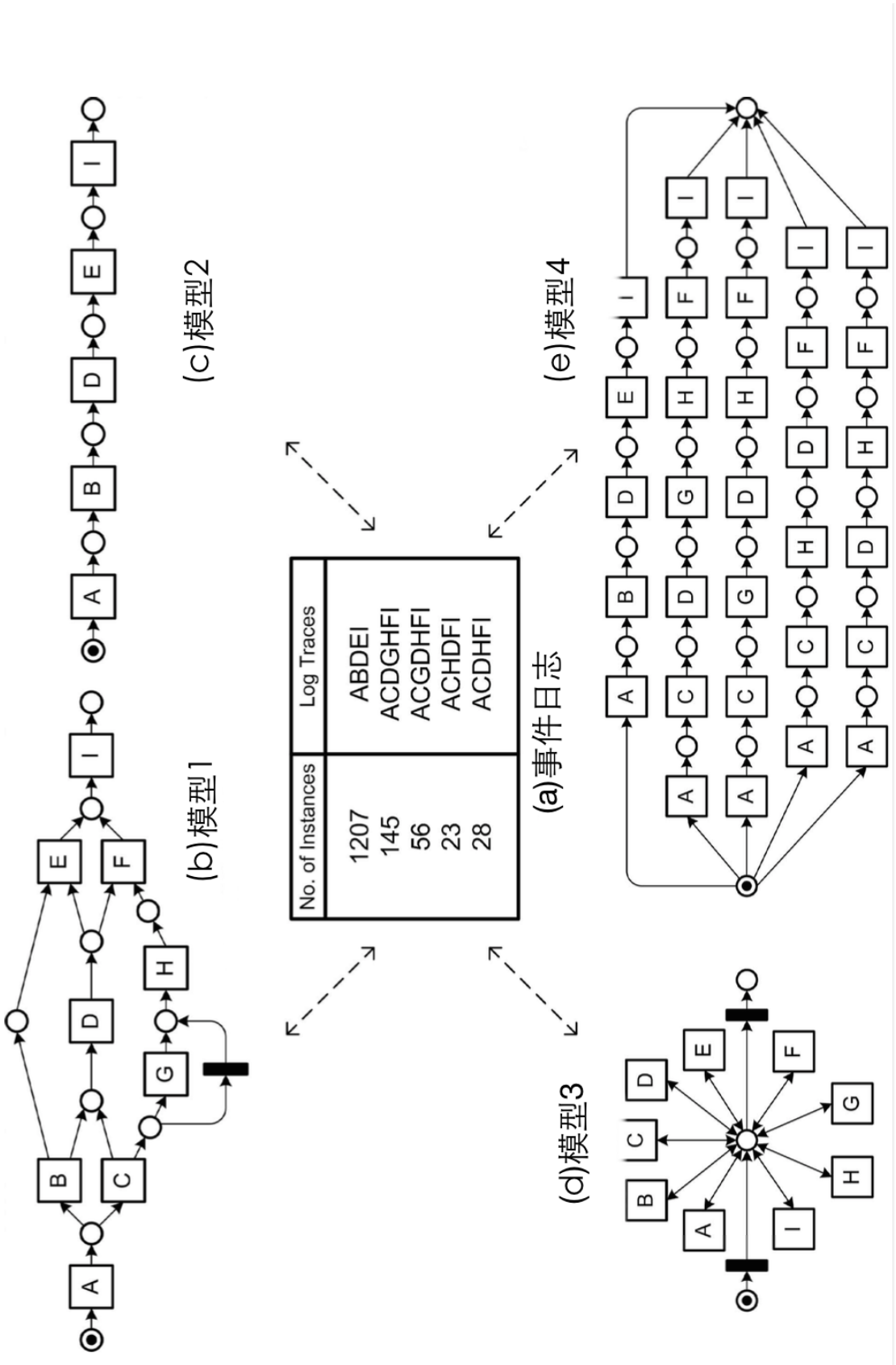


图 2-6 (a)为一个事件日志，(b)-(d)为四个能够产生事件日志(a)的模型

图 2-6 中展示了一个事件日志(a)以及四个不同的流程模型(b)-(e)。相较于模型(c)、(d)、(e)，模型(b)是对日志(a)的一个好的模型描述。对于模型(d)，它能够

再现活动 A-I 以任意顺序进行排列而构成的轨迹，因此模型(d)的 Fitness 是很好的，但是它的 Precision 却很差。对于模型(c)，它严格地仅能产生出<ABDEI>一种轨迹，而对于日志(a)中的其他序列，模型(c)无法再现它们。类似地，对于模型(e)，它严格地仅能产生出日志(a)中的五个轨迹。因此这两个模型的 Generalization 都很差。对于模型(e)，虽然它有很好的 Fitness 和 Precision，但是它的表达相较于其他三个模型来说过于复杂，因此它的 Structureness 很低。

在进行符合性检测时，实际上是需要上述几种维度一起作为衡量指标来评价模型的好坏的。然而近年来，人们对模型质量的评价往往是仅基于 Fitness 这一维度的。对此，Adriansyah 等人和 Weerdt 等人分别在[17]和[20]中，阐释了 Precision 和 Generalization 这两个维度对于符合性检测的重要性和相应的检测方法。

与传统的计算 Precision 方法相不同[18] [19]，[17]并没有直接用日志中的轨迹和模型来做符合性检测。Adriansyah 等人认为，传统的对 Precision 的计算方法忽略了日志中可能存在的不可靠性，因此这些方法计算出的 Precision 也会不可靠。Adriansyah 等人在[17]中首先用对齐的方法来对日志中的轨迹进行了修复，这将提高计算的可信性。然后，该方法将通过构建自动检测方法来对已修复的日志和模型进行 Precision 计算。

Weerdt 等人在[20]中通过对日志使用加权人工负事件生成算法，统计日志中的真的正事件（true positive events）、假的正事件（false positive events）、允许出现的泛化事件（allowed generalizations）和不允许出现的泛化事件（disallowed generalizations）数量，最终计算出 Precision 和 Generalization。

2.2.2 符合性检测方法

符合性检测可以用来测量事件日志同一个模型系统之间的符合程度或者是偏离程度，常用的方法可以分为日志重演和日志模型对齐[14]。

日志重演即测试日志中的每条轨迹是否能用模型重新生成出来。在 Petri 网中，如果一条轨迹能完整的从开始运行到结束，中间没有任何多余或者缺失的令牌，则该轨迹被称为完整发生序列。可以从头到尾遍历该条轨迹，每处理一个事件，查看该事件是否被使能。若是的话，则继续处理该事件，并且记录新的使能事件；若不是的话，则在该处有事件缺失,记录缺失并继续处理轨迹中接下来的事件。

日志模型对齐（Alignment）可以用来表示日志同模型发生偏离的位置[15]。通过比较不同的对齐方法，本文能够获得一种可以减少（或者是消除）日志同模型之间的偏离的最小操作集合。

日志模型对齐同拉文斯基距离等编辑距离（Edit Distance）有所不同[33]。首先，对齐操作是发生在轨迹和模型之间的，而非发生在轨迹之间的。模型可能会产生出很多种不同的轨迹，因此本文不可能通过列举所有模型可以产生的轨迹来寻找同已知轨迹偏离最小的那一个轨迹。其次，对齐操作的过程中，不同的偏离往往需要对应不同的代价。最后，任何对齐操作都不仅仅关注于轨迹和模型之间的距离，它们更关注的是在哪里发生了这些偏离。

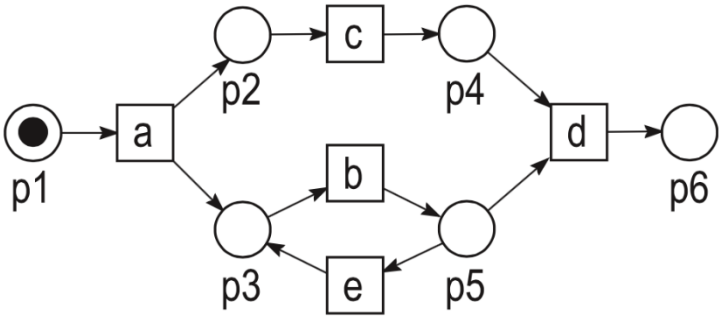


图 2-7 Petri 网模型 III

考虑图 2-7 中的模型 III 及一条轨迹 L= accd。模型 III 中的一条发生序列为 σ = abcd。对于 L 和 σ，它们之间的一个对齐方法如下图所示：

$$\begin{array}{|c|c|c|c|c|}\hline a & c & \gg & c & d \\ \hline a & c & b & \gg & d \\ \hline\end{array}$$

图 2-8 轨迹 L=accd 同 Petri 网模型 II 的一个对齐

图 2-8 中的第一行是轨迹 L，第二行是模型 II 的发生序列σ。其中第一行的>>表示轨迹 L 跳过了活动 b，第二行的>>表示σ跳过了活动 c。

定义（对齐）：对于一个有标签的模型 N(P, T, F, l, M)。对于一个轨迹和模型 N 的来说，它们之间的一个对齐就是一个序列 α 。

对于图 2-8 来说，轨迹 L=accd 的一个对齐就是(a, a)(c, c)(>>, b)(c, >>)(d, d)。

对每一条轨迹而言，都会有很多种对齐方法使它和模型 N 对齐。例如说对于图 2-7 中的模型 II 和轨迹 L= abcbd，图 2-9 中的三种对齐方法都是可以使 L 和模型 N 对齐的。

$$\alpha_1 = \begin{array}{|c|c|c|c|c|}\hline a & b & c & b & d \\ \hline a & b & c & \gg & d \\ \hline\end{array} \quad \alpha_2 = \begin{array}{|c|c|c|c|c|c|}\hline a & b & c & \gg & b & d \\ \hline a & b & c & e & b & d \\ \hline\end{array}$$

$$\alpha_3 = \begin{array}{|c|c|c|c|c|c|c|}\hline a & b & c & b & \gg & \gg & d \\ \hline a & \gg & \gg & \gg & c & b & d \\ \hline\end{array}$$

图 2-9 轨迹 L=accd 同 Petri 网模型 III 的三种对齐

在这[1]、[15]、[16]三篇文章中，它们都分别基于对齐概念和代价函数介绍了如何通过最小代价有效来将日志和模型进行对齐的方法。

2.3 展开技术

2.3.1 分支流程

分支流程是一种基于偏序顺序语法的 Petri 网结构的展开形式[31]。通过对原始 Petri 网进行变化，可以得到它的分支流程。分支流程可以大大简化对模型的结构和模型规定行为的分析难度。

在业务模型中，结构最简单的就是 Causal 网。

定义（Causal 网）：Causal 网是一个网络 N = (P,T,F), 需要满足对于$\forall p \in P$，都有 $|\bullet p| \leqslant 1$ 且$|p\bullet| \leqslant 1$。

不同于 Causal 网，带有选择分支的网络将会产生很多不同分支的流程操作，这对分析 Petri 网模型造成了困难。[31]提出了一种简化模型的方法来解决这类问题。首先，需要对所有的选择结构进行分支，使得每一条分支都是 Causal 网而不再含有选择分支结构，每一个分支都可以看作是一个独立的发生网。

定义（发生网）：发生网是一个三元组 O =（B，E，F），需要满足对于任意属于 B 的 b 来说，都有$|\bullet b| \leqslant 1$。

由分支得到的发生网之间会有同态关系[23]，该关系由同态函数来定义。

定义（同态函数）：一个从网 N 到网N'的同态函数是一个映射函数 π：$P \cup T \rightarrow P' \cup T'$。该函数需要满足一下两点：

1. P 包含$\pi(P')$ 且 T 包含$\pi(T')$；
2. 对于$\forall t \in T'$，$\pi(\bullet t)$是$\bullet t$ 和 $t\bullet$ 之间的双射，$\pi(t\bullet)$是$t\bullet$ 和$\pi(t)\bullet$之间的双射。

经过[29]中方法的处理，可以得到一个 Petri 网的流程分支。

(N',π) 是网络 N (P, T, F) 的流程分支，那么它将满足以下三点：

1. N'是一个发生网；
2. π是一个N和N'之间的同态函数；
3. 对于$\forall t_1, t_2 \in T'$，如果$\bullet t_1 = \bullet t_2$ 且$\pi(t_1) = \pi(t_2)$，那么$t_1 = t_2$。

对于图 2-10(a)中的网络 N，它对应的流程分支即为图 2-10(b)所示。图 2-10(b)中的网络都是仅有选择分支没有选择合并的发生网。根据同态函数，分支中的每个节点都可以映射到原网络中，例如：在图 2-10(b)中，有两个库所可以被映射到b_{end}，即$p_4: b_{end}$，$p_5: b_{end}$。

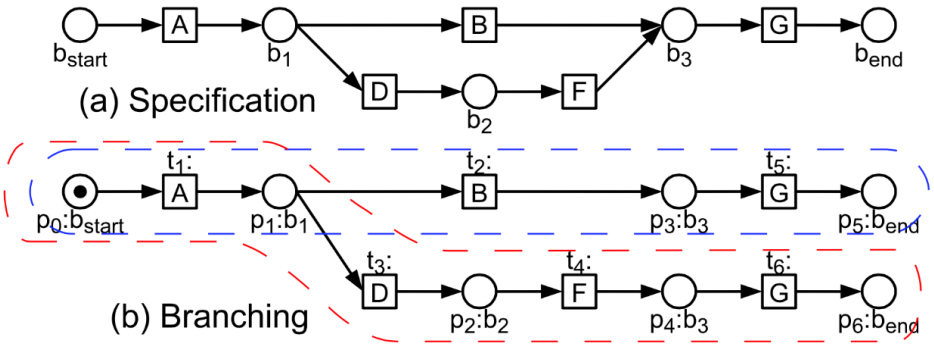


图 2-10 流程分支示例

2.3.2 完全有限前缀

原始 Petri 网描述的是结构信息，并非行为信息，即图结构上存在通路的在实际执行时并不一定存在通路，而完全有限前缀可以用于刻画行为特性。因此本文需要对原始 Petri 网进行处理，得到其对应的完全有限前缀。

定义（序关系）：N=(P, T, F)是一个出现网， x、y∈P∪T 是 N 中的两个结点：

- x 和 y 存在因果关系，如果 N 中存在一条从 x 到 y 的通路，记为 x < y；
- x 和 y 是互斥关系，如果 N 中存在两条路径各自通向 x、y，并且这两条路径开始于一个共同的库所,记为 x # y；
- 如果 x 和 y 既不是因果关系,也不是互斥关系,则和 y 是并发关系，记为 x *co* y。

定义（完全有限前缀）：一个 Petri 网系统可以展开成一个出现网,而完全有限前缀则是展开后的出现网的前缀部分，包含了出现网中的所有状态，并且在即将出现重复状态的位置截断，其中：

- [t]={x| x∈T, x→t}∪{t}；
- Mark([t])表示[t]中所有变迁执行后的状态；
- <是一个偏序关系，如果[t]⊂ [t']则[t]< [t']；
- 变迁 t 是一个截断变迁，如果存在另一个变迁t'满足[t']<「t」并且 Mark([t])=Mark([t'])， t'即为 t 的对应变迁；
- 完全有限前缀是出现网的子图，在截断变迁之后不再包含任何后续变迁并且包含了出现网的所有状态。

MacMillan 在[11]中的研究可以通过对网络的进行展开，并构建出一种初始状态有限、包含所有可达信息的 Petri 网模型的子结构（即完全有限前缀），避免了模型的状态爆炸问题。然而在某些情况下，该方法产生的最小完全前缀会比 Petri 网本身还要大。为了解决这个问题，Javier Esparaza 在[12]提出了一种对 MacMillan 展开方法的改进办法，使得展开得到的最小完全前缀始终小于 Petri 网原有的状态空间。

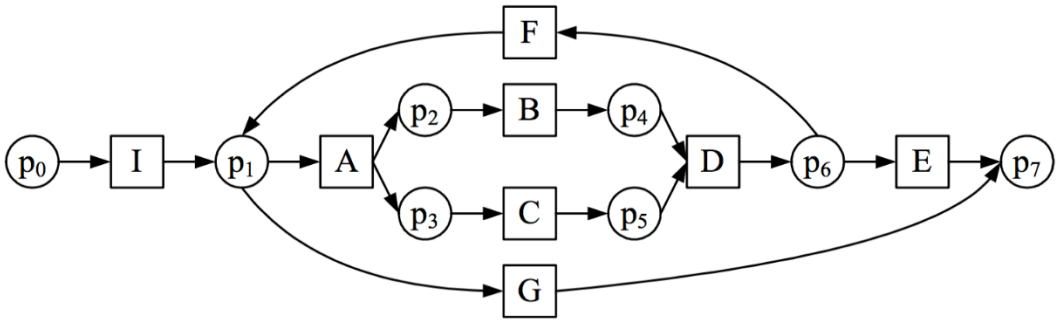


图 2-11 Petri 网模型 III

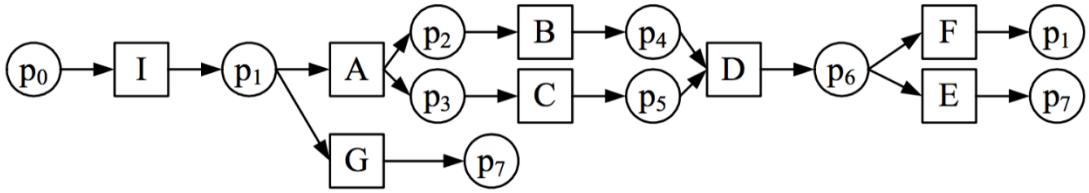


图 2-12 Petri 网模型 III 的完全有限前缀展开

图 2-12 给出了一个图 2-11 中的 Petri 网模型的完全有限前缀。其中，E 和 F 是阶段变迁，G 和 I 分别是它们的对应变迁。

2.4 日志与模型修复

日志与模型修复是指通过改变日志中的记录或者改变模型的结构使得日志中记录的操作行为能够符合模型中的定义。

2.4.1 控制流维度的日志与模型修复

模型修复

Dirk Fahland 等人在[24]中提出，给定一个 Petri 网模型 N 和一个日志 L，如果 L 中所记录的所有执行信息都符合 N 中的定义，那么模型 N 就是符合与日志 L 的，即模型 N 可以重现日志 L；如果模型 N 不能重现日志 L，那么就需要在模型 N 的基础上改变模型的结构使之成为能够符合于日志 L 的新模型N'，同时保证原有模型 N 和新的模型N'尽可能的相似。[24]中的具体方法如下：首先在日志 L 中，根据日志与模型发生偏离的位置将这些发生偏离的日志组成 L 的子日志L'；然后利用流程发现算法对每一个这样的子日志集找到相应的模型 N'；最后将这些模型 N'插入到原有模型 N 中发生相应偏离的地方。通过这样的方法，就可以获得一个保留了原有模型结构且能够重现所有日志情况的修复模型。该方法综合利用了符合性检测和流程发现方法，在尽可能保留原有模型结构的基础上，获得了 Fitness 更高的新模型。然而，该方法在符合性检测上仅仅以 Fitness 为度量标准，没有考虑 Generalization 和 Precision 等度量，且不能确保对模型的修改是最小化的。同时该方法也仅关注于控制流维度的模型修复，没有扩展到对数据和资源维度的模型修复。

日志修复

在统计学领域，信息缺失已经作为一个问题被广为研究。然而在业务流程领域里，相关问题的统计学方法却发展较慢。Andreas Rogge-Solti 等人针对业务流程提出了一种基于概率的活动预测算法[25]。该算法首先使用对齐方法来实现对日志的结构修复，然后再使用贝叶斯网络对流程模型中的变迁执行时间进行概率预测，最终完成对带有时间戳的日志的修复。

[28]中提出了一种日志与模型对齐的修复方法。对于给定的 Declarative 模型和一条轨迹，算法会构造两条路径，一条路径代表轨迹中的业务执行轨迹，另一条路径代表模型中的路径，如图所示。算法用>>表示略过该项活动，用 √ 表示该活动不属于关注范围内的活动。图 2-13 中表示模型中略过了活动 LIC，日志中略过了活动 SQ。算法通过为不同的对齐方法定义不同的代价函数，得出使得日志和模型对齐所需要付出的总代价，最终获得代价最小的修复方法。[28]使用了 A*算法来寻找全局代价最小的对齐方式，保证了算法的正确性。该方法支持事件冗余、事件缺失和事件乱序等全部可能的轨迹出错情况。

$$\gamma_1 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline \mathbf{L:} & \checkmark & LIC & CQ & \checkmark & CQ & \gg & \checkmark & \checkmark & \checkmark \\ \hline \mathbf{M:} & \checkmark & \gg & CQ & \checkmark & CQ & SQ & \checkmark & \checkmark & \checkmark \\ \hline \end{array}$$

图 2-13 对齐修复的路径构造

针对活动缺失的日志修复，[26]提供了一种加速修复的方法。首先，通过比较两个活动的后继节点结合和前驱节点集合，该算法可以判断这两个活动之间是否有缺失的其他活动。图 2-14 表示了一个铁路工厂工程图的生产过程，图 2-15 展现了记录该生产过程的三条轨迹。对于序列<A, B, C, E, G>来说，<A, B, C>是一个发生序列，其后继节点<A, B, C> •={b3, b4}。对于活动 E，其前驱节点为{b4, b5}，由于 E 的前驱集合并不属于序列<A, B, C>的后继集合，因此在序列<A, B, C>和活动 E 之间存在至少一个活动缺失。通过定义 gap 函数，该算法可以为中间存在活动缺失的两个活动填补所缺的活动序列。最后，通过综合利用 branching 索引、可达索引和剪枝技术，该算法在性能上得到了保证, 与基于对齐的方法相比，缺失事件修复方法的效率提升了 3-5 个数量级。

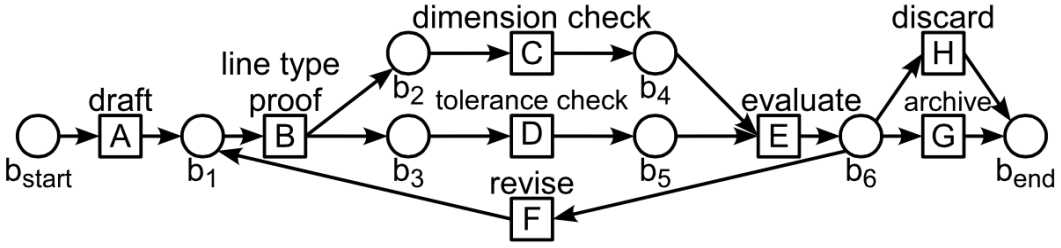


图 2-14 铁路工厂工程图生产过程的流程模型

ID	Sequence of events
1	<A, B, C, D, E, G>
2	<A, B, C, E, G>
3	<A, B, C, D, G>

图 2-15 铁路工厂工程图生产过程的部分日志

标签修复

现有的大多数日志修复方法都将日志视为一种序列信息而忽视了日志中存在的结构化信息，而对日志中的不健全结构化信息进行分析将有助于对日志数据的修复（不健全的结构化信息是指不能和模型中的描述相对应的结构化信息）。不同于从非结构化序列角度对日志进行修复的方法，在[27]中，王建民教授等第一次提出了利用日志结构化信息对日志中活动标签进行修复的方法。[27]中提出的 PTIME one-pass 算法从分析日志中的结构化信息入手，综合利用边界判断技术和剪枝技术，达到了快速有效地为执行序列的修复提供最佳解决方案的目的。针对结构较为丰富的事件日志，该算法的实验结果对比传统的序列修复有较大优势；而对于结构简略或是结构化信息不可靠的日志，该算法的修复能力就会受到限制。

2.4.2 面向多维度的日志修复

虽然学界和业界已有多种日志修复的方法，但是它们中的大多数都只关注于控制流上，即活动的执行顺序。这些方法没有考虑流程除控制流外的其他维度，例如说每个活动的分配资源、访问数据和活动执行时间。例如说当一个人去执行了不应该他去执行的活动的情况下，从控制流的角度来看，这个活动的执行符合了模型的规定，但是实际上业务的操作已经同流程模型发生了偏离。

对于如图 2-16 中模型，传统的符合性检测只会考虑控制流而忽略每个活动的其他属性。因此，对于轨迹：⟨(a,{A=3000, R=Michael, E=Pete}), (b,{V=OK, E=Sue, A=3000, R=Michael}), (c,{I=530, D=OK, E=Sue, A=3000, R=Michael}), (f,{E=Pete, A=3000, R=Michael})⟩，传统的符合性检测将认为它是有效的。

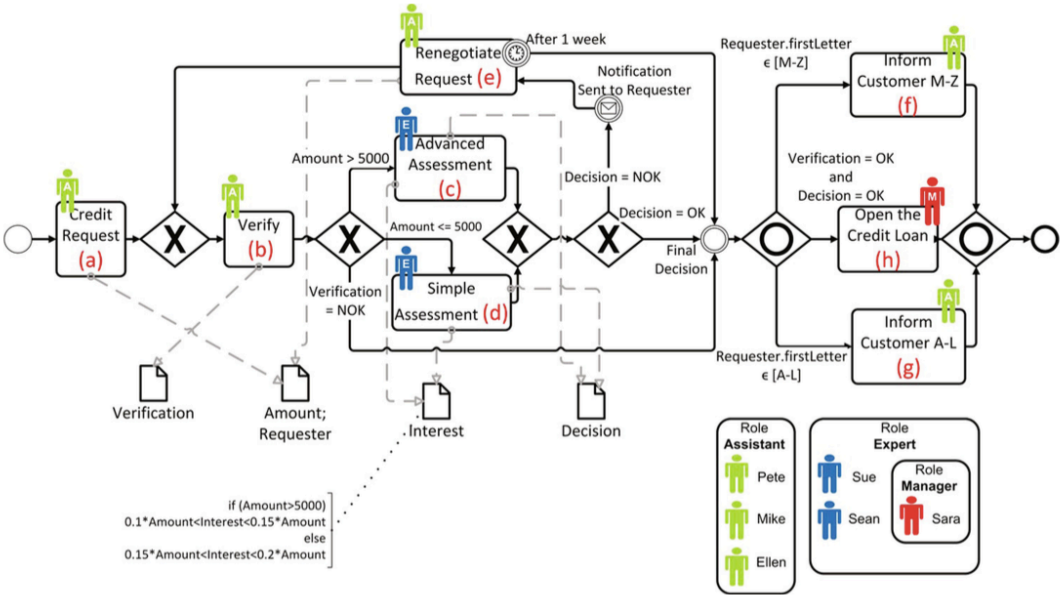


图 2-16 信用借贷流程模型

[29]提出的符合性检测方法不仅会考虑控制流维度，也会考虑活动的数据和资源维度。[29]中的方法是在[28]方法上的扩展，即将仅考虑控制流的对齐方法扩展到了同时检测数据和资源维度的对齐方法。该方法同样设计了 A*算法，保证最终能够得到全局代价最小的对齐方法。

对于上文所说的例子，[29]中的方法将会从以下 3 个角度来说明该轨迹的无效：

(1) 由于贷款额度小于 5000，因此活动 c 不应该被执行；

(2) 对于信用贷款，它的利率不应该多于 450 欧元，因此为它匹配 530 欧元的利率是有悖模型规则的；

(3) 活动 b 不应该由 Sue 来执行，因为他不具有 Assistant 的身份；

(4) 活动 h 没有被执行，因此最终的决定不会是“同意”。

Taghiabadi 等人在[30]中也针对模型中的数据、资源维度的符合性检测问题进行了研究。[30]通过对日志模型发生偏离原因的分析，分别从控制流和数据资源这两个维度上对日志进行修复。该方法在最大程度上将发生在控制流上的日志模型偏离和发生在数据资源上的日志模型偏离分开，并最终给出一个综合的诊断结果。

2.4.3 小结

现有的日志修复方法或是基于已有日志和模型进行修复或是基于日志中的结构化信息进行修复，然而当日志中的可用信息非常少（例如活动顺序全部丢失、活动属性无法利用等情况），日志修复问题的复杂性将变复杂，已有的日志修复方法就无法快速地对日志进行修复，例如：

(1) 当输入日志和模型包含活动的执行时间信息时，[25]将通过概率方法对

活动的执行进行预测，而当输入日志缺少执行时间信息时，该方法将退化为基于序列的对齐修复方法。

(2) 当输入日志和模型包含活动之间的结构化信息时，[27]方法可以快速有效地修复日志中发生的错误，而当输入日志缺少结构化信息时，该方法将无法得到问题的修复解。

(3) 当输入给定的轨迹时，[28]中的 A*方法将会针对该轨迹的对齐方案进行枚举，并给出最优解。而当输入轨迹中的活动为乱序且存在活动缺失或活动冗余时（即输入为活动的活动多集），首先需要对活动多集能够组成的轨迹进行枚举，然后再通过[28]中的方法对每个轨迹进行对齐方案的枚举，最终给出最优解。例如对于图 1-1 中的流程模型和事件日志{A, A, C, H}而言，如果使用[28]中的方法，则首先需要枚举所有可能的轨迹，得到如图 2-17 所示的状态空间，然后再对其中的每条轨迹分别使用 A*算法来寻找各自的最小代价修复方案，最后从每条轨迹的最小代价修复方案中再选出全局最小代价修复方案作为问题的最终解。由于存在两个阶段的枚举相乘（如图 2-18 所示），会导致状态空间的爆炸，因此[28]的方法无法有效地给出最优解。

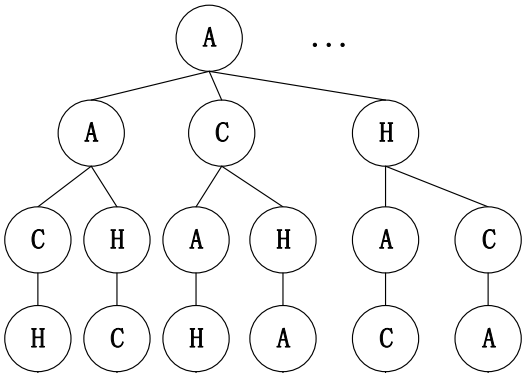


图 2-17 对活动多集{A,A,C,H}进行轨迹枚举展开后得到的状态空间

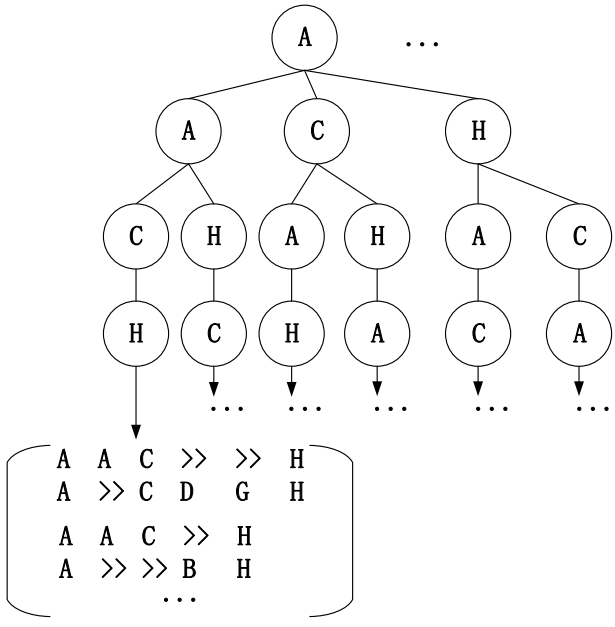


图 2-18 对活动多集{A,A,C,H}进行轨迹枚举和对齐枚举后的状态空间

本课题旨在针对当输入轨迹中的活动为乱序且存在活动缺失或活动冗余（即

输入为活动的活动多集）的情况下，分析日志修复问题的复杂性，并设计相应的算法，解决或避免上述两阶段枚举相乘而产生的爆炸问题，并最终能够给出最小代价的修复方案。

3 研究内容

本章将分别介绍本文的研究内容（如所示）：日志修复问题的形式化定义、日志修复问题的复杂性分析、日志修复问题的多种解决方法和评估对比算法的正确性及性能。

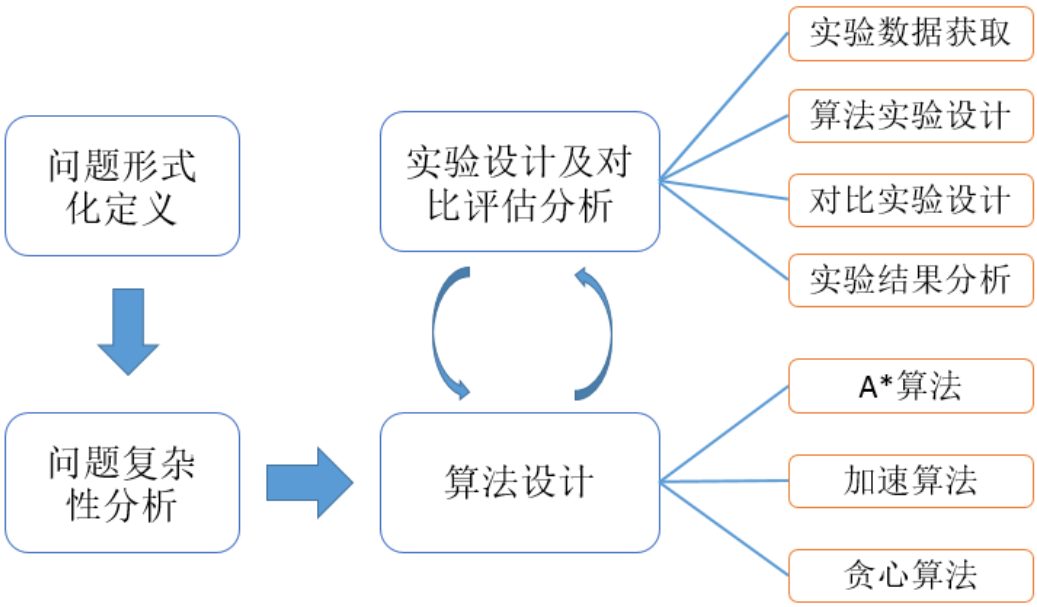


图 3-1 研究内容框架

3.1 日志修复问题的形式化定义

给定一个流程模型（如 Imperative 流程模型、Declarative 模型或者 Hybrid 模型）和一个活动多集，判定该活动多集是否能构成符合该流程模型的一条轨迹，若可以，其轨迹是什么；若不可以，多余或者欠缺的事件是哪些，其中最小代价的修复方案是什么？

为了分析上述问题的复杂性、设计日志修复算法，本文需要给出相应的形式化定义。

3.2 日志修复问题的复杂性分析

在课题研究过程中，本文试图为上述问题求解的复杂度进行分析，验证解决

该问题是否为 NP-Hard 问题、是否有多项式解、是否有启发式解。

3.3 日志修复问题的多种解决方法

在课题研究过程中，本文需要为上述日志修复问题设计多种解决方案。

首先，本文需要设计出 A*算法，得到问题的正确解。在此基础上本文希望通过对估值函数的合理设计及优化、添加各种辅助索引（可达索引、任务索引等）来提升算法性能。此外，本文需要设计贪心策略进而得到求解的贪心算法。

3.4 评估与对比算法结果与性能

实验方案的设计，包括实验数据（模型、日志）的获取、具体实验方案设计、参数调整实验设计等。

在算法、实验方案设计及实现之后，本文将会对新算法的实验结果进行准确度分析。同时，本文将会比较 A*算法、贪心算法和加速算法之间的性能。

除此之外，本文还需要对已有算法进行改造，使其能够处理活动多集，进而同本文所设计的算法进行对比。

4 研究方案与技术路线

4.1 问题的形式化定义

为了形式化地定义本文提出的问题，本文可能需要使用到的概念定义如下：

1. Petri 网相关基础定义；
2. ConDec 模型相关基础定义；
3. 对齐定义；
4. 完全有限前缀定义；
5. 多集定义；

通过使用 Petri 网相关基础定义、ConDec 模型相关基础定义和多集定义来对问题的输入进行描述。通过使用完全有限前缀的定义来分析输入模型。通过使用对齐的定义来描述日志模型修复方案。

4.2 通过归约的方法来验证日志修复问题的复杂性

对于本文所提出的日志修复问题，需要考虑两个方面：

1. 对于给定模型，如何从活动多集中快速组成一条有效的轨迹；

2. 对于给定模型，如何以最小代价地修复一条轨迹使其成为一条有效的轨迹。

在对问题复杂度的研究分析过程中，本文将试图为上述问题的求解过程分别归约为诸如集合覆盖问题、拓扑排序问题等已知的 NP-Hard 问题，进而验证解决本文所提出的问题是否为 NP-Hard 问题、是否有多项式解。

4.3 设计日志修复问题的多种解决方法

4.3.1 A*算法设计

对于一个给定的流程模型，和一个活动多集。在所有可能的修复方案中，如果某一个修复方案的总代价不大于其他任何修复方案的总代价，那么这个修复方案就是对于给定模型和活动多集的一个代价最小的修复方法。更进一步的，如果修复代价为零，即说明在原有的活动多集中存在能够组成有效轨迹的活动集合。在本文希望首先通过使用 A*算法来用修复代价最小的解。

A*算法作为一种在图种的路径搜索算法在[32]中被介绍。A*算法从一个源节点开始，遍历所有的相邻节点直到遇到目标节点，最终算法将会给出全局最优的路径。在 A*算法中，图种的每个节点都被赋予一个代价，该代价是由实际代价 g(n)和估计代价 h(n)的代数和来决定的，其中：

g: $V \rightarrow R^{+}$，该函数将计算从源节点到当前节点的最小代价。

h: $V \rightarrow R^{+}$，该函数是一个启发式的函数，它将估算从当前节点到目标节点的最小代价。

A*算法将会维护一个节点的优先队列。在队列中，代价越小的节点其访问优先级别越高，队列由此可以保证图中节点的访问顺序。算法的执行逻辑如下：

(1) 将源节点放入优先队列。

(2) 从优先队列中选取代价最小的节点，如果该节点属于目标节点，则算法结束并返回路径；如果该节点不属于目标节点，那么将该节点展开：首先计算每个后继节点的代价，然后将它们加入到优先队列中。

(3) 重复步骤(2) 。

在[28]中，算法的输入是一个给定的轨迹， 因此轨迹中的第一个活动节点即为其优先队列的初始化节点。在本文的问题中，输入变得更加松散，没有固定的轨迹起点，因此需要重新考虑优先队列的初始化问题。

其次， 本文需要设计实际代价计算方法，即对函数 g(n)进行设计，来刻画当前操作下累计的修复代价。

通过对活动多集中的活动进行组装可以形成一条轨迹，而在将轨迹同给定模型进行对齐的过程中，对齐方法有：1. 添加已有活动； 2. 添加新的活动。其中

添加已有活动是指从活动多集中提取活动来组装轨迹，添加新的活动是指从模型的变迁集合中选取不在活动多集中的活动，并将其添加到正在组装的轨迹中。组装新轨迹的代价操作即是修复方法中的第二个——添加新的活动。

为了表现当前正在组装的轨迹同流程模型之间的偏离程度以及区分不同修复操作的消耗，需要知道每一种修复办法（即一系列修复操作）的操作总代价。为此，需要一个代价函数K: $\sum_A \rightarrow R_0^+$来赋予每一个操作其对应的代价。最朴素的方法，就是讲每一个操作步骤的代价都赋予一个相同的常数。按照这样的代价函数，每一种修复方法的总代价就是操作步骤的累加和，可以单纯地比较每种修复方法的操作步骤数来衡量当前正在组装的轨迹同流程模型之间的偏离程度。然而，这样的代价函数是粗糙的。每一个修复操作的代价应该由其在流程模型中的特点来决定。因此，需要对不同的修复操作定义不同的代价函数，而每一种修复方法的总代价将是各个修复操作的代价总和，$K(\gamma): \sum_{(s',s'')} \kappa(s',s'')$。在此基础之上，本文还需要设计启发式的估计代价函数，通过使用不同的启发式函数来提升算法性能和准确度。

4.3.2 加速算法设计

通过 CFP 技术，本文可以获得一个流程模型的全部分支。然而，对于一个轨迹的重演或者对齐，本文并不需要尝试所有的分支路径。例如对于那些不包含轨迹活动的分支路径就可以被本文跳过。因此，通过在分支路径上构建索引，可以避免对无用的分支路径的计算[17]。同时，通过建立可达索引，也可以利用活动之间的关系来对轨迹进行剪枝。除此之外，通过利用代价函数，可以对访问路径进行剪枝。

因此，本文将综合利用分支索引和剪枝技术进而提高算法的效率。

4.3.3 贪心算法设计

本文希望验证解决修复无序事件日志的问题是否可以通过贪心策略来实现求解过程。如若可以，则本文希望通过设计合适的贪心策略来对 h(n)函数进行修改，使该启发式函数在使用贪心策略估算当前状态到目标状态距离的同时具备无后效性（即某个状态以前的过程不会影响以后的状态，只与当前状态有关），最终得到高效的算法。

为此，本文将试图寻找问题的最优子结构（即问题的最优解包含其子问题的最优解）[34]，验证由贪心选择得到的子问题的最优解与贪心选择组合在一起能否生成原问题的最优解。

4.4 评估与对比算法结果及性能

针对新算法的实验结果及其效率的评估对比，本小节将从试验数据获取方式、实验方案设计、实验结果准确性及算法效率评估和同其他算法的对比试验四个方

面来进行说明。

4.4.1 实验数据获取方式

本文将从中国移动通信集团公司获取相关的流程模型和实际事件日志，通过对原始事件日志进行处理，使其变成无序日志，最终作为本文问题的输入。

4.4.2 实验方案设计

本文需要针对不同的目的而进行实验设计。下面将分别从输入数据、参数调整和可扩展性三个方面分别介绍实验方案的设计思路。

对于实验输入数据规模，本文需要分别对输入模型和输入活动多集分别进行实验设计。对于输入模型而言，需要考虑模型的结构性和所包含的事件数两个方面。对于模型的结构性，需要对多循环情况、多并发情况、多互斥情况分别进行实验设计，对模型所包含的事件数需要对一般事件数量和较多事件数量分别进行实验设计。对于输入活动多集而言，需要考虑活动多集所包含的任务数量级别来进行实验设计。

对于实验的参数调整，本文需要对算法中可以调整的参数进行定量实验设计，进而得到每个参数对于实验结果的影响。

对于实验的可扩展性，

4.4.3 实验结果准确性评估

对本文而言，实验结果准确性有三层涵义：

1. 算法对输入的活动多集能够组成有效轨迹的判断是否正确；
2. 由算法得到的新的轨迹是否有效；
3. 由算法得到新轨迹的代价是否最小。

针对上述三层含义，本文将设计相应的准确度判别方案，分别判断 A*算法、加速算法和贪心算法的解的准确性。同时，本文也将对比 A*算法同贪心算法的准确度。

4.4.4 实验性能评估

对于实验效果的评估，本文将集中对比 A*算法、加速算法和贪心算法的性能，比较在不同输入规模（模型和活动多集）的情况下，三种算法的性能。

4.4.5 同已有算法的对比实验

本文希望对目前已有的日志修复算法进行改造，使它们能够处理活动多集。为此，本文需要将活动多集转换为轨迹集合，以此作为输入来对已有日志修复算法进行实验，最终同本文的算法实验结果进行对比分析。

5 预期成果及可能的创新点

5.1 预期成果

- 发表学术论文 1-2 篇；
- 基于完全前缀展开的无序事件日志修复 A*算法。
- 基于完全前缀展开的无序事件日志修复加速算法。
- 基于完全前缀展开的无序事件日志修复贪心算法。

5.2 可能的创新点

- 证明对于无序事件日志的修复问题是否是 NP-hard 问题；
- 无序事件日志的高校修复算法。

6 研究计划

1. 文献调研阶段（2015 年 7 月-2015 年 9 月）

调研了符合性检测、流程分支技术、完全有限前缀提取技术、日志模型对齐方法的相关知识，深入理解现有的日志修复研究成果，并重点关注完全有限前缀提取方法和日志模型对齐算法。

2. 验证解决完全松散日志修复问题的复杂性（2015 年 10 月-2016 年 2 月）

基于大量阅读文献和已有成果，为解决完全松散日志修复问题的复杂性进行相关验证。

3. 设计日志修复操作的代价算法及启发式的代价估计算法（2016 年 3 月-2016 年 6 月）

基于完全有限前缀和日志模型对齐方面已有的技术，对完全松散日志的修复操作设计代价计量算法，为每一步修复的代价估计设计启发式预估算法，最终完

成 A*算法的设计。

4. 算法的性质测试和性能评估（2016 年 7 月-2016 年 9 月）

使用大量预设模型和实际模型测试并评估算法准确度和性能。

5. 总结与科技论文撰写（2016 年 10 月-2016 年 12 月）

总结日志修复方法，分析工作中的创新点与共享点，完成一到两篇科技论文的撰写和投稿工作。

6. 撰写毕业论文（2017 年 1 月-2017 年 3 月）

总结研究过程中出现的问题和技术难点，撰写毕业论文。

工作内容	时间段
文献调研阶段	2015.7-2015.9
问题的复杂性验证	2015.10-2016.2
设计修复操作的代价算法及启发式的代价估计算法	2016.3-2016.6
算法的性质测试和性能评估	2016.7-2016.9
总结与科技论文撰写	2016.10-2016.12
撰写毕业论文	2017.1-2017.3

7 参考文献

[1] Adriansyah A, van Dongen B F, van der Aalst W M P. Towards robust conformance checking[C]//Business Process Management Workshops. Springer Berlin Heidelberg, 2011: 122-133.

[2] Pesic M, Schonenberg H, van der Aalst W M P. Declare: Full support for loosely-structured processes[C]//Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International. IEEE, 2007: 287-287.

[3] Pesic M, van der Aalst W M P. A declarative approach for flexible business processes management[C]//Business Process Management Workshops. Springer Berlin Heidelberg, 2006: 169-180.

[4] P. Pichler, B. Weber, S. Zugal, J. Pinggera, J. Mendling, H.A. Reijers, Imperative versus declarative process modeling languages: an empirical investigation, in: Proceedings of

Business Process Management Workshops 2010, Lecture Notes in Business Information Processing, vol. 99, 2011, pp. 383–394.

[5] Maggi F M, Slaats T, Reijers H A. The automated discovery of hybrid processes[M]//Business Process Management. Springer International Publishing, 2014: 392-399.

[6] Peterson J L. Petri net theory and the modeling of systems[J]. 1981.

[7] Karp R M, Miller R E. Parallel program schemata[J]. Journal of Computer and system Sciences, 1969, 3(2): 147-195

[8] Murata T. Petri nets: Properties, analysis and applications[J]. Proceedings of the IEEE, 1989, 77(4): 541-580.

[9] Shatz S M, Tu S, Murata T, et al. An application of Petri net reduction for Ada tasking deadlock analysis[J]. Parallel and Distributed Systems, IEEE Transactions on, 1996, 7(12): 1307-1322.

[10] Valmari A. A stubborn attack on state explosion[C]//Computer-Aided Verification. Springer Berlin Heidelberg, 1991: 156-165.

[11] McMillan K L. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits[C]//Computer Aided Verification. Springer Berlin Heidelberg, 1993: 164-177.

[12] Esparza J, Römer S, Vogler W. An improvement of McMillan's unfolding algorithm[M]//Tools and Algorithms for the Construction and Analysis of Systems. Springer Berlin Heidelberg, 1996: 87-106.

[13] Petri C A. Concurrency theory[M]//Petri Nets: Central Models and their properties. Springer Berlin Heidelberg, 1987: 4-24.

[14] De Leoni M, Maggi F M, van der Aalst W M P. Aligning event logs and declarative process models for conformance checking[M]//Business Process Management. Springer Berlin Heidelberg, 2012: 82-97.

[15] W.M.P. van der Aalst, A. Adriansyah, B.F. van Dongen, Replaying history on process models for conformance checking and performance analysis, Wiley Interdiscip. Rev.: Data Mining Knowl. Discov. 2 (2) (2012) 182–192.

[16] A. Adriansyah, B. van Dongen, W.M.P. van der Aalst, Conformance Checking using Cost-Based Fitness Analysis in: EDOC 2011, IEEE Computer Society, 2011, pp. 55–64.

[17] Adriansyah A, Munoz-Gama J, Carmona J, et al. Measuring precision of modeled behavior[J]. Information Systems and e-Business Management, 2014, 13(1): 37-67.

[18] Munoz-Gama J, Carmona J. Enhancing precision in process conformance: stability, confidence and severity[C]//Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on. IEEE, 2011: 184-191.

[19] Rozinat A, van der Aalst W M P. Conformance checking of processes based on monitoring

real behavior[J]. Information Systems, 2008, 33(1): 64-95.

[20] De Weerdt J, Vanthienen J, Baesens B. Determining process model precision and generalization with weighted artificial negative events[J]. Knowledge and Data Engineering, IEEE Transactions on, 2014, 26(8): 1877-1889.

[21] Rozinat, A., de Medeiros, A.K.A., Gu¨nther, C.W., Weijters, A.J.M.M.T., van der Aalst, W.M.P.: The Need for a Process Mining Evaluation Framework in Research and Practice. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 84–89. Springer, Heidelberg (2008)

[22] 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]

[23] Nielsen M, Plotkin G, Winskel G. Petri nets, event structures and domains, part I[J]. Theoretical Computer Science, 1981, 13(1): 85-108.

[24] Fahland D, van der Aalst W M P. Model repair—aligning process models to reality[J]. Information Systems, 2015, 47: 220-243.

[25] Rogge-Solti A, van der Aalst W M P, Weske M. Discovering stochastic Petri nets with arbitrary delay distributions from event logs[C]//Business Process Management Workshops. Springer International Publishing, 2014: 15-27.

[26] Wang J, Song S, Zhu X, et al. Efficient recovery of missing events[J]. Proceedings of the VLDB Endowment, 2013, 6(10): 841-852. ICDE 2015

[27] Wang J, Song S, Lin X, et al. Cleaning Structured Event Logs: A Graph Repair Approach[J].

[28] De Leoni M, Maggi F M, van der Aalst W M P. An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data[J]. Information Systems, 2015, 47: 258-277.

[29] De Leoni M, van der Aalst W M P, van Dongen B F. Data-and resource-aware conformance checking of business processes[C]//Business Information Systems. Springer Berlin Heidelberg, 2012: 48-59.

[30] Taghiabadi E R, Gromov V, Fahland D, et al. Compliance checking of data-aware and resource-aware compliance requirements[C]//On the Move to Meaningful Internet Systems: OTM 2014 Conferences. Springer Berlin Heidelberg, 2014: 237-257.

[31] Engelfriet J. Branching processes of Petri nets[J]. Acta Informatica, 1991, 28(6): 575-591.

[32] R. Dechter, J. Pearl, Generalized best-first search strategies and the optimality of An, J. ACM (JACM) 32 (1985) 505–536.

[33] V. Levenshtein, Binary codes capable of correcting deletions, inser- tions, and reversals, Sov. Phys. Dokl. 10 (8) (1966) 707–710.

[34] Cormen T H. Introduction to algorithms[M]. MIT press, 2009: 414-425.