软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10. 13328/j.cnki.jos.5809]

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

协同业务过程的建模及正确性修正*

莫 启[1,2], 代 飞[2,3], 笪 建[1,4], 朱 锐[1,2], 谢仲文[1,2], 李彤[1,2]

[1](云南大学 软件学院,云南 昆明 650091)

[2](云南省软件工程重点实验室(云南大学),云南 昆明 650091)

[3](西南林业大学 大数据与智能工程学院,云南 昆明 650091)

[4](淮安开放大学 信息工程系,江苏 淮安 223001)

通讯作者: 代飞, E-mail: 59671019@qq. com

摘 要: 由自底向上建模方法建立的协同业务过程中通常存在不一致，故对其进行正确性分析是确保其正确实施的重要手段. 现有方法大多关注正确性检测，这使得协同业务过程正确性分析过程复杂且耗时. 而正确性修正方法能够避免正确性检测方法中存在的重复检测和调整，但这方面研究较少，不能有效地应用于协同业务过程修正. 为此，基于简单路径提出一种协同业务过程正确性修正方法. 首先，在考虑活动同步及异步交互情况下，将部分正确协同业务过程行为抽象为完整的简单路径，并将其合并成核；然后，利用协调映射技术将核映射为修正业务过程，通过将所有的修正业务过程并发组合建立修正协同业务过程. 修正协同业务过程符合协同业务过程的实际特征，且含有修正前协同业务过程中所有完整轨迹，也未引入隐藏轨迹，从而避免有效性确认. 最后，通过实验与现有的方法进行对比分析，结果表明：相对已有工作，在考虑协同业务过程实际特征情况下，能够更加有效地对协同业务过程进行正确性修正.

关键词: 协同业务过程；正确性修正；简单路径；核；协调映射

中图法分类号: TP311

中文引用格式: 莫启,代飞,笪建,朱锐,谢仲文,李彤. 协同业务过程的建模及正确性修正.软件学报. http://www.jos.org.cn/1000-9825/ 5809.htm

英文引用格式: Mo Q, Dai F, Da J, Zhu R, Xie ZW, Li T. Modeling and Correctness Repairing for Collaborative Business Processes. Ruan Jian Xue Bao/Journal of Software, (in Chinese). http://www.jos.org.cn/1000-9825/5809.htm

Modeling and Correctness Repairing for Collaborative Business Processes

MO Qi[1,2], DAI Fei[2,3], DA Jian[1,4], ZHU Rui[1,2], XIE Zhong-Wen[1,2], LI Tong[1,2]

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

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

[3](School of Big Data and Intelligent Engineering, Southwest Forestry University, Kunming 650091, China)

[4](Department of Information Engineering, Huai'an Open University, Huai'an 223001, China)

* 基金项目: 国家自然科学基金(61862065, 61702442, 61662085); 云南省应用基础研究计划项目(2018FB105); 云南省软件工程重点实验室开放基金面上项目(2017SE201, 2016SE202); 云南省中青年学术和技术带头人后备人才培养经费(C6143002); 云南省教育厅科学研究基金资助性项目(2017ZZX227)

Foundation item: National Natural Science Foundation of China (61862065, 61702442, 61662085); The Application Basic Research Project in Yunnan Province (2018FB105); The Open Foundation of Key Laboratory for Software Engineering of Yunnan Province (2017SE201, 2016SE202); Yunnan Province Young Academic and Technical Leaders Funds for Training (C6143002); The Project of Yunnan Provincial Department of Education Science Research Fund (2017ZZX227)

收稿时间: 2018-5-9; 修改时间: 2018-9-18; 采用时间: 2018-12-29; jos 在线出版时间: 2020-4-21

Abstract: There are usually inconsistencies in collaborative business processes established by the bottom-up modeling method, so the correctness analysis is an important means to ensure its correct implementation. Most of the existing methods focus on correctness detection, which makes the analysis process of correctness of collaborative business processes complicated and time consuming. The correctness repairing method can avoid the duplicate detection and adjustment existing in the correctness detection method. However, this method is less researched and cannot be effectively applied to the repair of collaborative business processes. To this end, we propose a method of repairing the correctness of collaborative business processes based on the complete route. First, we abstract the behaviors of partial correct collaborative business processes into complete simple routes under the consideration of active synchronization and asynchronous interaction, and merged them into a core; Then, we use the coordination mapping to map the core to repaired business processes, and the repaired collaborative business process is established by combining all the repaired business processes concurrently. The repaired collaborative business process conforms to the actual characteristics of collaborative business processes, and contains all complete traces in the pre-repair collaborative business process, and no hidden traces are also introduced, thereby avoiding validation. Finally, we use experiments to compare the proposed method with the existing methods. The results show that compared with the existing work, under the consideration of the actual characteristics of collaborative business processes, we can more effectively repair collaborative business processes.

Key words: Collaborative Business Processes; Correctness Repairing; Simple Routes; Core; Coordination Mapping

随着全球经济化的发展和企业信息化程度的不断提高，企业的经营模式发生了重大的变化，企业的业务活动已从企业内单目标为导向的独立发展模式发展成为跨企业多目标合作的协同模式[1]. 在现代商业环境下，没有一个企业是孤立的[2-3]. 企业作为参与者参与到协作中，在协作过程中，它们彼此进行交互以完成特定业务功能. 近年来，随着 Internet 成为主流的计算平台，尤其是面向服务计算（Service-Oriented Computing，SOC）的快速兴起，使得不同组织业务过程间的交互成为可能，如企业信息系统[4]、电子商务[5]等. 为实现共同的商业目标，每个组织的业务过程通常需要跨越组织边界，同其他组织的业务过程进行交互和协作以形成相对稳定的过程视图. 学术界和工业界称这种存在复杂交互关系的业务过程为协同业务过程[6].

针对协同业务过程，国内外研究者提出多种建模方法. 归纳起来，这些建模方法可分为 2 类：1）自顶向下建模方法[7-8]. 该方法要求首先给出协同全局契约，然后定义映射规则并以此为基础从契约中生成每个组织的业务过程；2）自底向上建模方法[9]. 该方法允许组织独立地定义各自业务过程，之后以此为基础直接构建具有协同功效的协同业务过程. 由于受限于映射规则集，利用自顶向下建模方法通常仅能建立具有有限结构形态业务过程，灵活性和普适性不足[7-8]. 因此，本文关注自底向上建模方法.

然而，自底向上建模方法中的业务过程由不同组织开发，无法在设计阶段就预见其潜在的所有交互可能，因此在实际协作中，参与协作的业务过程间可能存在不一致（如死锁等），无法保证协同业务过程正确地实施. 因此，对协同业务过程的正确性（正确性可根据实际需要定义，如合理性等，在此不做区分）进行分析是当前业务过程管理（Business Process Management, BPM）领域内的研究热点.

针对协同业务过程正确性分析，国内外学者做了大量工作[9-10]. 这些研究工作主要围绕正确性检测方法开展. 该方法先构建形式化的协同业务过程并定义正确性约束（如无死锁、无活锁及未指定接收等），然后采用某种形式验证技术（如模型检测等）在协同业务过程上检测相容性是否满足. 正确性检测方法通常具有检测过程自动化，不需要人工干预，且在检测失败时能够给出诊断信息，便于业务设计人员发现并修正错误. 但其不足是若协同业务过程中存在多处不正确，则需要经过多次检测，且在每次检测后都需要对协同业务过程重新调整. 重复的检测及调整使得协同业务过程正确性分析复杂且耗时.

一种替代方法是正确性修正方法. 该方法关注早期设计阶段，根据正确性约束对业务过程内部结构进行修改以自动构建满足正确性约束的业务过程[11-12]. 这方面研究工作较少，且大多关注组织内单个业务过程，能够用于协同业务过程正确性修正方法[13-15]也存在如下不足：1）修正后的协同业务过程是集中式的，与协同业务过程的实际特征（如自治、分布及面向流程协作等）[16]不符；2）这些方法基于过程挖掘技术[17]提出，不能够保证修正协同业务过程中含有修正前协同业务过程中的所有完整轨迹（即完整简单路径对应活动执行序列），还可能引入隐藏轨迹（即不属于修正前协同业务过程中的其他轨迹），这需要进行额外确认；3）协同

业务过程实质上是由多个业务过程构成的并发交互系统，复杂程度高[16]. 直接利用这些方法对其进行修正存在修正效率低下问题. 例如，我们课题组最近工作[13]结合 Petri 网和过程挖掘的相关理论，提出一种针对协同业务过程修正方法. 但是该方法构建的协同业务过程存在中心流程（即只能通过该中心流程协调原有业务过程执行）. 同时，修正协同业务过程与修正前协同业务过程中含有所有完整轨迹也可能不一致，且存在执行失败情况. 这在实验部分进行详细阐述.

针对上述问题，本文采用标号迁移系统 LTS（Labeled Transition System）描述业务过程，并将其并发组合建模协同业务过程. 在考虑活动同步及异步交互情况下，通过将协同业务过程的行为抽象为简单路径，在此基础上，提出了针对部分正确的协同业务过程修正方法. 该修正方法能够确保修正协同业务过程与修正前的协同业务过程中含有的轨迹一致，且修正协同业务过程具有自治、分布及面向流程组合等特性.

本文主要贡献如下：

(1) 在考虑活动同步及异步交互情况下，提出了简单路径用来刻画协同业务过程的行为，并提出将完整的简单路径合并为核的算法. 本质上，核中包含了修正前的协同业务过程中所有正确的任务执行系列（即完整轨迹）. 以核为基础，提出了将其映射为修正业务过程方法，将所有的修正业务过程并发组合建立修正的协同业务过程；

(2) 理论上证明修正协同业务过程与修正前协同业务过程中含有轨迹一致，从而避免额外确认，降低修正代价

(3) 通过实验分析得出：相对已有的方法，在考虑协同业务过程实际特征（如自治、分布及面向流程组合等）的情况下，本文方法可实现更加有效的正确性修正并能够极大地提高修正效率，减少修正时间.

本文第 1 节给出方法概览；第 2 节给出协同业务过程定义；第 3 节对正确性进行分析；第 4 节对正确性修正方法进行讨论；第 5 节通过实验对本文方法有效性及效率进行评估；第 6 节为相关工作；第 7 节为全文总结.

1 方法概述

本文正确性修正方法提出是建立在如下一种事实，即采用自底向上建模方法在建立协同业务过程中无法预见其所有潜在的交互可能，从而导致建立模型中可能存在异常（如死锁等），这将严重地阻碍业务过程间的正确协作. 本文提出的正确性修正方法框架如图 1 所示.

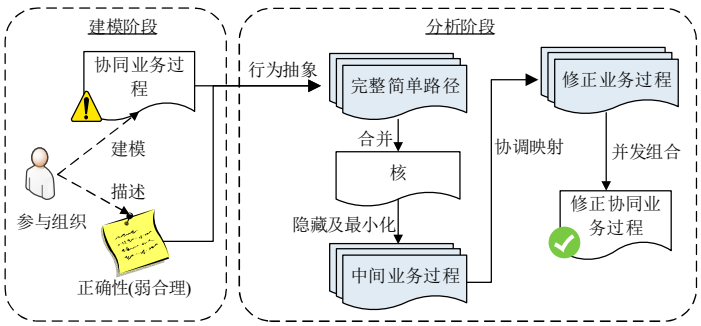

图 1 正确性修正方法概览

具体地，该正确性修正方法由如下两个 2 个阶段组成：

(1) 建模阶段. 参与组织首先独立地采用 LTS 建模各自业务过程，并将这些业务过程并发组合建立协同业务过程；之后参与组织间经协商，确定期望的系统正确性，本文以弱合理[11]定义；

(2) 分析阶段. 分析阶段将协同业务过程和正确性约束输入，自动构建具有正确性的协同业务过程. 分析阶段包含如下 4 个具体步骤：

(a) 基于弱合理将协同业务过程行为抽象为完整的简单路径；
(b) 将抽象的完整简单路径合并以构建核，核中包含修正前的协同业务过程中所有正确的任务执行系列；
(c) 将核隐藏、最小化后生成每个参与组织的中间业务过程；
(d) 将中间业务过程协调映射为修正业务过程，通过将修正业务过程并发组合建立修正协同业务过程.

下面将对图 1 所示的正确性修正方法进行详细地阐述.

2 协同业务过程定义

为了实现协同业务过程与需求一致性检测，参与组织需要首先对协同业务过程及期望需求进行建模.

2.1 协同业务过程建模

对协同业务过程进行形式定义是实现协同业务过程正确性修正的基础. 本节采用标号迁移系统对业务过程建模，并将其并发组合形成协同业务过程.

业务过程由参与组织活动集和建立其上的控制流组成，它由该组织自治管理，是构建协同业务过程基础. 本文采用标号迁移系统建模业务过程.

定义 1(业务过程) 业务过程是一个迁移系统 $BP=(S, s_0, F, A, \Delta)$，其中：

（1）S 为状态集合；

（2）$s_0 \in S$ 为初始状态；

（3）$F \subseteq S$ 为终止状态集合；

（4）$A= A_l \cup A_i$ 为活动集合，其中：A_l 为本地活动集合，A_i 为交互活动集合；

（5）$A_i=A_{si} \cup A_{ai}$，其中：A_{si} 为同步交互活动集合，A_{ai} 为异步交互活动集合；

（6）$A_{ai}=A_{as} \cup A_{ar}$，其中：$A_{as}$ 为异步发送活动集合，A_{ar} 为异步接收活动集合；

（7）$\Delta \subseteq S \times A \times S$ 为状态迁移关系集，用来刻画活动间执行顺序，即控制流. 对于 $\forall (r, a, s) \in \Delta$，记为迁移 $r \xrightarrow{a} s$，表示 BP 在执行活动 a 后由状态 r 转换为状态 s.

特别地，称 BP 是确定的当且仅当对于其中的任意两条迁移 $r \xrightarrow{a_1} s_1$ 及 $r \xrightarrow{a_2} s_2$，若 $a_1=a_2$，则 $s_1=s_2$. 本文中讨论业务过程均默认是确定的. 对于任意同步交互活动 $a \in A_{si}$，则协同业务过程中至少存在一个其他业务过程 BP'，BP'的同步交互活动集合中包含 a；而对于任意异步交互活动 $a \in A_{ai}$，若 $a \in A_{as}$，则 a 记为$!m$，表示发送消息 m 的活动，“!”表示发送动作，若 $a \in A_{ar}$，则 a 记为$?m$，表示接收消息 m 的活动，“?”表示接收动作. 对于业务过程 BP，其发送消息集为 $M^{!}_{BP}=\{m|!m \in A_{as}\}$，而接收消息集为 $M^{?}_{BP}=\{m|?m \in A_{ar}\}$.

采用标号迁移系统建模业务过程有如下 3 个方面原因：1）标号迁移系统具有直观图形表示，可以形式化地刻画系统的行为，是系统行为深入分析的可靠工具[17]；2）迁移系统作为一种中间表示，任意具有执行语义的过程模型（如利用 Petri 网或进程代数建模过程模型）都可以方便地转换为本文中以标号迁移系统描述业务过程[17]；3）便于后文中映射生成修正业务过程. 特别需要说明的是，由于 Petri 网具有直观图形化表示，严格形式语义和丰富的形式分析技术，被广泛地用于业务过程建模和分析中. 本文没有采用 Petri 网进行协同业务过程修正根本原因是在具体修正中需要首先生成协同业务过程完整简单路径，然后合成核（本质上是标号迁移系统），最后通过对核进行协调映射生成每个修正业务过程（也为标号迁移系统）. 而若采用 Petri 网建模协同业务过程，则具体修正仍然需要对应到标号迁移系统上开展，不够直观.

以业务过程为基础可建模协同业务过程. 本质上，协同业务过程可视为由多个业务过程构成的并发交互系统. 进程代数能够自然地以模块的方式构建复杂的交互并发系统[18]，借鉴其思想，本文提出并发操作符概念. 并发操作符提供了一种通过组合业务过程构建协同业务过程方法.

定义 2(协同业务过程) 设参与协同的业务过程为 $BP_1, \ldots, BP_n$，由这 n 业务过程构成的协同业务过程记

为 $CBP = BP_1 \| \ldots \| BP_n$.

特性需要说明的是，定义 2 只是将多个业务过程在形式上列为并发关系，*CBP* 中活动执行及交互可按实际场景定义. 本文同时考虑活动间同步交互及基于消息通信的活动间异步交互，参见定义 5.

对于任意协同业务过程，其状态用格局表示. 它由每个参与组织的运行状态及所配置的先进先出消息队列组成向量表示，形式定义如下.

定义 3(格局) 协同业务过程 *CBP* 状态表示为一个格局 $c = <s_1, Q_1, \ldots, s_n, Q_n>$，其中：$\forall i \in [1..n]$，$s_i$ 为业务过程 BP_i 的运行状态；Q_i 为业务过程 BP_i 的先进先出消息队列，用来存储其接收消息，本文中 Q_i 长度默认为无穷.

特别地，*CBP* 初始格局记为 $c_0 = <s_{01}, NIL, \ldots, s_{0n}, NIL>$，其中：$\forall i \in [1..n]$，$s_{0i}$ 为 BP_i 的初始状态，且其消息队列为空，即为 *NIL*；*CBP* 终止格局记为 $c_e = <s_{e1}, NIL, \ldots, s_{en}, NIL>$，其中：$\forall i \in [1..n]$，$s_{ei}$ 为 BP_i 的终止状态, 且其消息队列为空，表示所有的消息均被接收.

可对格局中的运行状态及消息队列进行重置，形式定义如下.

定义 4(格局重置) 设格局 $c = <s_1, Q_1, \ldots, s_n, Q_n>$，将业务过程 BP_i 的运行状态由 s_i 重置为 s_i'，记为 $c[{}^{s_i'}/_{s_i}]$，而将 BP_i 的消息队列由 Q_i 重置为 Q_i'，记为 $c[{}^{Q_i'}/_{Q_i}]$.

在协同业务过程中，活动执行除了受到本地控制流的约束外，还受到其他业务过程中活动执行情况的影响. 归纳起来，可分为如下 3 种情况：

1）若 $a \in A_l$，则活动 *a* 执行仅受到本地控制流影响；

2）若 $a \in A_{si}$，则活动 *a* 执行除了受到本地控制流影响外，还需要所有含有同步交互活动 *a* 的业务过程同时参与完成；

3）若 a=!m，则活动 *a* 执行除了受到本地控制流影响外，还受到其他需要接收消息 *m* 的业务过程的消息队列 *Q* 影响，即 *Q* 不满，且存在接收消息 *m* 的业务过程 *BP*. 在发送消息 *m* 之后，消息 *m* 的被添加至 *BP* 消息队列的尾部；而若 a=?m，则活动 *a* 执行除了受到本地控制流影响外，还受到自身的消息队列 *Q* 影响，即 *Q* 不空，且 *Q* 的第一个元素是消息 *m*. 在接收消息 *m* 之后，消息 *m* 从 *Q* 的头部移除.

根据上述对活动执行情况分析，定义协同业务过程点火规则.

定义 5(点火规则) 设 $CBP=BP_1\|BP_2\|\ldots\|BP_n$ 为一个协同业务过程，其格局为 $c = <s_1, Q_1, \ldots, s_n, Q_n>$，则其点火规则定义如下：

（1）若 $a \in A_l$，则 $c \xrightarrow{a} c'$，当且仅当 $s_i \xrightarrow{a} s_i' \wedge c'=c[{}^{s_i'}/_{s_i}]$；

（2）若 $a \in A_{si}$，则 $c \xrightarrow{a} c'$，当且仅当 $\forall a \in A_{si}^j \ (1 \le j \le n): s_j \xrightarrow{a} s_j' \wedge c'=c[{}^{s_j'}/_{s_j}]$；

（3）若 a=!m，则 $c \xrightarrow{a} c'$，当且仅当 $s_i \xrightarrow{a} s_i' \wedge \exists i, j \in [1..n]: m \in M_i^! \cap M_j^? \wedge |Q_j| < L_{Qj}$ (L_{Qj} 是 Q_j 的长度) $\wedge Q_j' = Q_j m \wedge c'=c[{}^{Q_j'}/_{Q_j}] \wedge c'=c[{}^{s_i'}/_{s_i}]$；

（4）若 a=?m，则 $c \xrightarrow{a} c'$，当且仅当 $s_i \xrightarrow{a} s_i' \wedge Q_i = mQ_i'' \wedge c'=c[{}^{Q_i''}/_{Q_i}] \wedge c'=c[{}^{s_i'}/_{s_i}]$.

特别需要说明的是，给定协同业务过程 *CBP*，若利用点火规则可产生无穷个格局，则称 *CBP* 具有无穷状态空间，否则称其具有有穷状态空间. 本文关注具有有穷状态空间的协同业务过程.

基于点火规则，我们可定义协同业务过程的行为，表示为路径. 下面给出几个与路径相关的定义.

定义 6(路径) 给定协同业务过程 *CBP*，其行为表示为路径，是 *CBP* 的格局和活动（即本地活动或交互

活动）组成的有穷或无穷序列. 对于任意路径 σ，其形如：$c_0 \xrightarrow{a_1} c_1 \ldots c_{n-1} \xrightarrow{a_n} c_n \ldots$，其中：$\forall i \in [1..n]$，迁移 $c_{i-1} \xrightarrow{a_i} c_i$ 符合定义 5 点火规则，c_i 为 *CBP* 的格局，c_0 为初始格局.

定义 7(路径的轨迹) 对于任意路径 r，$r=c_0 \xrightarrow{a_1} c_1 \ldots c_{n-1} \xrightarrow{a_n} c_n \ldots$，则将 r 中的活动依次连接，得到活动执行序列 $l= a_0{\wedge}a_{1\ldots}a_{n\text{-}1}{\wedge}a_n{\wedge}\ldots$称为 r 的迹，表示为 *trace*(r).

特别地，若格局 c 可通过执行迹 l 迁移至格局 c’，则记为 $c \xrightarrow{l} c$’.

定义 8(路径一致) 设 $r_p=c_{p0} \xrightarrow{a_1} c_{p1\ldots}c_{p(n\text{-}1)} \xrightarrow{a_n} c_{pn}$ 和 $r_q=c_{q0} \xrightarrow{b_1} c_{q1\ldots}c_{q(n\text{-}1)} \xrightarrow{b_n} c_{qn}$ 为两条有穷路径，称 r_p 和 r_q 一致，记为 r_p~r_q，当且仅当 r_p 的迹与 r_q 的迹相同.

3 正确性分析

为了对协同业务过程进行修正，首先需要明确协同业务过程的正确性. 目前，研究者提出多种针对协同业务过程的正确性标准，其中以合理性[9]及其变体（如弱合理[11]等）应用最为广泛. 由于合理性主要用于定义组织内业务过程正确性且较为严格[11]，因此本文利用弱合理性定义协同业务过程正确性.

定义 9(弱合理) 给定协同业务过程 *CBP*，c_0 和 c_e 分别为初始格局和终止格局，称 *CBP* 是弱合理的，当且仅当对于从 c_0 可达的任意格局 c，存在迹 σ，使得 $c \xrightarrow{\sigma} c_e$.

从定义 9 可以看出，弱合理性要求从初始格局可达的任意格局都能够到达终止格局. 这可确保每个业务过程均能够到达终止状态，因此可避免协同业务过程运行中可能出现的死锁和活锁等情形. 同时，终止格局（见定义 3）要求每个业务过程的消息队列均为空，能够确保协同中产生的消息均可合理地被接收.

本文中正确性性分析是基于路径开展. 实际中的协同业务过程通常含有循环结构，循环结构将导致路径数量和长度变为无穷，进而导致正确性检测无法判定. 为将问题收缩至有穷范围，同时记录下必要循环结构，本文定义简单路径.

定义 10(简单路径) 设 $\sigma = c_0 \xrightarrow{a_1} c_{1\ldots}c_{n\text{-}1} \xrightarrow{a_n} c_n \ldots$为协同业务过程 *CBP* 中的一条路径，称 σ 是简单路径，当且仅当对于任意格局 c_i 在 σ 中最多出现 2 次，即对于 $\forall c_i$，$N=\{c_j | c_i = c_j \wedge j \neq i\} \wedge |N| \leq 2$.

特别地，简单路径对应轨迹称为简单轨迹. 下面给出算法 1 用来计算一个协同业务过程中简单路径.

Algorithm1. Generating simple routes

Input: Collaborative business process *CBP*;

Output: Simple routes *R*;

```
1.  curRoute=[c0]; //a stack storing simple routes
2.  visRoutes=[]; //a queue recording visited configurations
3.  do
4.    r=curRoute[len-1], and get all successors Su of r;
5.    if Su=∅ then
6.      save curRoute to R and add r into visRoutes;
7.      curRoute=curRoute[0..len-3]; //backtrack node
8.    else
        //for a successor in Su, it can perform iff it occurs less than 2 times in curRoute or is in visRoutes
9.      if all successor in Su cannot be executed then
10.       save curRoute to R and add r into visRoutes;
11.       curRoute=curRoute[0..len-3]; //backtrack node
12.     else
13.       get the transition r --a--> s;
14.       curRoute=curRoute^a^s;
15.     end if
```

16. **end if**

17. **while** *curRoute*=∅

18. **return** *R*;

本质上，算法 1 是一种深度优先搜索算法. 设 *CBP* 中格局数为 n，且在搜索过程中每个格局入栈次数为 $\{c_1, \ldots, c_n\}$，则算法 1 的时间复杂度为 $O(n\times\sum_{i=m}c_i)$. 进一步地，对于 $\forall i\in[1..n]$，根据定义 10 可知 c_i 最大值为 2（对应算法第 8 行），即 c_i 存在于循环结构中被记录 2 次. 因此，算法 1 在最坏情况下的时间复杂度为 $O(2n^2)$. 特别地，由于本文关注具有有穷状态空间的协同业务过程，即其中含有的格局数是有穷的. 因此，可推出算法 1 是可以终止的, 进而可推出其中含有的简单路径数量及长度也是有穷的.

定义 11(完整的简单路径) 设 $\sigma=c_0\xrightarrow{a_1}c_1\ldots c_{n\text{-}1}\xrightarrow{a_n}c_n$ 为协同业务过程 *CBP* 中的一条简单路径，称 σ 为完整简单路径，当且仅当 c_n 为终止格局.

特别地，完整的简单路径对应轨迹称为完整轨迹. 由弱合理及简单路径，对协同业务过程进行合理性分析可转换为对简单路径分析，见定理 1.

定理 1 设 R 为协同业务过程 *CBP* 中所有的简单路径集合，若对于任意简单路径 $r\in R$，r 是完整简单路径，则 *CBP* 是弱合理的.

证明：设 c_m 为从 c_0 出发且经迹 σ 到达的任意一个格局，则可推出存在路径 $r=c_0\xrightarrow{a_1}c_1\ldots c_{m\text{-}1}\xrightarrow{a_m}c_m$，满足 $trace(r)=\sigma$，分两种情况讨论：

（1）若 c_m 为终止格局，则推出存在空迹 t，使得 $c_m\xrightarrow{t}c_e$，即 $c_0\xrightarrow{\sigma}c_e$.

（2）若 c_m 为非终止格局，分两种情况讨论：

1）若 r 满足 $\forall 1\le i, j\le m\wedge j\ne i$，$c_i\ne c_j$，则可知 r 为某条简单路径 $r_a\in R$ 的前缀片段. 由于 r_s 是完整路径，则可知必定存在迹 t，使得 $c_m\xrightarrow{t}c_e$，即 $c_m\xrightarrow{\sigma^\wedge t}c_e$.

2）若 r 存在片段 $c_i\xrightarrow{a_{i+1}}c_{i+1}\ldots c_j\ldots c_{k\text{-}1}\xrightarrow{a_k}c_k$ $(1\le i, j, k\le m)$，满足 $c_i=c_j=c_k\wedge\forall i<x, y<k\wedge x\ne\{i, j, k\}\wedge y\ne\{i, j, k\}\wedge x\ne y, c_x\ne c_y$，可知轨迹 r 中存在重复次数大于 2 的循环结构，则在 r 中使用 c_i 替换掉片段 $c_i\xrightarrow{a_{i+1}}c_{i+1}\ldots\xrightarrow{a_j}c_j$. 反复执行循环结构替换操作，最终得到替换后的 r'必定为简单路径，则可知 r'为某条简单路径 $r_s\in R$ 的前缀片段. 由于 r_s 是完整路径，则可知必定存在迹 t，使得 $c_m\xrightarrow{t}c_e$，即 $c_0\xrightarrow{\sigma^\wedge t}c_e$.

综合（1）和（2）及定义 9，结论成立.

证毕

基于定理 1，设 R 为协同业务过程 *CBP* 中所有的简单路径集合，R 中所有的完整简单路径为 R_c. 若 $|R_c|=0$，表示 R 中每条简单路径均不是完整路径，则称 *CBP* 是完全不正确的；若 $|R_c|\ne 0\wedge|R_c|<|R|$，表示 R 中有部分简单路径是完整简单路径，则称 *CBP* 是部分正确的；若 $|R_c|=|R|$，表示 R 中所有的简单路径均是完整路径.根据定理 1 可知 *CBP* 满足弱合理性，称 *CBP* 完全正确的.

只有在部分正确情况下，才可能且有必要对协同业务过程进行修正. 以获得的所有完整简单路径为基础，下面详细地阐述针对协同业务过程的修正方法.

4 正确性修正

针对部分正确的协同业务过程，其修正可分 3 步进行：（1）将所有完整的简单路径合并为核；（2）将核映射为修正业务过程；（3）将产生的修正业务过程利用并发操作符组合，构建修正协同业务过程.

对于部分正确的协同业务过程而言，根据第 3 节中阐述可知，其中必定含有完整的简单路径（即能够正确执行完成路径）和非完整的简单路径（即导致协同业务过程执行失败路径）. 为了捕获其中含有的所有正确路径（即完整的简单路径），本文提出了核的概念. 本质上，对于一个部分正确的协同业务过程，它对应的核是含有其中所有完整简单路径的一个迁移系统.

定义 12(核) 核是五元组 $c=(C, c_0, C_e, A, \Delta)$，其中：

（1）C 为格局集合，

（2）$c_0 \in C$ 为初始格局；

（3）$C_e \in C$ 为终止格局集；

（4）A 为活动集合；

（5）$\Delta \in C \times A \times C$ 为格局迁移关系集合.

特别地，对于任意核 c，其行为也可由路径表示. 对于 c 中任意路径 $r=c_0 \xrightarrow{a_1} c_1 \ldots c_{n-1} \xrightarrow{a_n} c_n \ldots$，满足 $\forall i \in [1..n]$，$c_{i-1} \xrightarrow{a_i} c_i$，当且仅当$(c_{i-1}, a_i, c_i) \in \Delta$.

下面提出算法 2，用来将所有完整的简单路径合并以构建核.

Algorithm2. Generating core

Input: All complete simple routes R;

Output: Core c;

19. define returned core $c = (C, c_0, C_e, A, \Delta)$;
20. **for** each complete simple route r in R **do**
21. get the length *len* of complete simple route r
22. **for** $i \in [0..len\text{-}1]$ **do**
23. get *from*=$r[i]$, *task*=$r[i+1]$, and *to*=$r[i+1]$;
24. **if** *from* is not in C **then**
25. add *from* to C;
26. **end if**
27. **if** *to* is not in C **then**
28. add *to* to C;
29. **end if**
30. **if** *task* is not in A **then**
31. add *task* to A;
32. **end if**
33. **if** (*from*, *task*, *to*) is not in Δ **then**
34. add (*from*, *task*, *to*) to Δ;
35. **end if**
36. **end for**
37. **end for**
38. set the srart state to c_0;
39. **for each** elem e in C **do**
40. **if** e is c_e **then**
41. add e to C_e;
42. **end if**
43. **end for**
44. **return** c;

算法 2 的基本思想是通过将完整的简单路径中的格局集和格局迁移集分别设置为核中的格局集和格局迁移集，从而构建核. 算法 2 时间复杂度取决于完整的简单路径的条数及其长度. 不妨设完整的简单路径条数为 m，每条完整的简单路径的长度为 n，则算法 2 时间复杂度为 $O(m \times n)$.

特别需要说明的是：由于完整的简单路径中记录循环结构，因此最终构建的核中循环结构不会丢失. 由此可推断，对于部分正确协同业务过程，通过算法 2 生成的核中只含有此协同业务过程中所有完整的简单路

径，见定理 2.

定理 2 给定部分正确协同业务过程 *CBP*，*CBP* 中含有的所有完整的简单路径记为 R_c. 设 c 是根据 R_c 构建的核，则对于 *CBP* 中任意完整的简单路径 r_c，c 中也存在 r_c，反之亦然.

证明：充分性. 设 $r_c= c_0 \xrightarrow{a_1} c_1 \ldots c_{n-1} \xrightarrow{a_n} c_n$ 为 *CBP* 中任意一条完整的简单路径 r_c，分 2 种情况讨论：

1）若 r_c 满足 $\forall 1 \le i, j \le n \wedge j \ne i$，$c_i \ne c_j$，可知 r_c 为完整的简单路径，推出 $r_c \in R_c$. 根据算法 2 可知，由 R_c 构建后的 c 中必定存在格局迁移关系集合 $\{(c_0, a_1, c_1), \ldots, (c_{n-1}, a_n, c_n)\} \subseteq \Delta$，由此格局迁移关系可生成路径 r_c.

2）若 r_c 中存在路径片段 $c_i \xrightarrow{a_{i+1}} c_{i+1} \ldots c_j \ldots c_{k-1} \xrightarrow{a_k} c_k$ $(1 \le i, j, k \le n)$，满足 $c_i=c_j=c_k \wedge \forall i<x, y<k \wedge x \ne \{i, j, k\} \wedge y \ne \{i, j, k\} \wedge x \ne y$，$c_x \ne c_y$，可知 r_c 中存在重复次数大于 2 的循环结构 $\eta=c_i \xrightarrow{a_{i+1}} c_{i+1} \ldots \xrightarrow{a_j} c_j$，则在 r_c 中使用 c_i 替换掉 η. 反复执行循环结构替换操作，最终得到替换后的 r_c'必定为简单路径，推出 $r_c' \in R_c$. 根据算法 2 可知，由 R_c 构建后的 c 中必定存在格局迁移关系集合 $\{(c_i, a_{i+1}, c_{i+1}), \ldots, (c_{j-1}, a_j, c_i)\} \subseteq \Delta$，由此可生成任意重复次数（设为 p，$p \ge 0$）循环片段，即 η^p. 根据定义 9 可知，轨迹 r_c 的长度有穷，可推出 r_c'中循环片段 η 重复次数是确定整数（设为 q），即 η^q. 由于 $q \le p$，可推出 η^q 可由 c 中格局迁移关系 $\{(c_i, a_{i+1}, c_{i+1}), \ldots, (c_{j-1}, a_j, c_i)\} \subseteq \Delta$ 生成. 同时，对于 r_c'中任意非循环结构片段，即 $c_{g-1} \xrightarrow{a_g} c_g$ $(1 \le g \le n)$，不存在 c_u，c_v $(1 \le u, v \le n \wedge u<g-1 \wedge v>g)$，使得 $c_u=c_v$，r_c'中存在 $c_{g-1} \xrightarrow{a_g} c_g$，根据算法 2 可知，$(c_{g-1}, a_g, c_g) \in \Delta$. 因此，从左至右扫描 r_c，对于非循环结构路径片段 $c_{g-1} \xrightarrow{a_g} c_g$，$r_c$'中存在 $c_{g-1} \xrightarrow{a_g} c_g$，则可推出其可由 c 中 $(c_{g-1}, a_g, c_g) \in \Delta$ 生成，而对于多次重复的循环结构 η^k $(\eta=c_i \xrightarrow{a_{i+1}} c_{i+1} \ldots \xrightarrow{a_j} S_i)$，$r_c$'中存在 $c_i \xrightarrow{a_{i+1}} c_{i+1} \ldots \xrightarrow{a_j} c_i$，则可推出其可由 $\{(c_i, a_{i+1}, c_{i+1}), \ldots, (c_{j-1}, a_j, c_i)\} \subseteq \Delta$ 生成，重复下去最终可由 c 生成 r_c.

必要性. 可按类似方式证明，限于篇幅，略.

证毕

给定部分正确协同业务过程 $CBP=BP_1||BP_2||\ldots||BP_n$，根据算法 2 生成的 *CBP* 的核为 c. 本文中，对 *CBP* 进行修正目标是要保证 *CBP* 中完整简单路径对应轨迹可在修正后的协同业务过程 $CBP_r=BP_{r1}||BP_{r2}||\ldots||BP_{rn}$ 中重现，且 CBP_r 中仅含有这些完整简单路径的轨迹，以避免后续有效性确认，提高修正效率. 其中，$BP_{r1}, \ldots, BP_{rn}$ 分别为业务过程 $BP_1, \ldots, BP_n$ 对应的修正业务过程. 根据定理 2，此问题可转换为保证 CBP_r 和核 c 的行为一致.

为了生成与 *CBP* 行为一致的 CBP_r，基于核 c 本文提出协调映射概念. 协调映射是指在映射过程中引入协调因子（对应活动），用于建模协调交互以实现协调逻辑，从而确保由映射生成 $BP_{r1},\ldots,BP_{rn}$ 并发组合形成 CBP_r 能够遵循 *cc* 行为. 从外界视角观察，在忽视协调因子情况下，协调映射可使得 CBP_r 只包含 *CBP* 中所有完整简单路径的轨迹.

在实际应用中，不是 *CBP* 中的每个活动都需要进行协调. 以核为基础，本文提出协调活动的概念.

定义 13(协调活动) 给定部分正确协同业务过程 $CBP=BP_1||BP_2||\ldots||BP_n$，*CBP* 对应的核 $c=(C, c_0, C_e, A, \Delta)$，称活动 $a \in A_1 \cup A_2 \cup \ldots \cup A_n$ 为协调活动，当且仅当存在格局 $c_1 \in C$，使得 $c_1 \xrightarrow{a} c_2$，且 $c_2 \notin C$，其中：$\forall i \in [1..n]$，A_i 为业务过程 BP_i 的活动集.

根据定义 13 可知，协调活动是指在协同业务过程执行中若不加以控制，则可能导致隐藏路径产生的活动，即从初始格局 c_0 至 c_2 或由 c_2 引出路径. 因此，需要对这些活动进行协调，使其执行后到达正确格局 c_1，避免达到异常格局 c_2. 而对于除协调活动外的其他活动，由于其执行后总能够到达正确状态，因此无需协调.

以核为基础，映射生成修正业务过程之前需先将核中不相关活动（即不属于此业务过程及不是协调活动的其他活动）隐藏，形式定义如下.

定义 14(隐藏) 给定部分正确协同业务过程 *CBP*，$BP=(S, s_0, F, A, \Delta)$ 为 *CBP* 中一个业务过程，*CBP* 对应的核 $c=(C, c_0, C_e, A, \Delta)$，协调活动集为 A_c，则根据 *BP* 及 A_c 对 c 进行隐藏后得到隐藏核为 $c'=(C', c_0', C_e', A', \Delta')$，其中：

（1）$C'= C$；

（2）$A'=\{\tau\} \cup \{a | \forall (r, a, s) \in c.\ \Delta \wedge a \in A \cup A_c\}$；

（3）对于 $\forall (r, a, s) \in \Delta$，如果 $a \in A \cup A_c$，则 $(r, a, s) \in \Delta'$，否则 τ 格局迁移关系 $(r, \tau, s) \in \Delta'$.

由定义 14 可知，隐藏是将 c 中格局迁移关系重置. 即若格局迁移关系中活动是 BP 中活动或是协调活动，则在重置的格局迁移关系中保持不变，否则将其设置为不可见活动 τ.

在对 c 进行隐藏后，需要将产生的隐藏核 c'最小化以生成修正业务过程. 当前，研究者提出多种行为最小化技术[19]，如轨迹最小化、分支最小化及互模拟最小化等. 由于本文关注协同业务过程间的路径一致，即路径对应轨迹相同，因此我们采用文献[20]中提出弱轨迹最小化算法对隐藏核进行最小化操作，最小化后生成一个中间有穷自动机. 本质上，标号迁移系统是一类特殊有穷自动机. 因此，通过将中间有穷自动机中状态集、开始状态、结束状态集、活动集及状态迁移关系分别对应到标号迁移系统中状态集、初始状态、终止状态集、活动集及状态迁移关系，则可由隐藏核生成一个中间业务过程. 这种转换过程非常直观，限于篇幅在此不再赘述. 通过利用文献[20]中提出最小化方法，可确保中间业务过程与隐藏核保持弱轨迹等价，且中间业务过程中不含有不可见活动.

基于最小化后生成的中间业务过程，根据协调活动集下面给出协调映射定义，以生成修正业务过程.

定义 15(协调映射) 给定部分正确协同业务过程 $CBP=BP_1||BP_2||\ldots||BP_n$，$BP=(S, s_0, F, A, \Delta)$为 CBP 中一个业务过程，A_c 为 CBP 中的协调活动集，最小化后生成中间业务过程为 $BP_m=(S_m, s_{m0}, F_m, A_m, \Delta_m)$，则根据 BP 和 A_c 对 BP_m 进行协调映射后得到修正业务过程为 $BP_r=(S_r, s_{r0}, F, A_r, \Delta_r)$，其中：

（1）$S_r = S_m \cup \{s_c | \forall (r, a, s) \in \Delta_m \wedge a \notin A \wedge a \in A_c\} \cup$
$\{s_{c1}, s_{c2} | \forall (r, a, s) \in \Delta_m \wedge a \in A \wedge a \in A_c\}$；

（2）$s_{r0}=s_{m0}$；

（3）$F_r=F_m$；

（4）$A_r = \{a | \forall (r, a, s) \in cc.\ \Delta_m \wedge a \in A \wedge a \notin A_c\} \cup$
$\{\text{SYNC\_1\_}a, \text{SYNC\_2\_}a | \forall (r, a, s) \in \Delta_m \wedge a \in A \wedge a \in A_c\} \cup$
$\{\text{SYNC\_1\_}a, \text{SYNC\_2\_}a | \forall (r, a, s) \in \Delta_m \wedge a \notin A \wedge a \in A_c\}$；

（5）$\Delta_r = \{(r, \text{SYNC\_1\_}a, s_c), (s_c, \text{SYNC\_2\_}a, s) | \forall (r, a, s) \in \Delta_m \wedge a \notin A \wedge a \in A_c\} \cup$
$\{(r, \text{SYNC\_1\_}a, s_{c1}), (s_{c1}, a, s_{c2}), (s_{c2}, \text{SYNC\_2\_}a, s) | \forall (r, a, s) \in \Delta_m \wedge a \in A \wedge a \in A_c\} \cup$
$\{(r, a, s) | \forall (r, a, s) \in cc.\ \Delta_m \wedge a \in A \wedge a \notin A_c\}$.

定义 15 中，对于 BP_m 中一个活动 a：1）若 a 不属于 BP 但需协调，则引入协调因子 SYNC_1_a, SYNC_2_a 及协调状态 s_c，用于将 BP_m 中格局迁移关系(r, a, s)设置为$(r, \text{SYNC_1_}a, s_c)$和$(s_c, \text{SYNC_2_}a, s)$，表示在状态 r 通过协调因子 SYNC_1_a 对活动 a 开始进行协调，并通过协调因子 SYNC_2_a 对活动 a 结束协调；2）若 a 属于 BP 且需协调，则引入协调因子 SYNC_1_a, SYNC_2_a 及协调状态 s_{c1}, s_{c2}，用于将格局迁移关系(r, b, s)设置为三个状态迁移关系$(r, \text{SYNC_1_}a, s_{c1})$, (s_{c1}, a, s_{c2})和$(s_{c2}, \text{SYNC_2_}a, s)$，表示在 a 执行前需进行协调，通过 SYNC_1_a 实现，执行完成后进行结束协调，通过 SYNC_2_a 实现；3）若 a 属于 BP 且无需协调，则格局迁移关系(r, b, s)保持不变.

本质上，修正业务过程是一个迁移系统. 特别地，给定一个业务过程 BP，其修正业务过程 BP'受 BP 行为（即迁移序列）和协调因子影响.根据我们实验发现（见第 5.2 节），BP'结构通常较 BP 复杂. 同时，本文中 BP'采用 LTS 进行描述，LTS 描述模型较其他图形化方式（如 BPMN、BPEL 和 Petri 网）描述模型复杂难懂. 这就限制了本文方法在实际中被用户接受程度. 事实上，研究者已经提出了一些方法[37]用于将以 LTS 描述模型转换为 Petri 网描述模型，且能够保持转换前后行为一致（如保持强互模拟等）. 因此，针对修正业务过程对于用户较难理解这个问题，可以考虑利用工具 Petrify[1]将其转换为 Petri 网来增强其实用性. 有关 Petrify 介绍及使用可以参见文献[37]，限于篇幅，在此不再阐述.

通过将修正业务过程集利用并发操作符组合，得到修正协同业务过程.

---

1 https://www.merriam-webster.com/dictionary/petrify

定义 16(修正协同业务过程) 给定部分正确协同业务过程 $CBP=BP_1||BP_2||\ldots||BP_n$，$BP_{r1}, \ldots, BP_{rn}$ 分别为业务过程 $BP_1, \ldots, BP_n$ 对应的修正业务过程，则修正协同业务过程为 $CBP_r=BP_{r1}||BP_{r2}||\ldots||BP_{rn}$.

在忽视协调因子情况下，修正协同业务过程中只含有修正前的协同业务过程中的所有完整轨迹，见定理 3.

定理 3 给定部分正确协同业务过程 $CBP=BP_1||BP_2||\ldots||BP_n$，$CBP_r=BP_{r1}||BP_{r2}||\ldots||BP_{rn}$ 为 CBP 的修正协同业务过程，则如下结论成立：

（1）对于 CBP 中任意完整的简单路径 r，CBP_r 中存在一条完整的简单路径 r'，满足 $trace(r)=trace(r')\uparrow A_{cf}$；

（2）对于 CBP_r 中任意完整的简单路径 r，CBP 中存在一条完整的简单路径 r'，满足 $trace(r')=trace(r)\uparrow A_{cf}$；

其中：A_{cf} 为 CBP 中所有协调因子集合；$trace(r)\uparrow A_{cf}$ 表示将 r 的任务执行序列中的协调因子移除后形成的新任务执行序列.

证明：（1）设 c 为根据算法 2 生成 CBP 的核，$r=c_0 \xrightarrow{a_1} c_1 \ldots c_{m-1} \xrightarrow{a_m} c_m$ 为 c 中任意一条完整的简单路径. 根据定义 14 可知，对于 r 中任意迁移 $c_{i-1} \xrightarrow{a_i} c_i$ $(1 \le i \le m)$，每个业务过程 BP_n $(1 \le i \le n)$对应隐藏核中存在如下迁移，记为 $C1$：

$$c_{i-1} \xrightarrow{a_i} c_i = \begin{cases} c_{i-1} \xrightarrow{\tau} c_i, & a_i \notin A \cup A_c \\ c_{i-1} \xrightarrow{a_i} c_i, & a_i \in A \cup A_c \end{cases}$$

根据文献[20]中提出最小化方法可知，所有的 τ -格局迁移关系（即由 τ 引起的格局迁移关系）被移除，且满足弱轨迹等价，则最小化隐藏核后生成的业务过程 BP 中有如下迁移，记为 $C2$：

$$c_{i-1} \xrightarrow{a_i} c_i = \begin{cases} s_i = s_{i-1}, & a_i \notin A \cup A_c \\ s_{i-1} \xrightarrow{a_i} s_i, & a_i \in A \cup A_c \end{cases}$$

根据定义 15，每个修正业务过程 BP_{rj} $(1 \le j \le n)$中迁移进行如下协调映射，记为 $C3$：

$$c_{i-1} \xrightarrow{a_i} c_i = \begin{cases} s_i = s_{i-1}, & a_i \notin A \wedge a_i \notin A_c \\ s_{i-1} \xrightarrow{a_i} s_i, & a_i \in A \wedge a_i \notin A_c \\ s_{i-1} \xrightarrow{\text{SYNC\_1\_}a} s_{c1} \xrightarrow{a_i} s_{c2} \xrightarrow{\text{SYNC\_2\_}a} s_i, & a_i \in A \wedge a_i \in A_c \\ s_{i-1} \xrightarrow{\text{SYNC\_1\_}a} s_c \xrightarrow{\text{SYNC\_2\_}a} s_i, & a_i \notin A \wedge a_i \in A_c \end{cases}$$

从左至右依次扫描 r，根据定义 5 中的点火规则可生成如下路径 r'，满足：

$$r'=k_0 \ldots k_{i-1} \xrightarrow{\text{SYNC\_1\_}a_i} k_{(i-1)i1} \xrightarrow{a_i} k_{(i-1)i2} \xrightarrow{\text{SYNC\_2\_}a_i} k_i \ldots k_m \ (a_i \in A_c)$$

或 $r'=k_0 \ldots k_{i-1} \xrightarrow{a_i} k_i \ldots k_m \ (a_i \notin A_c)$

根据 $C3$，由于 r 为完整的简单路径，则可推出 r'中 k_m 也为终止格局，故 r'是完整的简单路径. 进一步地，推出 $trace(r)=trace(r')\uparrow A_{cf}$ 成立. 根据定理 2，c 与 CBP 中完整的简单路径一致，故结论（1）成立.

（2）采用反证法证明. 设 CBP_r 中存在路径片段 $r_1=k_0 \xrightarrow{a_1} k_1 \ldots k_{k-1} \xrightarrow{a_k} k_k$，且 c 中存在对应路径片段 r_2，满足 $trace(r_2)=trace(r_1)\uparrow A_{cf}$. 但 CBP_r 中存在后续迁移 $k_k \xrightarrow{a} k_{k+1}$ 使得路径片段 $r_3=k_0 \xrightarrow{a_1} k_1 \ldots k_{k-1} \xrightarrow{a_k} k_k \xrightarrow{a} k_{k+1}$ 不为 c 中任意完整的简单路径的前缀. 按如下两步证明后续活动 a 不能发生：

1）先明确活动 a 是协调活动，即 $a \in A_c$. 针对路径片段 r_1，由于 c 中存在对应路径片段 r_2，满足 $trace(r_2)=trace(r_1)\uparrow A_{cf}$. 因此，从左至右扫描 r_1，对于 r_1 中路径片段：$k_{i-1} \xrightarrow{\text{SYNC\_1\_}a_i} k_{c1} \xrightarrow{a_i} k_{c2} \xrightarrow{\text{SYNC\_2\_}a_i} k_i$，或者 $k_{i-1} \xrightarrow{a_i} k_i$，则用 $c_{i-1} \xrightarrow{a_i} c_i$ 替换，重复下去最终可求得 k_k 对应于 c_k. 根据 $C3$ 可知，若后续迁移 $k_k \xrightarrow{a} k_{k+1}$ 发生后使得 r_3 不为 c 中任意完整简单路径的前缀，那么 $a \in A_c$.

2）再证明协调活动 a 在 CBP_r 中不能发生. 分两种情况讨论：

i）a 不在 c 中出现，即 $a \notin c.A$. 根据 $C3$ 可知，后续迁移 $k_k \xrightarrow{a} k_{k+1}$ 不存在；

ii）a 在 c 中出现，即 $a \in c.A$. 设有后续有效迁移 $k_k \xrightarrow{b} k_{k+1}$，使得 $r_4=k_0 \xrightarrow{a_1} k_1 \ldots k_{k-1} \xrightarrow{a_k} k_k \xrightarrow{b} k_{k+1}$ 为 c 中某条完整简单路径的前缀. 若活动 a 能执行，分两种情况讨论：

①若 a 和 b 属于同一个修正业务过程 BP_{rj} $(1 \le j \le n)$，则推出 a 和 b 处于选择关系. 根据 $C3$ 可知，若 a 能够执行，则 b 不能执行. 但迁移 $k_k \xrightarrow{b} k_{k+1}$ 是有效的，故活动 a 不能执行；

②若 a 和 b 属于不同的修正业务过程，不妨设分别属于业务过程 BP_{ri} 和 BP_{rj} $(1 \le i, j \le n)$. 根据 $C3$ 可推出，BP_{rj} 中存在迁移 $s_j \xrightarrow{b} s_{j+2}$，$s_j \xrightarrow{\text{SYNC\_1\_}a} s_{c1} \xrightarrow{a} s_{c2} \xrightarrow{\text{SYNC\_2\_}a} s_{j+1}$. 根据定义 5 点火规则，若活动 a 能够执行，则活动 b 不能执行. 但迁移 $k_k \xrightarrow{b} k_{k+1}$ 是有效的，故活动 a 不能执行.

综合 1）和 2）推导可知，结论（2）成立.

证毕

定理 3 中的结论（1）表明修正前的协同业务过中所有完整的简单路径都可以在修正协同业务过程中重现；而结论（2）则表明未引入隐藏轨迹. 这将避免修正后进行有效性确认，提高修正效率.

特别需要明确的是，本文方法可支持带有一般循环结构的过程模型，其原因在于修正是基于完整简单路径开展. 完整简单路径能够记录循环结构（即自循环和一般循环结构[17]），故通过算法 2 构建的核中循环结构也不会丢失. 最终，通过对核进行协调映射产生修正业务过程也含有已有环结构.

5 实验评价

本文提出一种协同业务过程正确性修正方法. 为了评估该方法的有效性，本节使用具有实际意义的协同业务过程集，通过与相关正确性修正方法进行实验对比，从修正协同业务过程具有协同业务过程实际特征、重现完整轨迹及未引入隐藏轨迹等方面来分析不同正确性修正方法之间的有效性差异.

5.1 实验准备

当前针对协同业务过程没有公开的数据集可供实验[21]，而为了评价本文提出方法的有效性，我们从已有的研究论文[22-23]及 BPMN 案例库[1]中选取 6 个较为典型协同业务过程进行实验. 其中：Order Product（记为 *OP*）、Purchase Order（记为 *PO*）及 Travel Booking System（记为 *TBS*）分别作为文献[22-23]中启发案例，用来阐述跨组织业务过程建模及分析方法有效性，代表性较强；Amazon Online Purchase（记为 *AOP*）、The Nobel Prize（记为 *TNP*）和 Incident Management（记为 *IM*）则来自 BPMN 案例库，能够反映实际的协同业务过程场景.

选取的 6 个协同业务过程具有的属性如表 1 所示，其中：No. of peers 表示协同业务过程中含有参与组织个数，No. of peer states, trans 表示协同业务过程中含有的状态数及迁移数. 采用标号迁移系统表示过程模型中的结构有顺序、选择和循环 3 类. 为了表明选取过程模型普遍性，选取的 6 个协同业务过程中含有这 3 类结构. 特别地，循环结构包含自循环和一般意义上循环结构这 2 类[17]. 这 2 类循环结构也反映在过程模型中，如 Purchase Order 中含有自循环，而 Order 中则含有一般循环结构等.

表 1 协同业务过程属性

Collaborative Business Process	No. of peers	No. of peer states, trans
Order	2	6, 7
Purchase Order	4	32, 36
Travel Book System	5	29, 31
Amazon Online Purchase	4	24, 25
The Nobel Prize	5	27, 25
Incident Management	5	29, 32

特别地，由于本文提出正确性修正方法面向部分正确协同业务过程，因此通过项目组讨论对选取 6 个协同业务过程的内部结构进行修改以注入错误，其中错误类型为死锁、活锁及消息未合理接收等.

目前，针对协同业务过程正确性修正方法较少. 本文选择此类中典型方法[13-15]作为实验比较对象，从支持协同业务过程的实际特征、重现完整轨迹及引入隐藏轨迹等方面进行对比分析.

特别地，文献[13]中方法的处理步骤是：先将每个业务过程采用 Petri 网进行建模，利用文献[9]中库所熔合和变迁熔合技术构建全局模型，并获得其所有完备轨迹[17]. 此处完备是指在全局模型中，若活动 b 能够在活动 a 执行后立即执行，则存在一条轨迹 σ，σ 中存在位置 i，满足 $\sigma(i)=a$，且 $\sigma(i+1)=b$；然后，将获得

[1] http://www. bpmn. org/

完备轨迹作为文献[13]中正确性修正方法输入，则可生成中心流程，最后，将中心流程与每个业务过程进行变迁熔合，则构建修正协同业务过程. 而文献[14-15]中方法的处理步骤是：先将每个业务过程采用 Petri 网进行建模；然后利用文献[9]中库所熔合和变迁熔合技术构建全局模型. 针对构建的全局模型，为保持修正模型一致性，获得其完备轨迹；最后，将获得的完备轨迹作为文献[14-15]中模型修正方法输入，则构建修正协同业务过程.

所有实验在一台 PC 上开展，软件环境为：Windows 10 及 JDK 1.7，硬件环境为：处理器为 Inter(R) Core(TM) i5-8250U CPU@ 1.60GHz、内存为 8GB.

5.2 实验结果及分析

5.2.1 支持个性化特征分析

通过对修正协同业务过程分析，可分析本文方法在支持协同业务过程实际特征方面有效性. 跨组织业务过程建模研究表明[6-10]，协同业务过程通常具有 4 类特征：自治性、分布性、交互性和隐私性. 其中：自治性是指每个业务过程由其所属组织管理及运行；分布性是指协同业务过程在结构和执行上具有分布性；交互性是指在业务过程在执行中需要与其他业务过程进行交互以推动全局流程演进；隐私性是指在跨组织环境下，参与组织希望在建模及分析过程避免将其内部流程信息暴露给其他参与组织. 特别地，在实际应用中，这 4 类特征主要是针对构建后的协同业务过程形态（即是否表现为分布式、有无中心流程等）及构建时是否暴露流程信息给参与组织来进行体现. 因此，上文在定义协同业务过程和阐述方法时没有提及这 4 类特征. 此外，应用这 4 类特征对协同业务过程评价主要是从定性角度开展，因此本文未在上文具体证明中及本节中应用实验数据验证这个结果.

特别需要说明的是，实验中默认本文方法和文献[13-15]中方法对协同业务过程进行修正均是由可信第三方 TTP（Trusted Third Party）[24]完成，因此均能支持隐私性. 在实际应用中，参与组织先将各自业务过程提交给 TTP；之后 TTP 对由其组合建立协同业务过程，并对其修正；最后 TTP 将每个修正业务过程返给对应参与组织. 在跨组织环境下，为保护隐私性，这种通过 TTP 对协同业务过程进行分析是常见和合理的[24].

针对选取的 5 个协同业务过程，本文方法和文献[13-15]中方法建立的修正协同业务过程对上述 4 类特征支持情况如表 2 所示，其中：*A* 表示自治性，*D* 表示分布性，*I* 表示交互性，*P* 表示隐私性.

表 2 特征支持结果

Model	Our Method	Method in [13]	Method in [14-15]
Order Product	*A*, *D*, *I*, *P*	*P*	*P*
Purchase Order	*A*, *D*, *I*, *P*	*P*	*P*
Travel Book System	*A*, *D*, *I*, *P*	*P*	*P*
Amazon Online Purchase	*A*, *D*, *I*, *P*	*P*	*P*
The Nobel Prize	*A*, *D*, *I*, *P*	*P*	*P*
Incident Management	*A*, *D*, *I*, *P*	*P*	*P*

从表 2 看出，经本文方法建立修正协同业务过程能够良好地支持上述 4 类特征，而经文献[13-15]中方法建立修正协同业务过程仅能支持隐私性. 因此，相较文献[13-15]中方法，本文方法能够更加有效地支持协同业务过程具有实际特征.

例如，对于协同业务过程集中订单采购 *OP*，*OP* 中业务过程 Customer 和 Vendor 如图 2 所示.

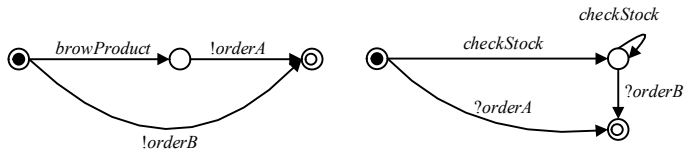

图 2 *OP* 中业务过程 Customer 及 Vendor

利用本文方法建立修正协同业务过程 $OP_{r\text{-}o}$ 由图 3 所示修正业务过程 Customer_r 和 Vendor_r 并发组合形成.

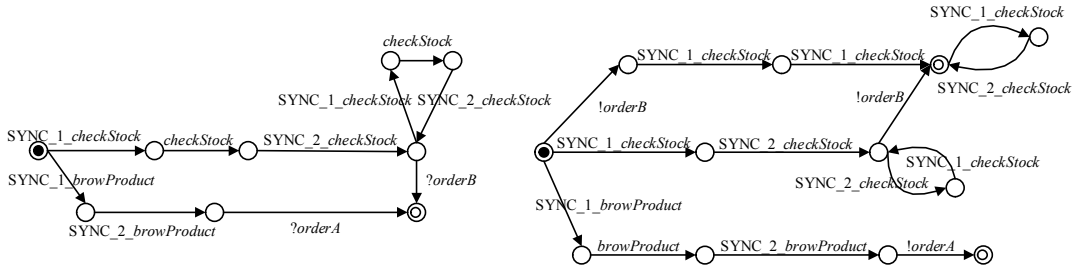

图 3 修正业务过程 Customer$_r$ 及 Vendor$_r$

利用文献[13]中方法建立修正协同业务过程 $OP_{r\text{-}z}$ 如图 4 所示，其中：蓝色虚线表示任务同步关系.

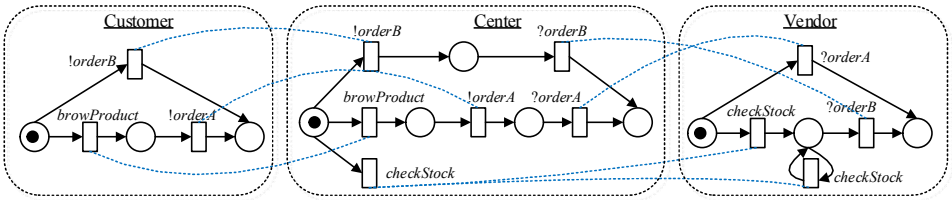

图 4 修正业务过程 $PO_{r\text{-}z}$

利用文献[14-15]中方法建立修正协同业务过程 $OP_{r\text{-}a}$ 如图 5 所示.

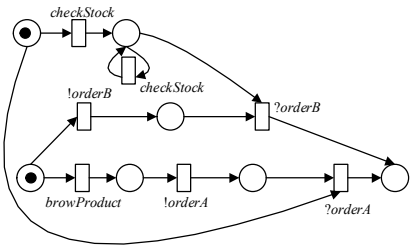

图 5 修正业务过程 $PO_{r\text{-}a}$

通过分析 $OP_{r\text{-}o}$、$OP_{r\text{-}z}$ 和 $OP_{r\text{-}a}$ 可知，$OP_{r\text{-}o}$ 由修正业务过程 Customer$_r$ 和 Vendor$_r$ 并发组合形成，即 $OP_{r\text{-}o}$=Customer$_r$||Vendor$_r$. 由于这 2 个修正业务过程独立存在，且每个修正业务过程由各自组织管理，同时 $OP_{r\text{-}o}$ 按照定义 5 中点火规则运行，表明其具有自治、分布及交互特性. 同时，具体修正由 TTP 完成，表明其具有隐私性；而由文献[13-15]中方法建立修正协同业务过程 $OP_{r\text{-}z}$ 和 $OP_{r\text{-}a}$ 实质上均是集中式的. 特别地，业务过程 Customer 和 Vendor 中活动及活动间交互不是并发执行，而是通过 Center 和 $OP_{r\text{-}a}$ 中控制流约束执行. 例如，Customer 内本地活动 *browProduct* 执行需要与 Center 内活动 *browProduct* 执行同步；而活动!*orderA* 和?*orderA* 间交互则由 Center 内控制流约束.

5.2.2 轨迹有效性分析

本小节分析修正协同业务过程中重现完整轨迹和未引入隐藏轨迹的有效性. 基于简单轨迹，本文提出精确度和泛化度这 2 个有效性计算指标.

精确度计算公式如式(1)所示.

$$P(M') = \frac{|acc(T_M, T_{M'})|}{|T_{M'}|} \tag{1}$$

式(1)中，$T_{M'}$为修正协同业务过程中所有的简单轨迹；T_M 为修正前协同业务过程中所有的完整简单轨迹；$acc(T_M, T_{M'})$为被 T_M 接受的 $T_{M'}$中轨迹子集. 对于精确度而言，计算结果值越高则表示修正协同业务过程中引入隐藏轨迹越少，修正效果越好.

泛化度计算公式如式(2)所示.

$$G(M') = \frac{|acc(T_{M'}, T_M)|}{|T_M|} \tag{2}$$

式(2)中，$T_{M'}$为修正协同业务过程中所有的简单轨迹；T_M为修正前协同业务过程中所有的完整简单轨迹；$acc(T_{M'}, T_M)$为被 $T_{M'}$接受的 T_M中轨迹子集. 对于泛化度指标，计算结果值越高则表示修正协同业务过程中重现修正前协同业务过程中的完整简单轨迹越多，修正效果越好.

根据式(1)和式(2)，本文方法和文献[13-15]中方法精确度和泛化度计算结果分别如图 6(a) - 6(b)所示.

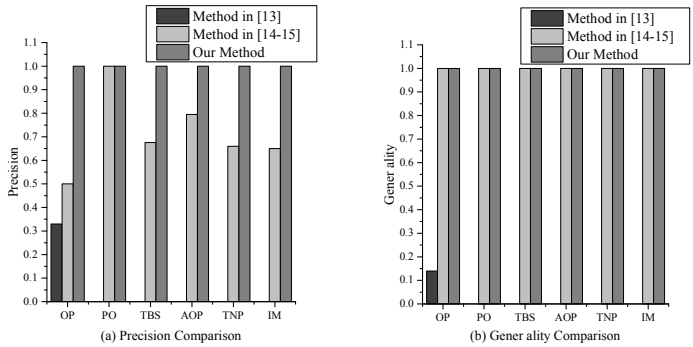

图 6 精确度和泛化度比较

从图 6(a)看出，本文方法的精确度高于文献[13-15]中方法的精确度. 同时，本文方法的泛化度与文献[14-15]中方法相同，但远高于文献[13]中方法. 由此，我们可得出如下结论：1）由于本文方法能够事先保证修正协同业务过程中含有修正前协同业务过程中所有完整轨迹，且未引入隐藏轨迹，使得本文方法的精确度和泛化度均为 1.0. 图 6 所示实验结果也与理论分析一致；2）文献[14-15]中方法在精确度方面表现良好，其均为 1.0，表明修正协同业务过程能够重现修正前的协同业务过程中所有完整轨迹.

在泛化度方面，文献[14-15]中方法的值介于 0.5 至 1.0 之间，表明在修正协同业务过程中引入了隐藏轨迹. 分析发现，这些隐藏轨迹是由修正协同业务过程中存在异常（如死锁等）引起；3）文献[13]中方法在精确度和泛化度方面都表现最差，其精确度和泛化度在选取的协同业务过程中大多为 0.0，表明修正前协同业务过程中几乎所有完整轨迹在修正协同业务过程丢失. 分析发现，这是由于修正协同业务过程中存在异常（如死锁等）和一些活动丢失（如 $PO_{r\text{-}z}$中活动 *checkStock* 丢失等）导致.

5.2.3 修正时间耗费分析

通过分析建立修正协同业务过程时间耗费，可评价本文方法修正效率. 利用本文方法和文献[13-15]中方法对协同业务过程进行正确性修正所耗费时间为 $T_{cr}=T_{ct}+T_r$，其中：T_{ct} 为获取修正前协同业务过程中所有完整轨迹时间，T_r为进行修正所耗费时间；同时，利用文献[13-15]中方法对协同业务过程进行修正后还需要进行有效性确认，所耗费时间为 $T_{cv}=T_{vp}+T_{vg}$，其中：T_{vp}为计算精确度时间，T_{vg}为计算泛化度时间. 本文方法和文献[13-15]中方法建立修正协同业务过程时间耗费如表 3 所示，其中：T_{total}为修正总耗费时间，即为正确性修正耗费时间和有效性确认所耗费时间之和. 特别地，为了确保结果客观性，每次实验重复 10 次，然后取平均值作为建立修正协同业务过程时间耗费.

表 3 时间耗费比较(单位: ms)

Model	Our Method			Method in [13]			Method in [14-15]		
	T_{cr}	T_{cv}	T_{total}	T_{cr}	T_{cv}	T_{total}	T_{cr}	T_{cv}	T_{total}
Order Product	80	0	80	2008	90	2098	38845	94	38939
Purchase Order	6912	0	6912	70122	44103	114225	193710	49773	243483
Travel Book System	3570	0	3570	112850	20900	133750	221971	59696	281667
Amazon Online Purchase	875	0	875	20090	3703	23793	69808	9177	78985
The Nobel Prize	578	0	578	4933	2895	7828	65537	8288	73825
Incident Management	10477	0	10477	39209	38559	77768	129672	131178	260850

从表 3 看出：1）利用本文方法修正所耗费时间远远小于文献[13-15]中方法，例如，针对 Travel Book System，本文方法修正耗费时间为 3570ms，而文献[13]和文献[14-15]中方法分别耗费 133750ms 和 281667ms. 分析发现，文献[13-15]中方法修正效率低下原因是由于上述 6 个协同业务过程中含有较多完备轨迹（如 Travel Book System 中含有 1458 条完备轨迹）和采用过程挖掘技术所致. 可以预见，若对更加复杂协同业务过程进行修正，则其时间将进一步增加. 而本文方法只需要获取所有的完整简单路径（如 Travel Book System 中只含有 384 条），并对核协调映射则可生成修正协同业务过程，避免文献[13-15]中方法使用过程模型挖掘算法（如 α 算法等）、案例差异检测等步骤，降低修正时间；2）由于本文方法能够事先保证修正协同业务过程中含有修正前的协同业务过程中所有完整轨迹，且未引入隐藏轨迹，这使得本文方法无须进行有效性确认，即有效性确认耗费时间为 0ms. 而文献[13-15]中方法有效性确认所耗费时间均较高，例如，针对 Travel Book System，文献[13]和文献[14-15]中方法进行有效性确认耗费时间分别为 20900ms 和 59696ms. 分析发现，这是由于选取的协同业务过程和对应修正协同业务过程中均含有较多的简单轨迹和完整轨迹所致. 可以预见，若对更加复杂协同业务过程进行有效性确认，则其所耗费时间将进一步增加；3）从修正耗费总时间观察，相较文献[13]和文献[14-15]中方法，本文方法的修正效率有极大提高，分别提高 14 倍和 42 倍.

综上分析，可以得出如下结论：相比文献[13-15]中提出正确性修正方法，在考虑协同业务过程实际特征的情况下，由于本文方法能够事先确保修正协同业务过程中含有修正前协同业务过程中所有完整轨迹，且未引入隐藏轨迹，从而使得本文方法可对协同业务过程进行更加有效修正，且能够极大地减少修正耗费时间.

6 相关工作

目前，针对协同业务过程正确性分析方法可分为两类：正确性检测方法和正确性修正方法. 下面对这两类方法中的一些典型文献进行介绍和分析.

6.1 正确性检测方法

国内外研究者针对协同业务过程正确性检测方法开展大量的研究. 归纳起来，这些方法可分为三类：基于 Petri 网检测方法、基于进程代数检测方法及基于自动机检测方法.

以传统的组织内业务过程建模方法为基础，文献[9]较早在国际上提出了 IOWF（Inter-Organizational Workflow）用于建模跨组织业务过程. 针对构建跨组织业务过程提出合理性概念用来定义其正确性，并基于 Petri 网的可达图提出跨组织业务过程正确性检测方法. 之后，国内外研究者以此为基础开展了大量的研究. 如文献[10]针对跨部门业务过程协同呈现出越来越复杂的特点，对 WF-net 扩展资源和消息等要素以建模参与部门的业务过程，继而使用库所熔合技术将资源库所和消息库所熔合从而得到跨部门协调业务过程模型. 针对构建跨部门业务过程，基于合理性定义其正确性并提出验证方法. 跨组织工作流网 IWF-nets（Inter-Organizational Workflow Nets）可有效地建模协同业务过程间基于消息的协作. 为了确保协同业务实施的正确性，文献[25]引入了兼容性和弱兼容性的概念，并针对 IWF-nets 的网结构提出了用于判定兼容性或弱兼容性的充要条件. 针对公共管理呈现出交互特征，为确保其正确地实施，文献[26]首先使用 BPMN（Business Process Modeling Notation）对其进行描述，之后将其 BPMN 模型转换成基于 Petri 网的模型，并采用 Petri 网的展开技术（Unfolding-based Technique）对相关性质（如死锁等）进行验证. 为有效地验证协同业务过程的行为正确性，文献[27]首先利用 BPMN 建模协同业务过程，然后将每个参与组织的流程转换为一类高级 Petri 网 ECATNets（Recursive ECATNets），最后基于库所熔合技术将其组合得到以 ECATNets 描述的模型. 由于该模型的语义被解释为条件重写逻辑，进而可以利用 Maude LTL 模型检测器对协同业务过程的行为正确性性质进行验证.

基于 Pi 演算，文献[16]较早提出一种利用进程代数建模跨组织业务过程方法. 即利用 Pi 演算的并发算子，将跨组织业务过程建模为一组自治且并发执行的组织内子流程的组合，子流程建模为组织内本地流程定义和组织间控制约束的组合，并利用工具 MWB（Mobile Workbench）验证抽象正确性. 为检测协同业务过程能够正确地协调，文献[28]提出了一种针对多个业务过程组合验证的概念框架. 它将 BPMN 建模的协同业务过程

转换以时间通信顺序进程 CSP+T（Communicating Sequential Processes+Time）表示进程，之后采用时序逻辑公式定义期望正确性性质，并利用模型检测方法在工具 FDR2（Failures-Divergences Refinement）上自动验证正确性性质是否满足. 为实现面向 Web 服务的流程无缝集成，文献[29]提出了一种基于递归组合代数的 Web 服务交互流程建模和验证方法. 先利用递归组合代数建模 Web 服务流程交互，并将其转换为递归组合交互图，之后利用递归组合规约语言来描述需求，并在递归组合交互图自动验证需求是否满足与否.

为了使得多个基于 Web 服务的业务过程的组合符合用户定义适配需求，文献[30]提出一种描述及验证多业务过程间行为适配方法. 它采用 LTS 描述业务过程及适配器，进而将业务过程与适配器组合来检测其是否相容来判断多业务过程的组合是否符合用户定义适配需求. 互操作是业务协同正确实施须具备的先决条件，为确保每个参与组织的互操作是否符合规定的需求，文献[31]首先使用 BPMN 来描述协同业务过程，之后将其 BPMN 模型转换成模型检测工具 UPPAAL 中的时间自动机网络（Timed Automata Network, TAN），并采用时序逻辑 TCTL（Time Computation Tree Logic）描述互操作需求从而实现互操作自动检测. 特别地，BPMN 模型中每个参与组织业务过程在 TAN 中对应一个时间自动机（Timed Automata, TA），这些 TA 并发组合构成了 TAN，即协同业务过程. 在后续工作中，为了更加有效地存储和描述互操作需求，文献[32]首先提出了一种领域描述语言用来描述互操作需求，继而将描述需求自动转换为时序逻辑 TCTL，并采用 TAN 建模协同业务过程. 通过 UPPAAL 实现互操作需求的自动验证.

上述提出正确性检测方法通常具有检测过程自动化，不需要人工干预，且在检测失败时能够给出诊断信息，便于业务设计人员发现并修正错误. 但其不足是若协同业务过程中存在多处不正确，则需要经过多次检测，且在每次检测后都需要对协同业务过程重新调整. 这种设计、验证、分析及纠错的过程使得协同业务过程正确性分析复杂且耗时.

6.2 正确性修正方法

相比正确性检测方法，针对业务过程正确性修正方法是较新课题，相关研究工作较少. 文献[11-12]针对以 Artifact 为中心业务过程提出一种正确性保持的设计方法. 它先采用 Petri 网建模以 Artifact 为中心业务过程，之后以合规性和弱合理定义其正确性，最后将业务过程与正确性合成以自动构建具有正确性的业务过程. 文献[33]采用开放工作流网 oWFN（Open Workflow Nets）建模跨组织环境下的业务过程，利用 CTL 公式定义期望正确性性质，根据以 CTL 公式定义性质对业务过程内部结构进行修改来构建跨组织环境下具有正确性的业务过程. 然而，上述提出正确性修正方法均面向组织内的单个业务过程. 由于协同业务过程具有自治性、分布性及涉及多个业务过程间同步或异步交互，因此上述提出方法不适用跨组织环境下协同业务过程.

近年来，面向轨迹的业务过程修正方法引起部分学者关注. 文献[14]首次在国际上提出此类方法. 首先利用已有的一致性检测来发现过程模型与轨迹间差异，并将体现这些差异轨迹从原轨迹中分离出来，以子轨迹表示. 针对这些子轨迹，构建其对应子结构并将其添加至原业务过程中，从而完成业务过程修正. 随后，文献[15]对文献[14]工作做了一些改进，即针对一致性检测中发现循环结构，构建其对应的子过程并将其添加至修正模型中. 文献[34]提出修正方法沿用文献[14-15]的思路. 针对工作流模型，首先利用监控矩阵刻画实际工作流模型足迹，通过比较工作流模型轨迹与足迹间差异，提出四种用于修正工作流模型算法. 我们课题组最近工作[13]结合 Petri 网和过程挖掘的相关理论，提出一种针对协同业务过程修正方法. 该方法首先获取部分正确协同业务过程中所有完整轨迹，之后利用 α 算法从这些完整轨迹构建相容协同业务过程，最后通过设置协同业务过程与业务过程间的任务执行同步关系以协调业务过程正确运行. 然而，利用上述方法建立的修正业务过程仍是集中式的，也存在修正前协同业务过程中完整轨迹可能会丢失和修正协同业务过程中可能会引入隐藏轨迹等问题. 相较而言，本文方法构建的协同业务过程是完全分布式的（即不存在中心流程），它执行能够正确终止且与修正前协同业务过程中含有的所有完整轨迹一致.

面向 Web 服务流程的自动组合方法也是与协同业务过程正确性修正比较相关工作. 文献[35]先裁剪合理性定义正确性（即相容性），针对部分相容 Web 服务流程采用 Petri 网建模每个服务的流程，然后将其进行组合并分析其相容性. 若分析结果部分相容，则产生适配器将来协调服务流程，使之前部分相容的服务流程转

换为完整相容，且未改变原有服务流程的内部结构. 为了应对服务流程间存在时序约束及消息匹配问题，文献[36]先采用 Petri 网建模每个服务的流程，然后根据消息匹配情况及时序约束构建适配器，从而使得所有服务的流程与适配器组合执行是正确的，以避免出现时序异常. 面向 Web 服务流程组合与通过传统业务过程组合建立协同业务过程最大区别是服务使用者通常没有修改第三方服务的权限，也不能要求服务提供者按照自己需求修改提供的服务. 因此，针对 Web 服务流程自动组合通常利用适配器实现，而针对业务过程组合主要是由可信第三方在设计阶段修改业务过程的内部结构来实现[35].

7 总结

在考虑活动同步及异步交互情况下，本文基于完整简单路径提出一种协同业务过程正确性修正方法. 该方法将部分正确的协同业务过程中所有的完整简单路径合并以构建核，通过协调映射来生成修正业务过程，将修正业务过程并发组合建立修正协同业务过程. 该方法一方面可确保修正协同业务过程符合其典型特征（如自治、分布等）；另一方面，修正协同业务过程只含有修正前协同业务过程中所有完整简单路径对应轨迹，可避免有效性确认，提高了修正效率.

未来工作主要针对以下两个方面的问题开展研究：1）本文定义正确性关注控制流，而数据流也是影响业务过程协作的一个重要因素. 因此，下一步将提出同时考虑控制流和数据流的协同业务过程正确性修正方法；2）弱合理可归于一般意义上正确性[17]，而参与组织期望系统功能或特性可能多种多样. 如何在考虑用户需求层面对协同业务过程进行修正是值得深入研究课题；3）本文方法在修正中需要先生成协同业务过程完整简单路径，然后合成核，最后通过对核进行协调映射生成每个修正业务过程. 在实际应用中，由于受限于协同业务过程的状态空间，本文方法也面临状态空间爆炸问题. 然而，本文主要关注修正方法的有效性（即修正协同业务过程符合实际特征及修正前后完整简单轨迹保持一致）和效率（即修正耗费时间较短）问题. 如何避免修正中出现的状态空间爆炸问题将在未来工作中着重讨论.

References:

[1] Lu Yahui, Ming Zhong, Zhang Li. Collaboration patterns of business process[J]. Computer Integrated Manufacturer Systems, 2011, 17(8): 1570-1579 (in Chinese)

[2] Sill A. Cloud, data, and business process standards for manufacturing[J]. IEEE Cloud Computing, 2016, 3(4): 74-80

[3] Yousfi A, Freitas AD, Dey A, et al. The use of ubiquitous computing for business process improvement[J]. IEEE Trans on Services Computing, 2016, 9(4): 621-632

[4] Li Ying, Luo Zhiling, Yin Jianwei, et al. Enterprise pattern: integrating the business process into a unified enterprise model of modern service company[J]. Enterprise Information Systems, 2017, 11(1): 1-21

[5] Yu Wangyang, Yan Chungang, Ding Zhijun, et al. Modeling and verification of online shopping business processes by considering malicious behavior patterns[J]. IEEE Trans on Automation Science and Engineering, 2016, 13(2): 647-662

[6] Mo Qi, Dai Fei, Zhu Rui, et al. An approach to extract public process from private process for building business collaboration[J]. Journal of Computer Research and Development, 2017, 54(9): 1892-1908 (in Chinese)

[7] Aalst W, Lohmann N, Massuthe P, et al. From public views to private view correctness by design for services[C]//LNCS 4937: Proc of the 4th International Workshop on Web Service and Formal Methods. Berlin: Springer, 2007: 139-153

[8] Aalst W, Weske M. The p2p approach to interorganizational Workflows[M]. Berlin: Springer, 2013

[9] Aalst W. Modeling and analyzing interorganizational workflows[C]//Proc of the 1th International Conference on Application of Concurrency to System Design. Los Alamitos, CA: IEEE Computer Society, 1998: 262-272

[10] Zeng Qingtian, Lu Faming, Liu Cong, et al. Modeling and verification for cross-department collaborative business processes using extended Petri nets[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2015, 45(2): 349-362

[11] Lohmann N. Compliance by design for artifact-centric business processes[J]. Information Systems, 2013, 38 (4): 606-618

[12] Lohmann N. Compliance by design for artifact-centric business processes[C]//Proc of the 9th International Conference on Business Process Management. Berlin: Springer, 2011: 99-115

[13] Zheng Ming, Mo Qi, Zhou Xiaoxuan, et al. Compatibility detection and correction of collaborative business process[J]. Journal of Frontiers of Computer Science and Technology, 2017, 11(12): 1907-1921 (in Chinese)

[14] Fahland D, Aalst W. Repairing process models to reflect reality[J]. LNCS, 2012, 7481: 229-245

[15] Fahland D, Aalst W. Model repair-aligning process models to reality[J]. Information Systems, 2015, 47(1): 220-243

[16] Zhang Jing, Wang Haiyang, Cui Lizhen. Research on cross-organization workflow modeling based on pi-calculus[J]. Journal of Computer Research and Development, 2007, 44(7): 1244-1251(in Chinese)

[17] Aalst W. Process Mining: Discovery, Conformance and Enhancement of Business Processes[M]. Berlin: Springer, 2011

[18] Milner R. Communication and Concurrency[M]. London: Prentice Hall, 1989

[19] Glabbeek RJV. The linear time - branching time spectrum[J]. LNCS, 1990, 3(1): 278-297

[20] J. E. Hopcroft, J. D. Ullman. Introduction to Automata Theory, Languages and Computation[M]. Boston: Addison Wesley, 1979

[21] Borkowski M, Fdhila W, Nardelli M, et al. Event-based failure prediction in distributed business processes[J]. Information Systems, 2017, DOI: https://doi. org/10. 1016/j. is. 2017. 12. 005

[22] Zhang Li, Lu Yahui, Xu Fangqian. Unified modeling and analysis of collaboration business process based on Petri nets and Pi calculus[J]. IET Software, 2010, 4(5): 303-317

[23] Xiao Fangxiong, Huang Zhiqiu, Cao Zining, et al. Describing and cost analyzing of web services composition using PPA[J]. Journal of Computer Research and Development, 2009, 46(5): 832-840 (in Chinese)

[24] Norta A, Eshuis R. Specification and verification of harmonized business-process collaborations[J]. Information Systems Frontiers, 2010, 12(4): 457-479

[25] Liu Guanjun, Jiang Changjun. Net-structure-based conditions to decide compatibility and weak compatibility for a class of inter-organizational workflow nets[J]. Science China Information Sciences, 2015, 58(7): 1-16

[26] Corradini F, Polini A, Re B. Inter-organizational business process verification in public administration[J]. Business Process Management Journal, 2015, 21(5): 1040-1065

[27] Kheldoun A，Barkaoui K，Ioualalen M. Formal verification of complex business processes based on high-level Petri nets[J]. Information Sciences, 2017, 385-386: 39-54

[28] Mendoza LE, Capel MI, Rez M, et al. Conceptual framework for business processes compositional verification[J]. Information & Software Technology, 2012, 54(2): 149-161

[29] Rai GN, Gangadharan GR, Padmanabhan V, et al. Web service interaction modeling and verification using recursive[J]. IEEE Transactions on Services Computing, 2018, DOI: 10. 1109/TSC. 2018. 2789454.

[30] Camara J，Salaün J，Canal C, et al. Interactive specification and verification of behavioral adaptation contracts[J]. Information & Software Technology, 2012, 54(7): 701-723

[31] Daclin N. Enabling model checking for collaborative process analysis: from BPMN to network of timed automata[J]. Enterprise Information Systems, 2014, 9(3): 279-299.

[32] Daclin N, Daclin SM, Chapurlat V, et al. Writing and verifying interoperability requirements: application to collaborative processes[J]. Computers in Industry, 2016, 82: 1-18.

[33] Martinez-Araiza U, Lopez-Mellado E. CTL model repair for inter-organizational business processes modelled as oWFN[J]. IFAC-PapersOnLine, 2016, 49 (2): 6-11

[34] Sun Yanan, Du Yuyue, Li Maozhen. A repair of workflow models based on mirroring matrices[J]. International Journal of Parallel Programming, 2016, 45(4): 1-20

[35] Tan Wei, Fan Yushun, Zhou Mengchu. A petri net-based method for compatibility analysis and composition of web services in business process execution language[J]. IEEE Trans on Automation Science and Engineering, 2009, 6(1): 94-106

[36] Du Y, Tan W, Zhou MC. Timed compatibility analysis of web service composition: a modular approach based on petri nets[J]. IEEE Transactions on Automation Science & Engineering, 2014, 11(2): 594-606

[37] Cortadella J, Kishinevsky M, Lavagno L, et al. Deriving Petri nets from finite transition systems[J]. IEEE Transactions on computers, 1998, 47(8): 859-882

附中文参考文献:

[1] 卢亚辉,明仲,张力.业务过程协同模式的研究.计算机集成制造系统,2011,17(8):1570-1579.

[6] 莫启,代飞,朱锐,等.从私有过程提取公共过程构建业务协同的方法.计算机研究与发展,2017,9(54):1892-1908.

[13]郑明,莫启,周小煊,等.协同业务过程的相容性检测及修正.计算机科学与探索,2017,11(12):1907-1921

[16]张静,王海洋,崔立真.基于 Pi 演算的跨组织工作流建模研究.计算机研究与发展,2007,44(7):1243-1251.

[23]肖芳雄,黄志球,曹子宁,等.基于价格进程代数的 Web 服务组合描述和成本分析.计算机研究与发展,2009,46(5):832-840.