计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.201100157

基于分组公平控制流结构的流程变体合并方法

王吴松　方　欢　郑雪文
安徽理工大学数学与大数据学院　安徽 淮南 232001
(2812374185@qq.com)

摘　要　合并流程变体模型能够快速地构建满足新需求的单一流程模型,对流程变体进行合并具有较大的实际应用价值,因此,文中提出了一种利用分组公平控制流结构的流程变体合并方法。首先,利用 Petri 网中的分组公平将流程变体分割为单个的变体片段;其次,提取出变体片段的控制流路径,并在此基础上构建其矩阵表现形式,进而将流程变体合并为单一的流程模型;最后,通过形式化证明验证合并后的流程模型可以捕获输入流程模型的所有行为,并且可以检测在合并模型中是否产生了不期望的行为。

关键词:流程变体;变体片段;流程合并;Petri 网;业务系统
中图法分类号　TP301.1

Process Variants Merging Method Based on Group-fair Control Flow Structure

WANG Wu-song,FANG Huan and ZHENG Xue-wen
College of Mathematics and Big Data,Anhui University of Science and Technology,Huainan,Anhui 232001,China

Abstract　Merging process variants models can quickly construct a single process model to meet a new demand. The issue of how to merge the process variants models is of great practical value. Therefore,a process variants merging method using group-fair control flow structure is proposed. Firstly,process variants are segmented into individual variant using group-fairness in Petri nets. Then,the control flow paths of the variant fragments are extracted and their corresponding matrix representation are constructed,then the variants are merged into a single flow model. Finally,it is proved that the merged process model captures all the behaviors of the input process models,and it can detect the unexpected behaviors of the merged model compared to the former input models.

Keywords　Process variants,Variant fragment,Process merging,Petri nets,Business system

1　引言

目前,许多组织都在使用进程感知信息系统(Process-Aware Information System)来管理其业务流程,在业务组织管理过程中经常出现一种情况,即针对某个的特定业务流程,往往存在多个对等的流程变体。因此,流程变体(process variants)也被定义为系统流程簇(模型)中相似但不同(similar-but-different)的模型。如产品的多种销售流程或不同国家的多种记账流程,以及流程模型的配置可能会随着时间、地理位置、不同的业务模块、产品以及客户类型的变化而变化。

在组织合并、收购或者重组的背景下,为了对业务流程模型进行管理,组织管理者若基于从头构建业务流程的流程开发方法,会出现成本高、耗时长且缺乏灵活性等诸多问题。为了克服这些问题,业务流程设计者往往将现有的业务流程模型重用到新的业务流程中,而不是采用重新开发的方法。因此,如何将属于不同组织或分支机构的诸多流程变体进行合并,并消除冗余,使之成为特定业务背景下的单一流程模型,能够通过创建协同作用来降低运营成本,是一项具有良好实践意义的研究课题。

近年来,随着流程模型设计技术的不断发展,研究者提出了许多针对流程变体的业务流程模型设计方法。如测量业务流程之间的相似性[1-3]和合并业务流程模型[4-9]。

流程变体是由具备共性与差异性的若干流程片段构成,因此变体之间存在一定的相似性,并且这种相似性已经成为流程变体簇中普遍存在的一种现象。如果这些片段能够被识别并分解为一个共享的子进程,那么流程变体簇的可维护性就会大大提高。为此,文献[1]提出了一种精细化过程结构树

到稿日期:2020-11-23　返修日期:2021-04-18
基金项目:国家自然科学基金(61572035,61902002);安徽省自然科学基金(1608085QF149);安徽省高校优秀青年人才基金(gxyqZD2018038);安徽省博士后基金(2018B288)
This work was supported by the National Natural Science Foundation of China (61572035, 61902002), Anhui Natural Science Fund (1608085QF149), Anhui University Excellent Young Talents Fund(gxyqZD2018038) and Anhui Post-doctoral Fund(2018B288).
通信作者:方欢(fanghuan0307@163.com)

与字符串匹配技术相结合的索引结构(Index Structure Combining Refined Process Structure Tree with String Matching，RPSDAG)，这种索引结构支持流程变体簇中重复片段(精确克隆)的快速检测。但是，这种索引结构仅限于寻找到流程变体簇中的精确克隆(重复片段)。因此，文献[2]在文献[1]的基础上提出了一种将 RPSDAG 与解析技术(Refined Process Structure Tree，RPST)、流程模型相似性相结合的技术来检测流程变体簇中的近似克隆(相似片段)。为了同时检测出这两类片段，文献[3]基于结构相似度和行为相似度对流程变体进行划分，进而对流程变体中的片段进行分组，以此来得到精确克隆与近似克隆。文献[4]利用流程变体间的精确克隆与近似克隆对流程模型进行合并，有效避免了对流程模型的无效合并。文献[5]则从事件日志集合中发现层次合并过程模型。为了不局限于只合并流程变体簇，文献[6]将以活动为中心的组织间业务流程合并为一个集成过程。文献[7]则基于 LMP 技术来合并和处理异构模型，从而支持所需的开发活动。但是，在合并模型的过程中可能会产生一些冲突，导致过程模型不一致。因此，文献[8]在研究流程模型合并的同时也对合并过程中可能会遇到的冲突类型进行了研究。文献[9]则提出了一种允许多个高级建模约束的模型合并方法。

Petri 网作为常用的系统建模语言之一，由于具备图形化的建模和坚实的数学基础而受到广泛推崇。其中公平性[10-16]被广泛应用于 Petri 网模型的性质分析，并且这一性质在复杂网系统建模、网系统节点划分、模型合成运算等方面为 Petri 网提供了极大的方便，降低了复杂运算的难度且利于形式化表达。在业务流程管理系统中，以 Petri 网作为建模语言的业务流程，不仅可以形式化模型，而且还可以将系统性质转变成状态方程或关联矩阵来进行分析[17-19]，但相较于状态方程而言，关联矩阵可用于构造更简洁的 Petri 网形式，并可用来证明系统性质。

本文利用现有的研究方法将图 1 中的两个流程变体合并为单一的流程模型。从图中不难发现，现有研究重复考虑了流程变体间具有相同源和汇的片段，这无疑增加了变体合并的工作量；同时，现有的研究方法忽略了流程模型中的行为特征，如图 1 所示，合并后的流程模型产生了原模型中不存在的活动序列，如迹$\{T_0T_1T_2T_6T_4T_5\}$。

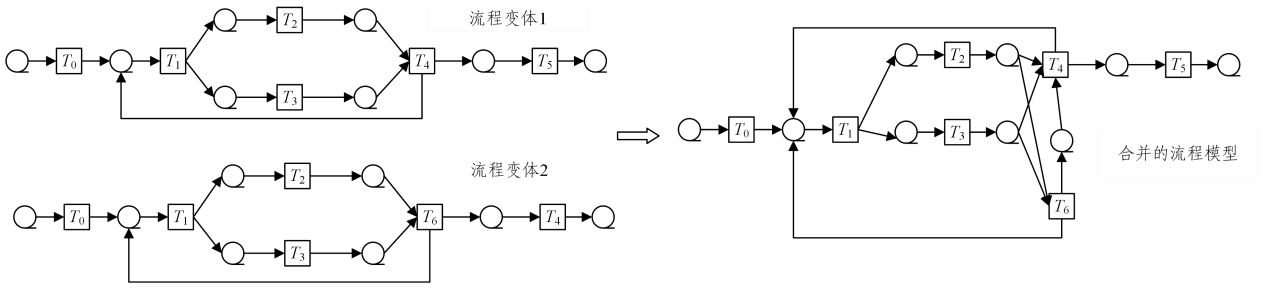


图 1 流程合并图

Fig. 1 Process merge diagram

基于此，本文提出了基于分组公平控制流结构的变体合并方法，用于实现复杂网系统的合并。本文的主要贡献有：1)在不考虑模型相似性的情况下，依据分组公平将流程变体拆分为独立、单一的变体片段；2)依据变体片段，得到该片段的控制流矩阵，通过将控制流矩阵进行对齐，从而实现流程变体的合并；3)防止合并后的流程模型中产生不期望的活动序列。

本文第 2 节给出了公平性和流程变体的基本概念；第 3 节通过引入变体片段和控制流矩阵，设计了基于分组公平控制流结构的流程变体合并算法；第 4 节通过图书馆借还书籍流程验证算法的可行性；最后总结全文并展望未来工作。

2 基本概念

为简便起见，Petri 网的相关定义不在此赘述，具体请参考文献[10]，本节主要介绍本文所涉及的公平性和流程变体的基本概念。

定义 1(公平关系与公平网[10]) 设 $N=(P,T;F)$为一个 Petri 网，$\Sigma_0=(N,M_0)$为初始网系统，$t_1,t_2\in T$。如果存在正整数 k，使得 $\forall M\in R(M_0)$和$\forall\sigma\in T^*$：$M[\sigma\rangle$都有：$\sharp(t_1/\sigma)=0\rightarrow\sharp(t_2/\sigma)\leqslant k$，则称$t_2$公平依赖于$t_1$，其中 $\sharp(t_i/\sigma)$为变迁t_i在变迁序列σ中的发生次数。此时，将Σ_0中公平依赖关系记为 $R(\Sigma_0)$，且$(t_2,t_1)\in R(\Sigma_0)$。若(t_1,t_2)，(t_2,t_1)同时属于 $R(\Sigma_0)$，即称t_1和t_2是Σ_0的一对公平关系变迁，或者称t_1和t_2处于公平关系，记为$(t_1,t_2)\in RF(\Sigma_0)$。

如果Σ_0中任意两个变迁都处于公平关系，则称Σ_0为公平网。

当网系统中只存在有限序列时，这个网始终是公平的。因此，只有当网系统中存在无限序列，才能对网系统的公平性进行分析。一个网系统若是公平的，那么分组也是公平的。为此，在接下来的定义中着重介绍与分组公平相关联的概念。

定义 2(分组公平关系[11]) 设$\Sigma_0=(N,M_0)$为一个网系统，如果存在变迁组$T_1,T_2\in T$且$T_1\cap T_2=\emptyset$，且存在正整数k，对于$\forall M\in R(M_0)$和$\forall\sigma\in T^*$；$M[\sigma\rangle$，使得对任意的t_1和t_2都有$\sum\limits_{t_1\in T}\sharp(t_1/\sigma)=0$，则$\sum\limits_{t_2\in T}\sharp(t_2/\sigma)\leqslant k$，$i,j\in\{1,2\}$且$i\neq j$，则称变迁组$T_1$和$T_2$在网系统中处于分组公平关系。

定义 3(分组公平网[11]) 设$\Sigma_0=(N,M_0)$为一个网系统，如果存在一个公平划分 $T/=T_1\cup T_2\cup\cdots\cup T_m$，则使得：

(1)$T=T_1\cup T_2\cup\cdots\cup T_m$，且$\forall i,j=\{1,2,\cdots,m\}$，若$i\neq j$都有$T_i\cap T_j=\emptyset$；

(2)$\forall i,j=\{1,2,\cdots,m\}$，$i\neq j$，且$T_i$和$T_j$在网系统中处于组公平关系；

则称网系统为关于 $T/$的一个分组公平 Petri 网系统。

定义 4(流程变体[20-21]) 假设 N 是一组流程模型，I 表示一组流程更改操作(即删除、插入、移动)，$S_0,\cdots,S_n,S'\in N$ 是一系列的流程模型。设 σ 是流程更改操作中的任意一个更改

操作，$\theta \in I$ 是一个对初始的流程模型进行更改操作的序列，将满足以下条件的流程模型称为 S 的流程变体：

(1) $S_0[\sigma\rangle S_n$ 表示 σ 作用于 S_0 且 S_n 是 S_0 应用 σ 所得到的流程模型；

(2) $S[\theta\rangle S'$ 表示存在 $S_0, S_1, \cdots, S_n \in N$ 并且 $S=S_0, S'=S_n, S_i[\sigma_i\rangle S_{i+1}, i=\{1,2,\cdots,n-1\}$。

3 流程变体的合并

从流程系统粗粒度的角度分析，流程变体是流程簇模型从基模型中衍生出的其他相似的流程模型。与之相对，从流程系统细粒度的角度分析，流程变体也可以是各模型中具有某些共性同时又具备差异性的模型片段。因此，为了将流程变体进一步片段化，即划分为更小的片段，提出了变体片段的概念。

定义 5(变体片段) 满足下列条件的三元组 $VF=(P,T;CF)$ 称为一个变体片段，其中：

(1) P 中的元素称为库所，T 中的元素称为变迁；

(2) $P \cup T \neq \emptyset$ 且 $P \cap T = \emptyset$；

(3) CF 是片段 VF 的流关系且 $CF \in (P \cup T) \cup (P \cup \{\perp\}) \cup (T \cup \{\perp\})$，符号 $\perp$ 代表缺失源节点或者目标节点的流关系；

(4) $|CF[\{P\}]| \leqslant 1 \wedge |CF^{-1}[\{P\}]| \leqslant 1$，即每个库所最多有一个输出控制流或一个输入控制流。

图 2 所示即为一个变体片段，而变体片段又可以看作是控制流路径的集合。因此，在定义 6 中给出了控制流路径的概念，在定义 7 中给出了控制流矩阵的相关概念。

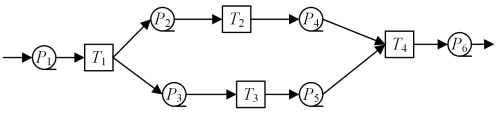


图 2 变体片段 VF

Fig. 2 Variant fragment VF

定义 6(控制流路径) 给定变体片段 VF，控制流路径为一个三元组 $CFP=(O_s, O_t, Seq)$，其中：

(1) $O_s \in P \cup T$，称之为源节点；

(2) $O_t \in P \cup T$，称之为目标节点；

(3) $Seq=\{(O_i, O_{i+1})\}, i \in 1,2,\cdots,n-1$ 是一个交替的控制流，其中 $O_1=O_s, O_n=O_t$，并且 n 是节点在 Seq 中出现的数量。

定义 7(控制流矩阵) 将变体片段映射到控制流矩阵 $\boldsymbol{CPM}=(N, CP, \mu)$，其中：

(1) $\mathcal{N}$ 是变体片段中所代表节点集；

(2) $CP=\Gamma_{VF}$ 是变体片段中控制流路径的集合；

(3) $\mu(a,a')=\{p \mid p \in CP \wedge p.O_s=a \wedge p.O_t=a'\}, (a,a') \in \mathcal{N}^2$，其中 μ 是一个函数，旨在将称为源节点与目标节点的一对节点映射给控制流路径 CP，以此将这两个部分连接起来。

图 2 中变体片段的控制流矩阵如表 1 所列，其中 p_1' 为变体片段中一条控制流路径。从表 1 可以看出，一个变体片段的控制流矩阵只囊括了该片段中所出现的节点。为了对变体片段中的节点进行比较，流程模型中的每个变体片段的控制流矩阵都必须包含同一套节点，因此，本文提出了控制流矩阵对齐的概念，以便合并后的流程模型包含所有的节点。

表 1 变体片段的控制流矩阵

Table 1 Control flow matrix of variant fragment

	P_1	P_2	P_3	P_4	P_5	P_6
P_1		p_1'	p_2'			
P_2				p_3'		
P_3					p_4'	
P_4						p_5'
P_5						p_6'
P_6						

$p_1'=(P_1,P_2,\{(P_1,T_1),(T_1,P_2)\}), p_2'=(P_1,P_3,\{(P_1,T_1),(T_1,P_3)\})$

$p_3'=(P_2,P_4,\{(P_2,T_4),(T_4,P_4)\}), p_4'=(P_3,P_5,\{(P_3,T_3),(T_3,P_5)\})$

$p_5'=(P_4,P_6,\{(P_4,T_4),(T_4,P_6)\}), p_6'=(P_5,P_6,\{(P_5,T_4),(T_4,P_6)\})$

定义 8(控制流矩阵对齐) 给定一组控制流矩阵 $\boldsymbol{CPMs}=(\boldsymbol{CPM}_1, \boldsymbol{CPM}_2, \cdots \boldsymbol{CPM}_m)$，挑选一个控制流矩阵 $\boldsymbol{CPM}_i$ 与矩阵集合中剩余的控制流矩阵进行对齐，记做 $\boldsymbol{CPM}_i^A=(N, CP, \mu)$，其中：

(1) $\boldsymbol{CPM}_i^A N = \bigcup_{j=1,2,\cdots,m} \boldsymbol{CPM}_j N$；

(2) $\boldsymbol{CPM}_i^A \cdot CP = \boldsymbol{CPM}_i \cdot CP$；

(3) 对于 $\forall a_1, a_2 \in \boldsymbol{CPM}_i^A \cdot \mathcal{N}$，则

$$\boldsymbol{CPM}_i^A \cdot \mu(a_1,a_2) = \begin{cases} \boldsymbol{CPM}_i \cdot \mu(a_1,a_2), & (a_1,a_2) \in \boldsymbol{CPM}_i \times \boldsymbol{CPM}_i \\ \emptyset, & \text{otherwise} \end{cases}$$

定义 9(不期望行为) 假设 S_i 为一组流程变体，MS 为合并一组流程变体后的流程模型。设 η 是流程变体中的迹，δ 是合并后流程模型中的迹，对齐这两个模型中的迹：

(1) $Align(\eta,\delta)=0$，对齐两个迹没有产生偏差，即模型间的迹是相同的；

(2) $Align(\eta,\delta) \neq 0$，对齐两个迹产生偏差，即模型间的迹不相同，对于产生偏差的迹集合 $\delta-\eta$，称之为不期望行为。

3.1 变体片段提取算法

与完整的流程变体相比，变体片段可以刻画没有指定输入或输出的活动，如图 2 所示。并且变体片段在结构上要远小于完整的流程变体，这对于组织管理者来说，更容易理解。因此，本文提出了一个提取变体片段的算法，实现了从完整流程变体中提取变体片段的功能，其伪代码如算法 1 所示。

算法 1 提取变体片段

```
输入：流程变体Si、流程变体Si的变迁集 t
输出：变体片段 VF
1. i←1
2. FOR i to n DO
3.     FOR 1 to | Si · t| DO
4.       σp←Find Transition Sequence(Si)
5.       Ax←CountNumber(tx,σp)
6.       Ay←CountNumber(ty,σp)
7.       IF Ax=0 ∧ Ay≤k then
8.         (ty,tx)∈R(Si)
9.           IF Ay=0 ∧ Ax≤k THEN
10.            (tx,ty)∈R(Si)
11.            tx and ty are in a fair relation
12.      END IF
13. END IF
14. END FOR
15.    IF ∀ tx and ty are in a fair relation THEN
```

```
16.       Si is a fair Petri net
17.   ELSE
18.       T/←Divide(t,m)
19.   FOR 1 to m DO
20.           IF tx,ty∈Tj,tz∈T−Tj THEN
21.         (tx,ty)∈RF(Si),(tx,tz)∉RF(Si)
22.     IF Vaule(T/)="true"∧∀(Tj,T−Tj)∈RF(Sl)THEN
23.           Si is group-fair Petri net
24.         END IF
25.       END IF
26.     END FOR
27.   END IF
28.   VF←map(T/)
29. END FOR
30. Return VF
```

算法 1 将变体片段从流程变体中提取出来，下面对算法 1 中的步骤做进一步的分析。算法 1 中第 3－16 行是判断流程变体S_i是否为一个公平网，其中第 4 行是在流程变体中寻找一个变迁序列σ_p；第 5－6 行为计算变迁t_x和t_y在变迁序列σ_p中出现的次数；第 7－14 行是对变迁t_x和t_y的公平性进行判断；第 15－16 行则根据任意变迁t_x和t_y的公平性来判断流程变体是否为公平网。如果流程变体不是一个公平网，则继续执行第 18－27 行的步骤，其中第 18 行将流程变体中的变迁划分为 m 块，以此得到变迁组；第 19－22 行是对变迁组中变迁的公平性进行判断；第 23－27 行是判断流程变体是否为分组公平 Petri 网。第 28 行将公平分支正确的变迁组映射给变体片段。第 30 行返回变体片段的值。

3.2 将变体片段构建成控制流矩阵

将一个变体片段转变成一个控制流矩阵，首先是提取变体片段的控制流路径，而控制流路径则定义了两个节点之间的交替序列，使得一个路径的源是前一个路径的目标；其次，将控制流路径进行处理得到控制流矩阵。下面通过算法 2 来具体描述控制流矩阵的构建。

算法 2 构建控制流矩阵

输入：变体片段VF_i

输出：控制流矩阵 **CFM**

```
1. CFP={},CFM={}
2. i←1
3. FOR i to n DO
4.    CFP←{(Os,Ot,⟨Os,Ot⟩)|(Os,Ot)∈CF}
5.    FOR all(Os,Ot,Seq)DO
6.      IF CF⁻¹{Os}≠∅ then
7.              FOR all N∈CF⁻¹[{Os}]∧¬Seq.object f(O) DO
8.                IF(O,Ot,(O,Os)::Seq)∉CFP then
9.                  CFP←CFP∪(O,Ot,(O,Os)::Seq)
10.             END IF
11.           END FOR
12.   END IF
13.   IF CF{Ot}≠∅ then
14.       FOR all N∈CF[{Ot}]∧¬Seq.object f(O) DO
15.         IF(Os,O,Seq::(Ot,O))∉CFP then
16.             CFP←CFP∪(Os,O,Seq::(Ot,O))
17.           END IF
18.         END FOR
19.       END IF
20. END FOR
21. Return CFP
22. CFM←map(CFP)
23. Return CFM
```

算法 2 通过从变体片段中提取控制流路径，再将控制流路径映射到矩阵中，从而得到变体片段的控制流矩阵。算法 2 中第 1 行是创建一个控制流路径和控制流矩阵的空集。第 3－20 行是从变体片段中提取该片段的控制流路径。其中第 4 行是生成由单个控制流构成的控制流路径，并将其置放到预先定义好的集合 CFP 中；第 5 行为一个循环条件，循环的是所有的单个控制流路径；第 6－19 行是通过在变体片段中的控制流集来扩展已提取的控制流，以此来搜索由多个控制流组成的控制流路径。其中，第 6－12 行表示为一个节点存在源节点，将此源节点所扩展到该节点所代表的单个控制流路径中，以此得到一个新的控制流路径；第 13－19 行是与之相反的情况，用来表示一个节点存在目标节点。算法 2 中的第 8 行与第 15 行的符号::用以表示将节点插入到路径的头部或者尾部。同时，算法 2 中的第 7 行和第 14 行则是为了检测在扩展控制流路径的过程中源节点和目标节点是否已经存在于控制流路径的情况，以此来减少循环的次数。算法 2 中第 21 行是返回控制流路径，并在第 22 行中将其映射到矩阵中，从而得到变体片段的控制流矩阵。

经由算法 2 将变体片段构建成控制流矩阵，这种控制流矩阵是矩阵的一种特殊表达形式，那么我们将依据矩阵的相关属性对控制流矩阵的相关性质进行进一步的分析。

定理 1（行属性） 给定控制流矩阵 $\boldsymbol{CPM}=(N,CP,\mu)$，$\forall a_i,a_j,a_k\in N,\mu(a_i,a_j)\neq\emptyset,\mu(a_i,a_k)\neq\emptyset$ 并且存在 $\mu(a_i,a_j)\neq\emptyset,\mu(a_i,a_k)\neq\emptyset$ 两条控制流路径 p_1,p_2，使得 $p_1\in\mu(a_i,a_j)$，$p_2\in\mu(a_i,a_k)$，则 $p_1.seq.first=p_2.seq.first$。

证明：假设a_1,a_2,a_3是 N 中的 3 个节点，并且在这 3 个节点中存在两条控制流路径 $p_1=\mu(a_1,a_2)$，$p_2=\mu(a_1,a_3)$，且 $p_1\neq p_2$。依据定义 8，p_1和p_2是分别描述节点对(a_1,a_2)与(a_1,a_3)的控制流路径。在控制流路径p_1中，$o_1=a_1,o_n=a_2$，在控制流路径p_2中，$o_1'=a_1,o_n'=a_3$。

现假设 $p_1.seq.first\neq p_2.seq.first$，即 $(o_1,o_2)\neq(o_1',o_2')$。因为在两条控制流路径中$o_1=o_1'=a_1$，那么$o_2\neq o_2'$。因此，节点a_1至少包含了o_2和o_2'两个输出流，即$|CF[\{a_1\}]|\geqslant 2$，与定义 6 矛盾，因而假设不成立，定理 1 得证。

定理 2（列属性） 给定控制流矩阵 $\boldsymbol{CPM}=(N,CP,\mu)$，$\forall a_i,a_j,a_k\in N,\mu(a_j,a_i)\neq\emptyset,\mu(a_k,a_i)\neq\emptyset$，并且存在两条控制流路径$p_1,p_2$，使得 $p_1\in\mu(a_j,a_i)$，$p_2\in(a_k,a_i)$，则 $p_1.seq.last=p_2.seq.last$。

证明：假设a_1,a_2,a_3是 N 中的 3 个节点，并且在这 3 个节点中存在两条控制流路径 $p_1=\mu(a_1,a_2)$，$p_2=\mu(a_3,a_2)$，且 $p_1\neq p_2$。依据定义 8，p_1和p_2是分别描述节点对(a_1,a_2)与(a_3,a_2)的控制流路径。在控制流路径p_1中，$o_1=a_1,o_n=a_2$，在控制流路径p_2中，$o_1'=a_3,o_n'=a_2$。

现假设 $p_1.seq.last \neq p_2.seq.last$，即 $(o_{n-1}, o_n) \neq (o'_{n-1}, o_n')$。因为在两条控制流路径中 $o_n = o_n' = a_2$，那么 $o_{n-1} \neq o'_{n-1}$。因此，节点 a_2 至少包含了 o_{n-1} 和 o'_{n-1} 两条输入流，即 $|CF^{-1}[\{a_2\}]| \geqslant 2$，与定义 6 矛盾，因而假设不成立，定理 2 得证。

因此，对于控制流矩阵中给定的行节点而言，一个列节点内的所有控制流路径抑或是不同列节点的控制流路径都共享来自给定的行节点的输出控制流，即存在许多控制流路径从行节点转移到同一列节点或不同列节点；同样地，对于控制流矩阵中给定的列节点而言，一个行节点内的所有控制流路径抑或是不同行节点的控制流路径都共享给定的列节点的输入控制流，即存在许多控制流路径从同一行节点或不同行节点转移到列节点。

由表 1 可以看出，控制流路径 p_1' 和 p_2' 位于同一行、p_5' 和 p_6' 位于同一列，并且 $p_1'.seq.first = p_2'.seq.first$，$p_5'.seq.last = p_6'.seq.last$，可以确定表 1 中的控制流矩阵满足定理 1 与定理 2。

3.3 流程变体合并

将控制流矩阵中的节点对齐后，就可以将其合并为一个单独的控制流矩阵。算法 3 详细地描述了控制流矩阵的对齐与合并。

算法 3 流程变体合并方法

输入：控制流矩阵集

输出：合并的控制流矩阵

```
1. ACFM_i={},MCFM={}
2. i←1,j←1
3. FOR i to m DO
4.     CFM_i←Select(CFMs)
5.     FOR j to m DO
6.       IF CFM_i≠CFM_j THEN
7.         ACFM_i.N←Union(CFM_i.N,CFM_j.N)
8.         IF ∀(a_x,a_y)∈CFM_i THEN
9.           ACFM_i.CP←CFM_i.CP
10.          ACFM_i.μ(a_x,a_y)←CFM_i.μ(a_x,a_y)
11.        ELSE
12.          ACFM_i.μ(a_x,a_y)←∅
13.        END IF
14.      END IF
15.    END FOR
16.    ACFM_i←(ACFM_i.N,ACFM_i.CP,ACFM_i.μ(a_x,a_y))
17. END FOR
18. MCFM.N←ACFM_i.N
19. MCFM.CP←Union(ACFM_i.CP)
20. MCFM.μ←Union(ACFM_i.μ)
21. MCFM←(MCFM.N,MCFM.CP,MCFM.μ)
22. Return MCFM
```

通过算法 3 完成对控制流矩阵中节点的对齐以及合并，下面将详细地对算法 3 中的步骤进行描述。算法 3 中第 1 行首先创建两个为空集的 $\boldsymbol{ACFM}_i$ 和 $\boldsymbol{MCFM}$。第 3－17 为行为控制流矩阵对齐的过程，其中第 4 行是在控制流矩阵集中选择一个控制流矩阵；为了减少对齐过程中出现控制流矩阵与自身进行对齐，在第 6 行添加了一个判断条件；第 7 行将选中的控制流矩阵 $\boldsymbol{CFM}_i$ 中的节点与剩余控制流矩阵中的节点进行并集操作；第 8 行是将选中的控制流矩阵中的路径映射到 $\boldsymbol{ACFM}_i$ 中；第 8－13 行是为对齐的控制流矩阵中行与列的元素做出判断，如果对齐的控制流矩阵中的节点对都属于原先选中的控制流矩阵，则其中的元素不为空，否则就为空集。第 18－21 行是将对齐的控制流矩阵进行合并。第 22 行是返回合并的控制流矩阵。

从某种意义上来说，一组控制流矩阵的合并就是将其中的节点集扩展到另一些与之对应的节点集上。但是，合并过程还会将控制流矩阵中的控制流路径与其他控制流矩阵中控制流路径进行扩展，其扩展根据的是后者的源节点（目标节点）与前一个控制流路径的目标节点（源节点）相同的原则。因此，合并控制流矩阵后所得到的控制流矩阵还需要满足行属性与列属性，即定理 1 和定理 2。如若合并后的控制流矩阵不满足行属性，则需要在源节点和后续节点之间插入一个可配置的发散网关，其标签是控制流路径的源节点；同样地，如若合并后的控制流矩阵不满足列属性，则需要在目标节点与前继节点之间插入一个可配置的聚合网关。

3.4 不期望行为的检测

将流程变体簇合并为一个单一的流程模型容易产生不期望行为。算法 4 阐述了检测合并模型中的不期望行为的详细步骤。

算法 4 不期望行为检测方法

输入：控制流矩阵、流程变体中的迹、合并模型中的迹

输出：不期望行为

```
1. IF CFM.μ≠∅ THEN
2.     CFM.μ=1
3. ELSE
4.     CFM.μ=0
5. END IF
6. i=1
7. FOR i TO j
8.     CFM⁻←CFM_j−CFM_i
9. END FOR
10. IF CFM⁻.μ≠0 THEN
11.    IF a∈AMCFM 且|CF⁻¹[{a}]|≥2 THEN
12.      IF Align(MCFM(a),AMCFM(a))≠0 THEN
13.        Find undesired behaviors
14.      ELSE
15.        No undesired behaviors
16.      END IF
17.    END IF
18. END IF
19. Return undesired behaviors
```

算法 4 中的第 1－5 行是对控制流矩阵中的控制流路径进行数值化。第 6－9 行是对数值化以后的控制流矩阵进行一个简单的矩阵减运算。第 10－18 行为不期望行为的检测过程，如果合并的控制流矩阵与实际流程模型产生的控制流矩阵相

减后控制流路径的数值不为 0，则说明合并模型中存在不期望行为；算法 4 中的第 11 行是在由合并模型产生的控制流矩阵中，寻找至少包含两条输入流的活动节点；第 12－17 行将流程变体中包含此类活动节点的迹与合并模型中包含此类活动节点的迹进行对齐，并通过比对两条迹是否产生偏差，从而寻找到合并模型中的不期望行为。若对齐流程变体中的迹与合并模型中的迹产生偏差，则合并模型中的迹就是不期望行为，反之，则不是不期望行为。

4 案例研究

为了更好地描述流程变体的合并方法，本文通过图书馆借还书籍的业务流程来进行阐述，此业务流程囊括了图书馆借还书籍的 3 种流程变体，即两种线下图书馆借还书籍的流程模型和一种线上图书馆借还书籍的流程模型，分别为 S_1（见图 3(a)）、S_2（见图 3(b)）、S_3（见图 3(c)）。

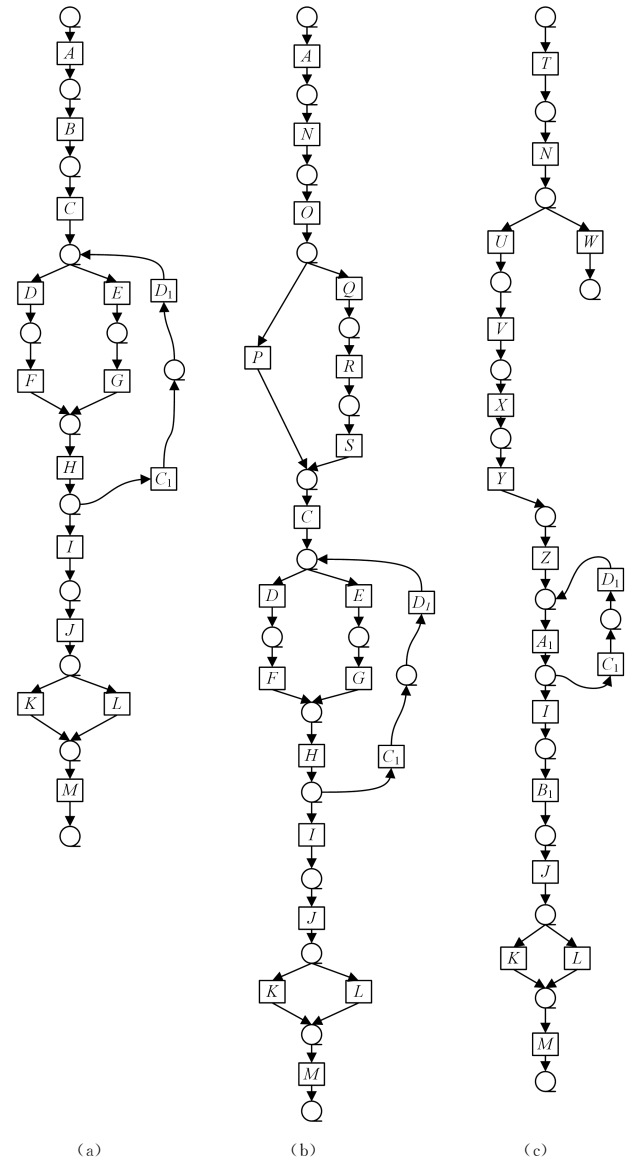


图 3 图书馆借还书籍流程变体模型图

Fig. 3 Model diagram of library loan and return books process variants

现对图 3 中流程模型的变迁做如下说明：A 表示读者刷卡入馆，B 表示无目的性的借书，C 表示到借阅区浏览，挑选所需图书，D 表示自助机借书，E 表示服务台借书，F 表示刷借书证，核对相关信息，G 表示所在楼层服务台办理借书手续，H 表示借书成功，刷卡出馆，I 表示阅读完毕，J 表示还书前核实，K 表示信息正常，L 表示超期罚款，M 表示还书成功，N 表示指定书籍借阅，O 表示检索馆藏，P 表示书籍在架，Q 表示书籍不在架，R 表示预约指定书籍，S 表示预约书籍到馆，T 表示登录网上图书馆网址，U 表示网上办理委托申请，V 表示委托申请办理中，W 表示取消委托申请，X 表示委托申请成功，Y 表示核实地址，Z 表示书籍运送，A_1 表示借书成功，B_1 表示将书寄回，C_1 表示未阅读完毕，D_1 表示进行续借。

本文以图书馆借还书籍业务流程作为研究案例，以此来验证文中所提出的 4 个算法，并实现流程变体的合并工作，图 3 中流程变体的迹具体如表 2 所列。

表 2 流程变体中的迹

Table 2 Traces in process variants

变体	迹 η
变体 1	$\langle ABCDFHIJKM\rangle$，$\langle ABCEGHIJKM\rangle$，$\langle ABCDFHIJLM\rangle$ $\langle ABCEGHIJLM\rangle$，$\langle ABCEGHC_1D_1EGHIJKM\rangle$ $\langle ABCDFHC_1D_1DFHIJKM\rangle$，$\langle ABCEGHC_1D_1EGHIJLM\rangle$ $\langle ABCDFHC_1D_1DFHIJLM\rangle$
变体 2	$\langle ANOQRSCDFHIJKM\rangle$，$\langle ANOQRSCEGHIJKM\rangle$ $\langle ANOQRSCEGHIJLM\rangle$，$\langle ANOPCDFHC_1D_1DFHIJKM\rangle$ $\langle ANOPCEGHC_1D_1EGHIJKM\rangle$，$\langle ANOPCEGHIJKM\rangle$ $\langle ANOPCDFHC_1D_1DFHIJLM\rangle$，$\langle ANOPCDFHIJKM\rangle$ $\langle ANOPCDFHIJLM\rangle$，$\langle ANOPCEGHIJLM\rangle$ $\langle ANOQRSCDFHIJLM\rangle$，$\langle ANOQRSCEGHC_1D_1EGHIJLM\rangle$ $\langle ANOQRSCDFHC_1D_1DFHIJLM\rangle$ $\langle ANOQRSCEGHC_1D_1EGHIJKM\rangle$ $\langle ANOPCEGHC_1D_1EGHIJLM\rangle$
变体 3	$\langle TNW\rangle$，$\langle TNUVXYZA_1IB_1JKM\rangle$，$\langle TNUVXYZA_1IB_1JLM\rangle$ $\langle TNUVXYZA_1C_1D_1A_1IB_1JKM\rangle$ $\langle TNUVXYZA_1C_1D_1A_1IB_1JLM\rangle$

图 3 中的 3 个流程模型存在无限的活动发生序列，因此执行算法 1 中的第 3－16 行，得出图 3 中所陈列的 3 个图书馆借还书籍业务流程不是公平网的结论。

之后执行算法 1 中的第 18 行，得到 m 块的变迁组，并在表 3 中将变迁公平分支结果列出来。

表 3 流程变体变迁公平分支结果

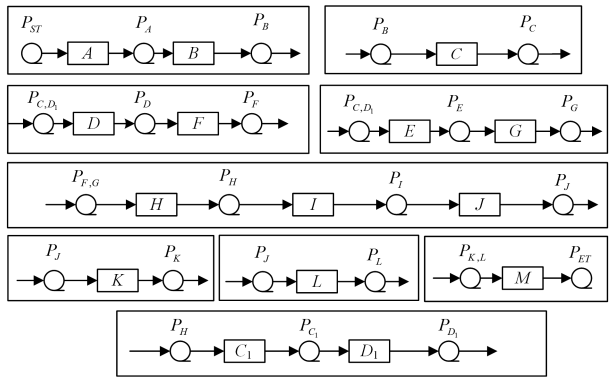
Table 3 Fair branch results of process variants

流程变体	公平划分
S_1	$T_1\cup T_2\cup T_3\cup T_4\cup T_5\cup T_6\cup T_7\cup T_8\cup T_9$
S_2	$T_1\cup T_2\cup T_3\cup T_4\cup T_5\cup T_6\cup T_7\cup T_8\cup T_9\cup T_{10}\cup T_{11}$
S_3	$T_1\cup T_2\cup T_3\cup T_4\cup T_5\cup T_6\cup T_7$

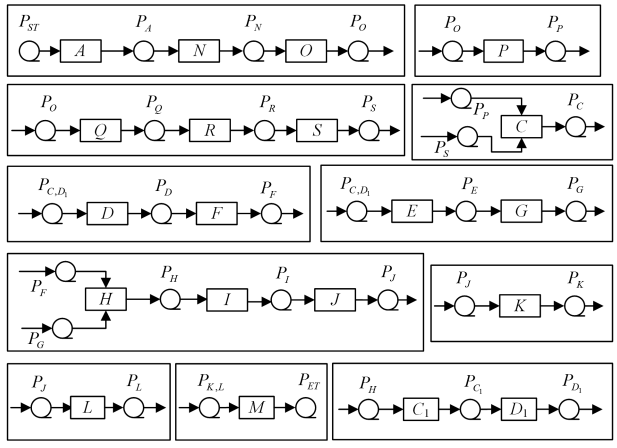
流程变体 S_1 中公平分支所得到的变迁组中包含的变迁具体如下：$T_1=\langle\{A\},\{B\}\rangle$，$T_2=\langle\{C\}\rangle$，$T_3=\langle\{D\},\{F\}\rangle$，$T_4=\langle\{E\},\{G\}\rangle$，$T_5=\langle\{H\},\{I\},\{J\}\rangle$，$T_6=\langle\{K\}\rangle$，$T_7=\langle\{L\}\rangle$，$T_8=\langle\{M\}\rangle$，$T_9=\langle\{C_1\},\{D_1\}\rangle$。流程变体 S_2 中公平分支所得到的变迁组中包含的变迁具体如下：$T_1=\langle\{A\},\{N\},\{O\}\rangle$，$T_2=\langle\{P\}\rangle$，$T_3=\langle\{Q\},\{R\},\{S\}\rangle$，$T_4=\langle\{C\}\rangle$，$T_5=\langle\{D\},\{F\}\rangle$，$T_6=\langle\{E\},\{G\}\rangle$，$T_7=\langle\{H\},\{I\},\{J\}\rangle$，$T_8=\langle\{K\}\rangle$，$T_9=\langle\{L\}\rangle$，$T_{10}=\langle\{M\}\rangle$，$T_{11}=\langle\{C_1\},\{D_1\}\rangle$。流程变体 S_3 中公

平分支所得到的变迁组中包含的变迁具体如下：$T_1=\langle\{T\},\{N\}\rangle$，$T_2=\langle\{W\}\rangle$，$T_3=\langle\{U\},\{V\},\{X\},\{Y\},\{Z\},\{A_1\},\{I\},\{B_1\},\{J\}\rangle$，$T_4=\langle\{K\}\rangle$，$T_5=\langle\{L\}\rangle$，$T_6=\langle\{M\}\rangle$，$T_7=\langle\{C_1\},\{D_1\}\rangle$。

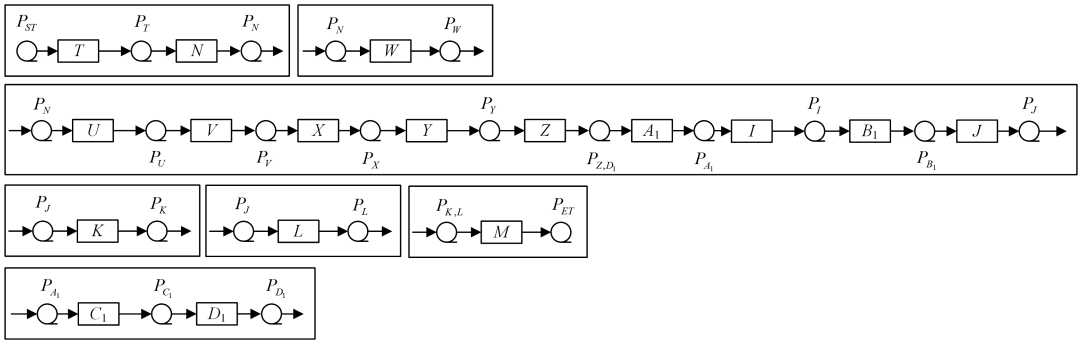
流程变体中的变迁公平分支后，执行算法 1 中的第 19—27 行，得出 3 个流程变体均为分组公平 Petri 网。3 个流程变体经过算法 1 中的上述步骤，共得到了 23 个变迁组。之后执行算法 1 的第 28 行，将以 T/为分支方式的流程变体映射给变体片段，并在图 4 中展示了各流程变体的变体片段。



(a)流程变体S_1的变体片段



(b)流程变体S_2的变体片段



(c) 流程变体S_3的变体片段

图 4 流程变体S_i的变体片段

Fig. 4 Variant fragment of process variant

从图 4 可以看出，图书馆借还书籍流程的 3 个流程变体中都包含了 3 组完全相同的变体片段$VF_{6,8,4}^{1,2,3}=\langle K\rangle$，$VF_{7,9,5}^{1,2,3}=\langle L\rangle$，$VF_{8,10,6}^{1,2,3}=\langle M\rangle$，其中，上标的数字代表变体簇中流程变体的个数，而下标的数字表示在流程变体中所代表的变体片段顺序。同时，这 3 组变体片段拥有完全相同的源节点与汇节点，即变体片段$VF_{6,8,4}^{1,2,3}$，$VF_{7,9,5}^{1,2,3}$，$VF_{8,10,6}^{1,2,3}$的源节点和汇节点分别为P_J和P_K、P_J和P_L、$P_{K,L}$和P_{ET}。基于此，在变体片段控制流路径的提取、控制流矩阵的构建及合并这 3 个步骤中将不考虑流程变体中的这 3 组变体片段，而是直接将其置放到合并后的流程模型中。

图 4 中各流程变体的变体片段($VF_{6,8,4}^{1,2,3}$，$VF_{7,9,5}^{1,2,3}$，$VF_{8,10,6}^{1,2,3}$除外)作为算法 2 的输入，以此来提取变体片段的控制流矩阵。将以流程变体S_1提取出的变体片段阐述接下来的算法步骤。首先，执行算法 2 中的第 1 行，创建一个为空集的控制流路径和控制流矩阵的集合；然后执行算法 2 中的第 4 行，生成变迁间的单个控制流路径，每个变体片段中的单个控制流路径将在表 4 中列出；同时，将其置放到集合 CFP 中，即对于变体片段VF_1来说，$CFP=((P_{ST},A,\{(P_{ST},A)\}),(A,P_A,\{(A,$

P_A)}),(P_A,B,{(P_A,B)}),(B,P_B,{(B,P_B)}))。

表 4 变体片段的单个控制流路径

Table 4 Single control flow path for variant fragments

变体片段VF_i^j	单个控制流路径
VF_1^1	(P_{ST},A,{(P_{ST},A)})
	(A,P_A,{(A,P_A)})
	(P_A,B,{(P_A,B)})
	(B,P_B,{(B,P_B)})
VF_2^1	(P_B,C,{(P_B,C)})
	(C,P_C,{(C,P_C)})
VF_3^1	(P_C,D,{(P_C,D)})
	(P_{D_1},D,{(P_{D_1},D)})
	(D,P_D,{(D,P_D)})
	(P_D,F,{(P_D,F)})
	(F,P_F,{(F,P_F)})
VF_4^1	(P_C,E,{(P_C,E)})
	(P_{D_1},E,{(P_{D_1},E)})
	(E,P_E,{(E,P_E)})
	(P_E,G,{(P_E,G)})
	(G,P_G,{(G,P_G)})
VF_5^1	(P_F,H,{(P_F,H)})
	(P_G,H,{(P_G,H)})
	(H,P_H,{(H,P_H)})
	(P_H,I,{(P_H,I)})
	(I,P_J,{(I,P_J)})
	(P_I,J,{(P_I,J)})
	(J,P_J,{(I,P_J)})
VF_9^1	(P_H,C_1,{(P_H,C_1)})
	(C_1,P_{C_1}{(C_1,P_{C_1})})
	(P_{C_1},D_1,{(P_{C_1},D_1)})
	(D_1,P_{D_1},{(D_1,P_{D_1})})

执行算法 2 中第 6－19 行，将变体片段中由单个控制流路径构成的 CFP 扩展为由多个控制流路径组成。流程变体 S_1 中的变体片段扩展后的控制流路径如表 5 所列。而对于已经出现在集合 CFP 中的控制流路径，在扩展时不会对其进行考虑，即算法 2 中的第 7 行与第 14 行。

表 5 流程变体S_1中变体片段的控制流路径

Table 5 Control flow paths of variant fragments in process variant S_1

VF_i^j	控制流路径
VF_1^1	(P_{ST},P_B,{(P_{ST},A),(A,P_A),(P_A,B),(B,P_B)})
VF_2^1	(P_B,P_C,{(P_B,C),(C,P_C)})
VF_3^1	(P_C,P_F,{(P_C,D),(D,P_D),(P_D,F),(F,P_F)})
	(P_{D_1},P_F,{(P_{D_1},D),(D,P_D),(P_D,F),(F,P_F)})
VF_4^1	(P_C,P_G,{(P_C,E),(E,P_E),(P_E,G),(G,P_G)})
	(P_{D_1},P_G,{(P_{D_1},E),(E,P_E),(P_E,G),(G,P_G)})
VF_5^1	(P_F,P_J,{(P_F,H),(H,P_H),(P_H,I),(I,P_I),(P_I,J),(I,P_J)})
	(P_G,P_J,{(P_G,H),(H,P_H),(P_H,I),(I,P_I),(P_I,J),(I,P_J)})
VF_9^1	(P_H,P_{D_1},{(P_H,C_1),(C_1,P_{C_1}),(P_{C_1},D_1),(D_1,P_{D_1})})

对于剩下的两个流程变体中的变体片段也通过相同的方法得到变体片段扩展后的控制流路径，分别在表 6 和表 7 中列出了这两个变体的控制流路径。

表 6 流程变体S_2中变体片段的控制流路径

Table 6 Control flow paths of variant fragments in process variant S_2

VF_i^j	控制流路径
VF_1^2	(P_{ST},P_O,{(P_{ST},A),(A,P_A),(P_A,N),(N,P_N),(P_N,O),(O,P_O)})
VF_2^2	(P_O,P_P,{(P_O,P),(P,P_P)})
VF_3^2	(P_O,P_S,{(P_O,Q),(Q,P_Q),(P_Q,R),(R,P_R),(P_R,S),(S,P_S)})
VF_4^2	(P_P,P_C,{(P_P,C),(C,P_C)})
	(P_S,P_C,{(P_S,C),(C,P_C)})
VF_5^2	(P_C,P_F,{(P_C,D),(C,P_C),(P_D,F),(F,P_F)})
	(P_{D_1},P_F,{(P_{D_1},D),(D,P_D),(P_D,F),(F,P_F)})
VF_6^2	(P_C,P_G,{(P_C,E),(E,P_E),(P_E,G),(G,P_G)})
	(P_{D_1},P_G,{(P_{D_1},E),(E,P_E),(P_E,G),(G,P_G)})
VF_7^2	(P_F,P_J,{(P_F,H),(H,P_H),(P_H,I),(I,P_I),(P_I,J),(I,P_J)})
	(P_G,P_J,{(P_G,H),(H,P_H),(P_H,I),(I,P_I),(P_I,J),(I,P_J)})
VF_{11}^2	(P_H,P_{D_1},{(P_H,C_1),(C_1,P_{C_1}),(P_{C_1},D_1),(D_1,P_{D_1})})

表 7 流程变体S_3中变体片段的控制流路径

Table 7 Control flow paths of variant fragments in process variant S_3

VF_i^j	控制流路径
VF_1^3	(P_{ST},P_N,{(P_{ST},T),(T,P_T),(P_T,N),(N,P_N)})
VF_2^3	(P_N,P_W,{(P_N,W),(W,P_W)})
VF_3^3	(P_N,P_J,{(P_N,U),(U,P_U),(P_U,V),(V,P_V),(P_V,X),(X,P_X),(P_X,Y),(Y,P_Y),(P_Y,Z),(Z,P_{D_1}),(P_{D_1},A_1),(A_1,P_{A_1}),(P_{A_1},I),(I,P_I),(P_I,B_1),(B_1,P_{B_1}),(P_{B_1},J),(J,P_J)})
	(P_N,P_J,{(P_N,U),(U,P_U),(P_U,V),(V,P_V),(P_V,X),(X,P_X),(P_X,Y),(Y,P_Y),(P_Y,Z),(Z,P_Z),(P_Z,A_1),(A_1,P_{A_1}),(P_{A_1},I),(I,P_I),(P_I,B_1),(B_1,P_{B_1}),(P_{B_1},J),(J,P_J)})
VF_7^3	(P_{A_1},P_{D_1},{(P_{A_1},C_1),(C_1,P_{C_1}),(P_{C_1},D_1),(D_1,P_{D_1})})

之后根据算法 2 中的第 22 行，将控制流路径映射到控制流矩阵中，以此来得到各变体片段的控制流矩阵，具体情况如表 8 所列。

表 8 变体片段的控制流矩阵

Table 8 Control flow matrix for variant fragments

(a)变体片段VF_1^1

	P_{ST}	P_A	P_B
P_{ST}		p_1	
P_A			p_2
P_B			

p_1=(P_{ST},P_A,{(P_{ST},A),(A,P_A)})

p_2=(P_A,P_B,{(P_A,B),(B,P_B)})

(b)变体片段VF_2^1

	P_B	P_C
P_B		p_3
P_C		

p_3=(P_B,P_C,{(P_B,C),(C,P_C)})

(c)变体片段VF_3^1,VF_5^2

	P_C	P_D	P_F	P_{D_1}
P_C		p_4		
P_D			p_5	
P_F				
P_{D_1}		p_6		

p_4=(P_C,P_D,{(P_C,D),(D,P_D)})

p_5=(P_D,P_F,{(P_D,F),(F,P_F)})

p_6=(P_{D_1},P_F,{(P_{D_1},D),(D,P_D)})

(d)变体片段VF_4^1, VF_6^2

	P_C	P_E	P_G	P_{D_1}
P_C		p_7		
P_E			p_8	
P_G				
P_{D_1}		p_9		

$p_7=(P_C,P_E,\{(P_C,E),(E,P_E)\})$
$p_8=(P_E,P_G,\{(P_E,G),(G,P_G)\})$
$p_9=(P_{D_1},P_E,\{(P_{D_1},E),(E,P_E)\})$

(e)变体片段VF_5^1, VF_7^2

	P_F	P_G	P_H	P_I	P_J
P_F			p_{10}		
P_G			p_{11}		
P_H				p_{12}	
P_I					p_{13}
P_J					

$p_{10}=(P_F,P_H,\{(P_F,H),(H,P_H)\}), p_{11}=(P_G,P_H,\{(P_G,H),(H,P_H)\})$
$p_{12}=(P_H,P_I,\{(P_H,I),(I,P_I)\}), p_{13}=(P_I,P_J,\{(P_I,J),(I,P_J)\})$

(f)变体片段VF_9^1, VF_{11}^2

	P_H	P_{C_1}	P_{D_1}
P_H		p_{14}	
P_{C_1}			p_{15}
P_{D_1}			

$p_{14}=(P_H,P_{C_1},\{(P_H,C_1),(C_1,P_{C_1})\})$
$p_{15}=(P_{C_1},P_{D_1},\{(P_{C_1},D_1),(D_1,P_{D_1})\})$

(g)变体片段VF_1^2

	P_{ST}	P_A	P_N	P_O
P_{ST}		p_1		
P_A			p_{16}	
P_N				p_{17}
P_O				

$p_1=(P_{ST},P_A,\{(P_{ST},A),(A,P_A)\})$
$p_{16}=(P_A,P_N,\{(P_A,N),(N,P_N)\})$
$p_{17}=(P_N,P_O,\{(P_N,O),(O,P_O)\})$

(h)变体片段VF_2^2

	P_O	P_P
P_O		p_{18}
P_P		

$p_{18}=(P_O,P_P,\{(P_O,P),(P,P_P)\})$

(i)变体片段VF_3^2

	P_O	P_Q	P_R	P_S
P_O		p_{19}		
P_Q			p_{20}	
P_R				p_{21}
P_S				

$p_{19}=(P_O,P_Q,\{(P_O,Q),(Q,P_Q)\})$
$p_{20}=(P_Q,P_R,\{(P_Q,R),(R,P_R)\})$
$p_{21}=(P_R,P_S,\{(P_R,S),(S,P_S)\})$

(j)变体片段VF_4^2

	P_P	P_S	P_C
P_P			p_{22}
P_S			p_{23}
P_C			

$p_{22}=(P_P,P_C,\{(P_P,C),(C,P_C)\})$
$p_{23}=(P_S,P_C,\{(P_S,C),(C,P_C)\})$

(k)变体片段VF_1^3

	P_{ST}	P_T	P_N
P_{ST}		p_{24}	
P_T			p_{25}
P_N			

$p_{24}=(P_{ST},P_T,\{(P_{ST},T),(T,P_T)\})$
$p_{25}=(P_T,P_N,\{(P_T,N),(N,P_N)\})$

(l)变体片段VF_2^3

	P_N	P_W
P_N		p_{26}
P_W		

$p_{26}=(P_N,P_W,\{(P_N,W),(W,P_W)\})$

(m)变体片段VF_3^3

	P_N	P_U	P_V	P_X	P_Y	P_Z	P_{A_1}	P_I	P_{B_1}	P_J	P_{D_1}
P_N		p_{27}									
P_U			p_{28}								
P_V				p_{29}							
P_X					p_{30}						
P_Y						p_{31}					p_{36}
P_Z							p_{32}				
P_{A_1}								p_{33}			
P_I									p_{34}		
P_{B_1}										p_{35}	
P_J											
P_{D_1}											

$p_{27}=(P_N,P_U,\{(P_N,U),(U,P_U)\}), p_{28}=(P_U,P_V,\{(P_U,V),(V,P_V)\})$
$p_{29}=(P_V,P_X,\{(P_V,X),(X,P_X)\}), p_{30}=(P_X,P_Y,\{(P_X,Y),(Y,P_Y)\})$
$p_{31}=(P_Y,P_Z,\{(P_Y,Z),(Z,P_Z)\}), p_{32}=(P_Z,P_{A_1},\{(P_Z,A_1),(A_1,P_{A_1})\})$
$p_{33}=(P_{A_1},P_I,\{(P_{A_1},I),(I,P_I)\}), p_{34}=(P_I,P_{B_1},\{(P_I,B_1),(B_1,P_{B_1})\})$
$p_{35}=(P_{B_1},P_J,\{(P_{B_1},J),(J,P_J)\}), p_{36}=(p_Y,p_{D_1},\{(p_Y,Z),(Z,p_{D_1})\})$

(n)变体片段VF_7^3

	P_{A_1}	P_{C_1}	P_{D_1}
P_{A_1}		p_{37}	
P_{C_1}			p_{15}
P_{D_1}			

$p_{37}=(P_H,P_{C_1},\{(P_H,C_1),(C_1,P_{C_1})\})$
$p_{15}=(P_{C_1},P_{D_1},\{(P_{C_1},D_1),(D_1,P_{D_1})\})$

由算法2得到各变体片段的控制流矩阵后，将其作为算法3的输入。首先，执行算法3的第1行，创建两个为空集的$\boldsymbol{ACFM}_i$和$\boldsymbol{MCFM}$；其次，执行第3—17行，以此得到流程变体S_1对齐的控制流矩阵，并在表9中将其列出。

表9 流程变体S_1中对齐的控制流矩阵

Table 9 Aligned control flow matrices in process variant S_1

	P_{ST}	P_A	P_B	…	P_{D_1}
P_{ST}		p_1			
P_A			p_2		
P_B					
⋮					
P_{D_1}					

表9中的元素只有原属于变体片段VF_1时才不为空集，其余部分都为空集。限于篇幅原因，其他变体片段对齐后的控制流矩阵就不在此一一列举。

将每个控制流矩阵一一对齐后，对齐后的控制流矩阵就包含了3个流程变体中的所有节点，利用对齐的控制流矩阵，就可以间接地将流程变体进行合并。

算法3中第18—20行是将对齐的控制流矩阵中的节点和控制流路径映射到合并的控制流矩阵。限于篇幅原因，

表 10 只列出合并的控制流矩阵的部分矩阵。

表 10 合并的控制流矩阵

Table 10 Consolidated control flow matrices

⋮⋯	P_E	P_F	P_G	P_H	P_I	P_J	P_N	⋯
P_E			p_8					
P_F				p_{10}				
P_G				p_{11}				
P_H					p_{12}			
P_I						p_{13}		
P_J							⋮	
P_N						⋯	⋱	
⋮								

合并后的控制流矩阵仍需要满足定理 1 和定理 2 的性质。通过检测表 10 中的控制流矩阵发现，矩阵中的某些行或列不能满足定理 1 或定理 2。

若源节点为一个库所，为了实现行属性，直接在源节点上添加一个可配置的发散语义注释；若源节点为一个变迁，则在其后续节点上对其添加一个可配置的发散语义注释。同样，为了实现列属性，直接在源节点上添加一个可配置的聚合语义注释；若源节点为一个变迁，则在其前置节点上对其添加一个可配置的聚合语义注释。那么表 10 中的第 1 行可以表示为 $p_1=(P_{ST},P_A,\{(P_{ST},P_{ST}^{div}),(P_{ST}^{div},A),(A,P_A)\})$，$p_{24}=(P_{ST},P_T,\{(P_{ST},P_{ST}^{div}),(P_{ST}^{div},T),(T,P_T)\})$，余下的控制流路径进行相同操作。

表 11 合并模型实际的控制流矩阵

Table 11 Consolidated control flow matrices

⋮⋯	P_E	P_F	P_G	P_H	P_I	P_J	P_K	P_L	P_M	P_N	⋯
P_E			p_8								
P_F				p_{10}							
P_G				p_{11}							
P_H					p_{12}						
P_I						p_{13}					
P_J							p_{39}	p_{40}			
P_K									p_{41}		
P_L									p_{42}		
P_M											
P_N											⋯
⋮									⋮		⋱

表 12 合并模型中的不期望行为

Table 12 Unexcepted behavior in merged models

迹 δ
$\langle TBCDFHIJKM\rangle,\langle TBCEGHIJKM\rangle,\langle TBCDFHIJLM\rangle,$ $\langle TBCEGHIJLM\rangle,\langle ANUVXYZA_1IB_1JLM\rangle$
$\langle TBCDFHC_1D_1DFHIJKM\rangle,\langle TBCEGHC_1D_1EGHIJKM\rangle$
$\langle TBCDFHC_1D_1DFHIJLM\rangle,\langle TBCEGHC_1D_1EGHIJLM\rangle,$ $\langle TNOQRSCDFHIJLM\rangle,\langle TNOPCEGHIJLM\rangle$
$\langle TNOPCDFHIJKM\rangle,\langle TNOPCEGHIJKM\rangle,\langle TNOPCDFHIJLM\rangle$ $\langle TNOQRSCDFHIJKM\rangle,\langle TNOQRSCEGHIJKM\rangle,$
$\langle TNOPCEGHC_1D_1EGHIJKM\rangle,\langle ANUVXYZA_1IB_1JKM\rangle$
$\langle TNOQRSCEGHIJLM\rangle,\langle TNOPCDFHC_1D_1DFHIJKM\rangle$
$\langle TNOPCDFHC_1D_1DFHIJLM\rangle,\langle TNOPCEGHC_1D_1EGHIJLM\rangle$
$\langle TNOQRSCDFHC_1D_1DFHIJKM\rangle,\langle ANW\rangle$
$\langle TNOQRSCEGHC_1D_1EGHIJKM\rangle$
$\langle TNOQRSCDFHC_1D_1DFHIJLM\rangle,$ $\langle TNOQRSCEGHC_1D_1EGHIJLM\rangle$
$\langle ANUVXYZA_1C_1D_1A_1IB_1JKM\rangle,\langle ANUVXYZA_1C_1D_1A_1IB_1JLM\rangle$

执行算法 3 中的第 21 行，将合并的控制流矩阵转变为由 Petri 表示的网系统，如图 5 所示。图 5 所示合并模型的实际控制流矩阵的部分矩阵如表 11 所列。表 10、表 11 所示控制流矩阵的区别之处在于，表 10 是由 3 个流程变体的控制流矩阵经过合并得到的，而表 11 是图 5 中的合并模型通过算法 2 得到的。同时，图 5 中合并模型的迹如表 2 和表 12 所列。

流程变体合并后，必须确保合并模型中的行为保持不变。换言之，合并模型中的行为是在原始的流程变体已经完成的情况下才能进行的。但是合并后的流程模型会产生一些不期望的行为。

执行算法 4 中的第 1—5 行，将表 10 和表 11 的控制流矩阵中的 p_i 数值化为 1，其余的数值化为 0。限于篇幅，数值化后的矩阵将不再展示。之后执行算法 4 的第 6—9 行，将数值化的控制流矩阵进行矩阵减运算，发现相减后的矩阵中数值化的控制流路径存在不为 0 的情况，说明合并模型中产生了不期望行为。

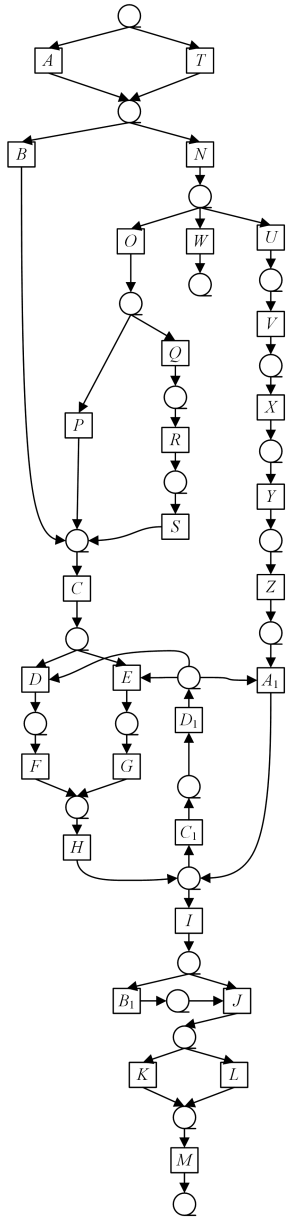


图 5 流程变体合并图

Fig. 5 Process variants merge diagram

执行算法 4 的剩余步骤从而检测出合并模型中的不期望

行为。寻找由合并模型产生的控制流矩阵中$|CF^{-1}[\{a\}]|\geqslant 2$的节点，得到以下节点：$B,C,H,D,E,J,A_1,I,M,N,C_1$。执行算法4从而手动寻找到合并模型中的不期望行为。合并模型中的不期望行为具体如表12所示。

为了避免合并模型中产生不期望的行为，需要对其中的某些活动进行行为约束，记为BC_a。

不难看出，活动J,M,I,H,C_1的发生不受到其他活动的限制。同理，活动N也不受活动A与T的限制，即活动N不管是活动A发生还是活动T发生，都会发生。因此，不用对上述5个活动进行行为约束。而活动B,C,D,E的发生依赖于活动A的发生，活动A_1的发生依赖于活动T的发生。除了$|CF^{-1}[\{a\}]|\geqslant 2$的活动需要添加行为约束外，合并模型中有些活动也需要进行行为约束，否则极易产生不期望行为，如活动O的发生依赖于活动A与N，活动W的发生依赖于活动T与N，活动B_1则直接依赖于活动T的发生，活动P与Q则依赖于活动A,N,O的发生。因此，上述需要进行约束的活动也可以记为：$BC_{B,C,D,E}=\langle A\rangle$，$BC_{A_1}=\langle T\rangle$，$BC_O=\langle A,N\rangle$，$BC_W=\langle T,N\rangle$，$BC_{P,Q}=\langle A,N,O\rangle$。只有当上述活动的行为约束满足条件时，才有发生的可能。

结束语 本文通过变体片段的控制流矩阵来帮助流程设计者设计新的业务流程。本文提出的合并方法是基于控制流矩阵，通过分组公平将流程变体分割为一个个变体片段，每一个变体片段都用一个控制流路径矩阵来表示。经过矩阵对齐后，一次性将变体片段合并转变为一个完整的流程模型，同时本文还提供了一组行为约束，以保持原流程模型中的行为，避免在合并模型中产生不期望行为。但是，这种不期望行为需通过一种手动的方法来检测，耗时且易出错。在未来的工作中，希望能提出自动检测不期望行为的方法，并对其进行行为约束。

参 考 文 献

[1] UBA R, DUMAS M, GARCIABANUELOS L, et al. Clone detection in Repositories of Business Process Models[C]// International Conference on Business Process Management. Berlin: Springer, 2011: 248-264.

[2] EKANAYAKE C C, DUMAS M, GARCIABANUELOS L, et al. Approximate clone detection in repositories of Business Process Models[C]// International Conference on Business Process Management. Berlin: Springer, 2012: 302-318.

[3] SARNO R, GINARDI H, PAMUNGKAS E W, et al. Clustering of ERP business process fragments[C]// International Conference on Computer Control Informatics and Its Applications, Jakarta, Indonesia, 2013: 319-324.

[4] HUANG Y, HE K, FENG Z, et al. Business Process Consolidation Based on E-RPSTs[C]// 2014 IEEE World Congress on Services. IEEE, 2014: 354-361.

[5] ASSY N, DONGEN B F V, AALST W M P V D. Discovering Hierarchical Consolidated Models from Process Families[C]// International Conference on Advanced Information Systems Engineering. Springer, Cham, 2017: 314-329.

[6] KUNCHALA J, YU J, YONGCHAREON S, et al. Towards merging collaborating processes for artifact lifecycle synthesis [C]// Australasian Computer Science Week Multiconference. ACM, 2017: 1-8.

[7] DISSAUX P, HALL B. Merging and Processing Heterogeneous Models[C]// Proceeding of the 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016). Toulouse, France, 2016.

[8] HACHEMI A, AHMED NACER M. Reusing Process Patterns in Software Process Models Modification[J]. Journal of Software: Evolution and Process, 2018, 30(8): e1938.

[9] BEUTEL M C, BOROZANOV V, GOKAY S, et al. Semi-automated Business Process Model Matching and Merging Considering Advanced Modeling Constraints[C]// 19th International Conference on Enterprise Information Systems. Porto, 2017: 324-331.

[10] WU Z H. Petri Net Introduction[M]. Beijing: Mechanical Industry Press, 2006.

[11] HAN J H, FANG H, LIU X P. Overview: Fairness in Petri Net, Analysis and Application[J]. Journal of System Simulation, 2012, 24(3): 521-535.

[12] ZHOU Y, LE X B, KUANG Y C. Petri net union decomposition technology and its application[J]. Computer Engineering & Science, 2013, 35(4): 125-129.

[13] XIA C L. Structural Analysis and Applications of Synthesis of Petri Nets Shared T-type Subnet[J]. Computer Science, 2007 (3): 240-245.

[14] DU Y Y, LI X Z. Analysis on liveness and fairness of shared path composition nets[J]. Journal of Chinese Computer Systems, 2000(9): 997-1000.

[15] JIA G Y, WU Z H, ZHANG G S. Properties of Union Operation for Petri Nets[J]. Journal of Shandong University of Science and Technology (Natural Science), 2004(1): 47-50.

[16] XUE Y, LI C J. Composition Operations for Fuzzy Petri Nets [J]. Journal of Hangzhou Dianzi University (Natural Sciences), 2012, 32(3): 79-82.

[17] STAINES A S. Matrix Representations for Ordinary Restricted Place Transition Nets[J]. WSEAS Transactions on Computers Archive, 2017(16): 23-29.

[18] STAINES A S, BARDIS N. Ordinary Petri Net Matrices[C]// Itm Web of Conferences. EDP Sciences, 2019, 24: 02007.

[19] WU Y F, TAN W A. Method for Calculating the Distance Between Process Models Based on the Correlation Matrix of Petri Net[J]. Computer & Digital Engineering, 2018, 46(3): 429-436.

[20] KUNCHALA J, YU J, YONGCHAREON S, et al. An approach to merge collaborating processes of an inter-organizational business process for artifact lifecycle synthesis[J]. Computing, 2020, 102(4): 951-976.

[21] ASSY N, CHAN N N, GAALOUL W. An Automated Approach for Assisting the Design of Configurable Process Models[J]. IEEE Transactions on Services Computing, 2015, 8(6): 874-888.

WANG Wu-song, born in 1995, postgraduate. His main research interests include process mining and analysis and management of process variants.

FANG Huan, born in 1982, Ph.D, professor, is a member of China Computer Federation. Her main research interests include petri nets theory and applications, behavior profile, change mining and process mining.