arXiv:2303.10039v1 [cs.FL] 17 Mar 2023

关于类型化 Jackson 网的可重构性与可再发现性（扩展版）

DaniëlBarenholz[1],Marco Montali[2], Artem Polyvyanyy[3], Hajo A. Reijers[1], Andrey Rivkin[2,4],和 Jan Martijn E. M. van der Werf[1]

[1] 信息与计算科学系，乌得勒支大学，Princetonplein 5, 3584 CC 乌得勒支，荷兰{d.barenholz,h.a.reijers,j.m.e.m.vanderwerf}@uu.nl[2] 计算机科学学院，博尔扎诺自由大学，piazza Domenicani 3, 39100，博尔扎诺，意大利montali@inf.unibz.it[3] 墨尔本大学，维多利亚州 3010，澳大利亚 artem.polyvyanyy@unimelb.edu.au[4] 应用数学与计算机科学系，丹麦技术大学，Richard Petersens Plads 321, 2800 Kgs. Lyngby，丹麦 ariv@dtu.dk

摘要。 流程发现算法旨在根据历史系统执行生成的数据构建模型，使模型能够良好地描述系统。因此，流程发现算法的一项理想属性是可再发现性，这确保算法能够构建出与原始系统行为等价的模型。系统通常会同时执行多个流程，并通过对象操作进行交互。本文提出了一个框架，用于开发流程发现算法，以基于使用标识符引用所操作对象的类型化 Jackson 网，构建描述交互流程的模型。类型化 Jackson 网具有可重构性 这一属性，即分解后的类型化杰克逊网的流程组合及其交互所生成的模型，与原始系统双模拟等价。我们利用这一属性来证明，如果流程发现算法能够确保可再发现性，那么交互进程系统也具有可再发现性。

1 引言

业务流程是各种系统的基础。业务流程由一组活动构成，执行这些活动旨在实现组织的业务目标。业务流程的示例包括零售商的订单到收款流程、医院的医疗评估流程，或银行的信用审查流程。业务流程使用流程建模语言（如 Petri 网）进行建模，并用于交流和分析目的 [1]。

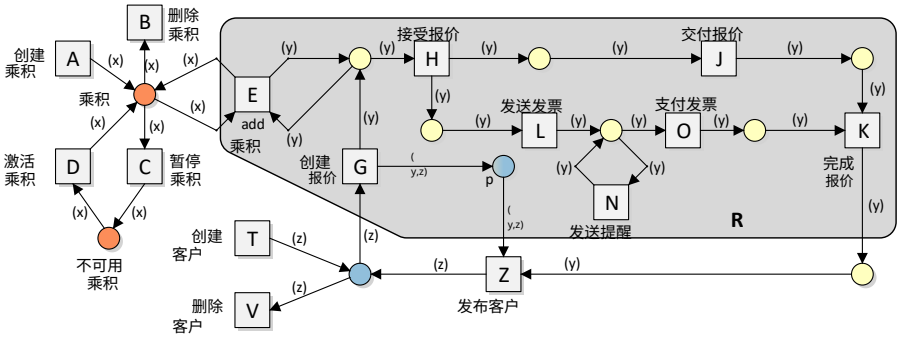


图1：由三个交互进程构成的零售商系统。

Petri 网以图形方式表示流程中活动的流转，并可用于对各种并发行为和顺序行为进行建模 [18]。

流程发现算法旨在根据系统历史流程执行所产生、并记录在事件日志中的数据，自动构建模型，使模型能够良好地描述该系统。发现算法所期望具备的一个性质是可再发现性。该性质指出，如果系统 S（以模型 M 表示）生成了事件日志 L，那么具备可再发现性这一性质的发现算法应当能够根据 L 构建出 M。换言之，该算法能够根据模型生成的数据对系统模型进行逆向工程。现有算法中，只有少数能够保证这一性质。例如，如果模型是一个块结构工作流网，且事件日志是直接后继完备的，那么 α-Miner 算法 [22] 能够再发现生成该事件日志的网。同样，在事件日志直接后继完备这一假设下，Inductive Miner [16] 能够再发现不含重复变迁、自循环或静默变迁的过程树。

大多数现有流程发现算法都假设系统只执行一个流程 [4]。因此，事件日志被定义为一组序列，其中每个序列描述一个流程实例的执行过程。然而，许多信息系统（例如企业资源规划系统）并不满足这一假设。系统通常会执行多个相互交互的流程 [10,23]。例如，考虑图1所示的零售商系统，它执行三个流程：订单流程、产品流程和客户管理流程。这些流程彼此交织。具体而言，只有可用产品才能被订购，而且客户一次只能有一个订单。因此，事件并不属于单一流程，而是与多个流程相关。例如，考虑某个事件日志中的事件 e：该事件发生时，系统为某位客户 c 执行了变迁 G，并创建了一个新订单 o。事件e 同时关联客户流程实例 c 和订单流程实例 o。传统流程发现技术要求将事件 e 存储在多个事件日志中，并生成多个模型，每个流程 [7]对应一个模型。

工件或以对象为中心的流程发现 [5,17]以及智能体系统发现 [20,21]采用了不同的方法。在以对象为中心的流程发现中，

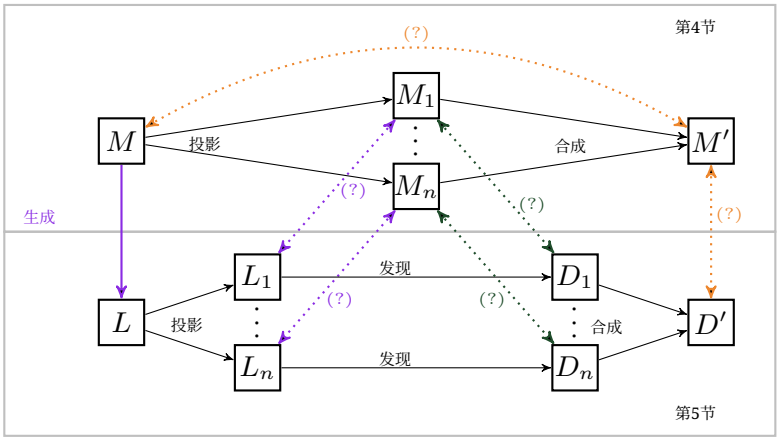


图2：交互进程系统的可再发现性框架。

事件并非只关联到单个对象，而是可以关联到存储在以对象为中心的事件日志 [8] 中的多个对象。现有的以对象为中心的发现算法会将输入事件日志投影到每种对象类型上，从而生成一组“扁平化”的事件日志。对于每个事件日志，都会发现一个模型，之后再将这些模型合并为一个模型 [5]。一般来说，扁平化会造成信息损失 [7], ，因为在这一步中，事件可能消失 [5], 、重复（收敛） [3], ，或导致错误的事件顺序（发散） [3]。在智能体系统发现中，系统不再被视为由相互交互的对象构成，而是被视为由多个自主智能体组成；每个智能体都驱动其自身的流程，这些流程彼此交互，以实现系统的总体目标 [20]。智能体系统发现算法会将输入事件日志分解为多个事件日志，其中每个日志都由一个智能体（类型）执行的事件以及一个交互事件日志组成；随后发现智能体模型和交互模型，并将它们组合成最终系统 [21]。

本文研究在何种条件下，事件日志中的投影能够保证交互进程的可再发现性，这些交互进程以类型化 Jackson 网表示。类型化 Jackson 网是带标识符的带类型 Petri 网的一个子类 [19,23]。类型化 Jackson 网这一类受到 Box Algebra [9] 和 Jackson 网 [14],的启发；它们是块结构工作流网的（表示形式），并且凭构造即可健全 [2] [16]。正如我们所展示的，类型化 Jackson 网具有一种特殊性质：它们是可重构的。仅组合每种类型的投影，不足以重构类型化 Jackson 网。相反，若考虑所有类型组合构成的子集闭集，组合便会返回系统的原始模型。我们将展示如何利用可重构性来构建一个框架，借助传统流程发现算法实现类型化 Jackson 网的可再发现性。该框架采用分而治之策略，如图 2所示。这一策略的核心思想是，将系统某个模型 M 生成的事件日志 L 投影为日志 L_1，…， L_n。然后，如果这些投影后的事件日志满足某种流程发现算法的条件，那么将所得模型 D_1，…， D_n 组合成模型 D'，就应当

能够重新发现系统的原始模型。在此框架中，我们证明，每个投影后的事件日志也是相应投影模型的事件日志。因此，如果某种流程发现算法能够保证投影模型的可再发现性，那么就可以使用类型化 Jackson 网的组合算子来确保原始系统的可再发现性。

下一节将介绍基本概念。在第3节中，我们将介绍类型化 Jackson 网；如第4节所示，这类网是可重构的。我们将在第5节中定义一个用于开发发现算法的框架，以保证可再发现性。最后，我们将在第6节结束本文。

2 预备知识

设 S 和 T 为两个可能是无限集的集合。 S 的幂集记为 $\mathrm{P}(S) = \{S' \mid S' \subseteq S\}$， $|S|$ 表示 S的基数。若两个集合 S 和 T 不相交，则称它们互不相交，其中 $S \cap T = \emptyset$， $\emptyset$ 表示空集。两个集合 S和 T的笛卡尔积定义为 $S \times T = \{(a, b) \mid a \in S, b \in T\}$。对于某个集合 S 以及集合 T_s （ $s \in S$），其广义笛卡尔积定义为 $\Pi_{s \in S} T_s = \{f : S \to \bigcup_{s \in S} T_s \mid \forall s \in S : f(s) \in T_s\}$。给定一个关系 $R \subseteq S \times T$，其值域定义为 $\mathrm{rng}(R) = \{y \in T \mid \exists x \in S : (x, y) \in R\}$。类似地，$R$ 的定义域定义为 $\mathrm{dom}(R) = \{x \in S \mid \exists y \in T : (x, y) \in R\}$。将关系的定义域限制到集合 U上，定义为 $R_{|U} = \{(a, b) \in R \mid a \in U\}$。

多重集 m over S 是如下形式的映射 $m : S \to \mathbb{N}$，其中 $\mathbb{N} = \{0, 1, 2, \ldots\}$ 表示自然数集合。对于 $s \in S$， $m(s) \in \mathbb{N}$表示 s 在多重集 m中出现的次数。若 $m(s) = n$，则记作 s^n 。对于$x \notin S$， $m(x) = 0$。我们用 $S^{\oplus}$ 表示 S 上的所有有限多重集构成的集合，并且也用 $\emptyset$ 表示空多重集。多重集的大小定义为$|m| = \sum_{s \in S} m(s)$。 $m \in S^{\oplus}$ 的支撑集是其中至少出现一次的元素组成的集合：$\mathrm{supp}(m) = \{s \in S \mid m(s) > 0\}$。给定 S上的两个多重集 m_1 和 m_2：(i) $m_1 \subseteq m_2$ （分别为 $m_1 \subset m_2$）当且仅当对每个 $s \in S$， $m_1(s) \leq m_2(s)$（分别为 $m_1(s) < m_2(s)$）；(ii) 对每个 $s \in S$，$(m_1 + m_2)(s) = m_1(s) + m_2(s)$；(iii) 若$m_1 \subseteq m_2$，则对每个 $s \in S$，$(m_2 - m_1)(s) = m_2(s) - m_1(s)$。

在 S 上的长度为 $n \in \mathbb{N}$ 的序列是一个函数 $\sigma : \{1, \ldots, n\} \to S$。若 $n > 0$ 且 $\sigma(i) = a_i$，对于 $1 \leq i \leq n$，我们记作 $\sigma = \mathbf{h}a_1, \ldots, a_n\mathbf{i}$。序列$\sigma$ 的长度记作 $|\sigma|$。长度为 0 的序列称为空序列，记作 ϵ。在 S 上的所有有限序列的集合记作 S^*。若存在 $1 \leq i \leq |\sigma|$，使得 $\sigma(i) = a$ 且 $\mathrm{supp}(\sigma) = \{a \in S \mid \exists 1 \leq i \leq |\sigma| : \sigma(i) = a\}$，则记作$a \in \sigma$。两个序列 ν、$\gamma \in S^*$ 的连接记作 $\sigma = \nu \cdot \gamma$，它是一个序列，定义为 $\sigma : \{1, \ldots, |\nu| + |\gamma|\} \to S$，满足对于$1 \leq i \leq |\nu|$， $\sigma(i) = \nu(i)$；对于 $|\nu| + 1 \leq i \leq |\nu| + |\gamma|$， $\sigma(i) = \gamma(i - |\nu|)$。序列在集合 T 上的投影通过归纳定义为 $\epsilon_{|T} = \epsilon$，$(\mathbf{h}a\mathbf{i} \cdot \sigma)_{|T} = \mathbf{h}a\mathbf{i} \cdot \sigma_{|T}$ （若$a \in T$），否则为 $(\mathbf{h}a\mathbf{i} \cdot \sigma)_{|T} = \sigma_{|T}$。用单射 $r : S \to T$ 对序列进行重命名，通过归纳定义为 $\rho_r(\epsilon) = \epsilon$，以及 $\rho_r(\mathbf{h}a\mathbf{i} \cdot \sigma) = \mathbf{h}r(a)\mathbf{i} \cdot \rho_r(\sigma)$。重命名可扩展到序列多重集，如下所示：给定一个多重集 $m \in (S^*)^{\oplus}$，我们定义 $\rho_r(m) = \sum_{\sigma \in \mathrm{supp}(m)} \sigma(m) \cdot \rho_r(\sigma)$。例如，$\rho_{\{x \mapsto a, y \mapsto b\}}(\mathbf{h}x, y\mathbf{i}^3) = \mathbf{h}a, b\mathbf{i}^3$。

有向图是一个二元组 (V, A)，其中 V 是顶点集，$A \subseteq V \times V$ 是弧集。若存在双射 $b: V_1 \to V_2$，使得 $(v_1, v_2) \in A_1$ 当且仅当 $(b(v_1)\ b(v_2)) \in A_2$，则图 $G_1 = (V_1, A_1)$ 与 $G_2 = (V_2, A_2)$ 同构，记作 $G_1 \leftrightsquigarrow G_2$。

给定一个由 A 个（动作）标签构成的有限集，在 A 上的（标记）迁移系统（LTS）是一个元组 $\Gamma_A = (S, A, s_0, \to)$，其中 S 是（可能为无限的）状态集，s_0 是初始状态，而 $\to \subset (S \times (A \cup \{\tau\}) \times S)$ 是迁移关系，其中 $\tau \notin A$ 表示静默动作 [12]。下文中，我们用 $s \xrightarrow{a} s'$ 表示 $(s, a, s') \in \to$。令 $r: A \to (A' \cup \{\tau\})$ 为一个单射全函数。用 r 对 Γ 进行重命名，定义为 $\rho_r(\Gamma) = (S, A \backslash A', s_0, \to')$，其中 $(s, r(a)\ s') \in \to'$ 当且仅当 $(s, a, s') \in \to$。给定集合 T，隐藏定义为 $\hat{}\ \mathtt{H}_T(\Gamma) = \rho_h(\Gamma)$，其中 $h: A \to A \cup \{\tau\}$，满足：若 $t \in T$，则 $h(t) = \tau$；否则 $h(t) = t$。给定 $a \in A$，$p \overset{a}{\Rightarrow} q$ 表示如下定义的弱迁移关系：(i) $p \overset{a}{\Rightarrow} q$ 当且仅当 $p(\xrightarrow{\tau})^* q_1 \xrightarrow{a} q_2 (\xrightarrow{\tau})^* q$；(ii) $p \overset{\tau}{\Rightarrow} q$ 当且仅当 $p(\xrightarrow{\tau})^* q$。这里，$(\xrightarrow{\tau})^*$ 表示 $\xrightarrow{\tau}$ 的自反传递闭包。

设 $\Gamma_1 = (S_1, A, s_{01}, \to_1)$ 和 $\Gamma_2 = (S_2, A, s_{02}, \to_2)$ 为两个标记迁移系统。若对每一对 $(p, q) \in R$ 和 $a \in A \cup \{\tau\}$，只要 $p \xrightarrow{a}_1 p'$，就存在$q' \in S_2$ 使得 $q \xrightarrow{a}_2 q'$ 且 $(p', q') \in R$，则关系$R \subseteq (S_1 \times S_2)$ 称为强模拟，记作 $\Gamma_1 \prec_R \Gamma_2$。关系 R 称为弱模拟，记作 $\Gamma_1 \preccurlyeq_R \Gamma_2$，当且仅当对每一对 $(p, q) \in R$ 和 $a \in A \cup \{\tau\}$，只要 $p \xrightarrow{a}_1 p'$，就有 $a = \tau$ 且 $(p', q) \in R$，或者存在 $q' \in S_2$ 使得 $q \overset{a}{\Rightarrow}_2 q'$且 $(p', q') \in R$。若 $\Gamma_1 \prec \Gamma_2$ $(\Gamma_1 \preccurlyeq_R \Gamma_2)$ 且 $\Gamma_2 \prec_{R^{-1}} \Gamma_1$ $(\Gamma_2 \preccurlyeq_{R^{-1}} \Gamma_1)$，则关系 R 称为强（弱）双模拟，记作$\Gamma_1 \sim_R \Gamma_2$ $(\Gamma_1 \approx_R \Gamma_2)$。给定强（弱）（双）模拟 R，若 $(p, q) \in R$，则称状态 $p \in S_1$ 与 $q \in S_2$ 强（弱）根式（双）相似，记作 $p \sim_R^r q$（相应地，$p \approx_R^r q$）。当且仅当 $(s_{01}, s_{02}) \in R$ 时，该关系称为根式关系。根式关系用上标 ^{r} 表示。

加权 Petri 网是一个四元组 (P, T, F, W)，其中 P 和 T 分别是两个不相交的库所集合和变迁集合，$F \subseteq ((P \times T) \cup (T \times P))$ 是流关系，而 $W: F \to \mathbb{N}^+$ 是一个权函数。对于 $x \in P \cup T$，我们记作 ${}^\bullet x = \{y \mid (y, x) \in F\}$ 表示 x的前置集，$x^\bullet = \{y \mid (x, y) \in F\}$ 表示 x的后置集。我们将前置集和后置集的记法逐元素扩展到集合。若 Petri 网未定义权函数，则假设对所有 $f \in F$，$W(f) = 1$。N的标识是一个多重集 $m \in P^\oplus$，其中 $m(p)$ 表示库所 p $\in P$中的令牌数。若 m(p) > 0，则库所 p 在标识 m中称为有标记的。有标记的 Petri 网是一个元组 (N, m)，其中 N 是一个标识为 m的加权 Petri 网。变迁 t $\in T$ 在 (N, m) 中使能，记作 (N, m) [ti，当且仅当对所有 p $\in {}^\bullet$t，W((p, t)) $\leq m$(p)。使能的变迁可以发射，产生标识 m'，当且仅当对所有 p $\in P$，m'(p) $+ W$((p, t)) $= m$(p) $+ W$((t, p))；记作 (N, m) [ti(N, m')。我们将发射的记法扩展到序列。序列 $\sigma \in T^*$ 是发射序列，当且仅当 $\sigma = \epsilon$，或者存在标识 m_0、.....、m_n，使得对 $1 \leq i \leq |\sigma| = n$，$(N, m_{i-1})[\sigma(i)\mathrm{i}(N, m_i)$，记作 $(N, m_0)[\sigma\mathrm{i}(N, m_n)$。若上下文明确，则省略加权 Petri 网 N。(N, m)的可达标识集合定义为 R(N, m) $= \{m' \mid \exists \sigma \in T^*: m[\sigma\mathrm{i}m'\}$。$(N, m)$的所有可能有限发射序列集合为

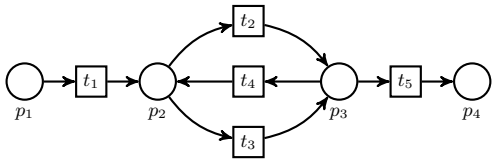


图 3：一个块结构的工作流网示例。每个块都对应于 Jackson 类型中的一个节点（p_1; (t_1; ((((p_2; (($t_2 + t_3$) ; p_3)) #t_4) ;(t_5; p_4)))). 例如，变迁 t_2 和 t_3 之间的选择对应于节点 (p_2; (($t_2 + t_3$) ; p_3))。

记作 L(N, m_0) ={$\sigma \in T^* \mid m[\sigma$i$m'$}。有标记的 Petri 网 ($N, m$) 的语义，其中 N =(P, T, F, W)，定义为标记迁移系统 $\Gamma_{N,m}$ =($P^{\oplus}, T, m_0, \rightarrow$)，其中 ($m, t, m'$) $\in \rightarrow$ 当且仅当 $m[t$im'。Petri 网 N =(P, T, F, W) 的底层图为 ($P \cup T, F$)。两个 Petri 网 N和 N'同构，记作 $N \leftrightsquigarrow N'$，当且仅当它们的底层图同构。

工作流网（简称 WF-net）是一个元组 N =(P, T, F, W, in, out)，满足：（i）(P, T, F, W) 是一个加权 Petri 网；（ii）in、out $\in P$分别是源库所和汇库所，其中 $^\bullet$属于 = out$^\bullet$ = $\emptyset$；（iii） $P \cup T$中的每个节点都位于一条从 in 到 out 的有向路径上。若对于某个 $k \in \mathbb{N}$， N 满足以下条件，则称其为 k-健全：（i）它能正确完成，即对于所有可达标识 $m \in$ R(N, [in^k]，如果[out^k] $\subseteq m$，那么 m = [out^k]；（ii）它是弱终止的，即对于任意可达标识 $m \in$R(N, [in^k])，终止标识是可达的，即 [out^k] $\in$R(N, m)；（iii）它是准活的，即对于所有变迁 $t \in T$，都存在一个标识 $m \in$R(N, [in])，使得 $m[t$i。若该网是 1-健全的，则称其为健全网。若它对所有 $k \in \mathbb{N}$ 都是 k-健全的，则称其为广义健全网 [13]。

3 类型化 Jackson 网：建模交互流程

本节将类型化 Jackson 网介绍为带标识符的带类型 Petri 网的子类。我们表明，这一类是 Jackson 网的自然扩展，而 Jackson 网是块结构工作流网的表示。类型化 Jackson 网在构造上具有标识符健全性和活性。

3.1 Jackson 网

WF-nets 不对活动的控制流施加任何限制，而块结构 WF-nets 则将控制流划分为逻辑块 [15]。每个“块”代表一个可执行的单一工作单元，该工作单元可以是原子的（单个变迁），也可以包含多个步骤（多个变迁）。图 3 展示了一个块结构 WF-net。块结构 WF-nets 的主要优势在于，按照定义，块结构可确保 WF-net 是健全的 [15,16,14]。本文中，我们考虑 Jackson

类型和 Jackson 网 [14]。AJacksonType 是一种数据结构，用于捕获 WF-net 单次执行所涉及的所有信息。

定义 1（Jackson 类型 [14]）。*Jackson* 类型的集合 J 由以下文法递归定义：

$$\begin{aligned}\mathcal{J} &::= \mathscr{A}^p \mid \left(\mathscr{A}^p; \left(\mathcal{J}^t; \mathscr{A}^p\right)\right)\\ \mathcal{J}^t &::= \mathscr{A}^t \mid \left(\mathcal{J}^t; \left(\mathcal{J}^p; \mathcal{J}^t\right)\right) \mid \left(\mathcal{J}^t + \mathcal{J}^t\right)\\ \mathcal{J}^p &::= \mathscr{A}^p \mid \left(\mathcal{J}^p; \left(\mathcal{J}^t; \mathcal{J}^p\right)\right) \mid \left(\mathcal{J}^p \parallel \mathcal{J}^p\right) \mid \left(\mathcal{J}^p \# \mathcal{J}^t\right)\end{aligned}$$

其中，$\mathscr{A} = \mathscr{A}^p \cup \mathscr{A}^t = \{a, b, c, \ldots\}$ 表示分别用于库所和变迁的两个不相交的原子类型集合，符号 ;、k +、$\#$ 分别表示序列、并行、选择和循环。◁

同一个工作流网可能对应多个 Jackson 类型。例如，Jackson 类型 $((p_1;\ t_1)$;(((p2; ((t2 + t3) ; p3)) #t4) ; (t5; p4))) 也描述了图 3 中的工作流网。每个网都有唯一的表示形式 [14], ，称为其范式。我们定义类型之间的代数等价，以便将其重写为范式。

定义 2（代数等价、范式 [14]）。代数等价 $\equiv_{alg}$ 是 *Jackson* 类型集合上的最小等价关系，满足以下六条规则：

$$\begin{array}{cc}((J_0; J_1); J_2) \equiv_{alg} (J_0; (J_1; J_2)) & ((J_0 + J_1) + J_2) \equiv_{alg} (J_0 + (J_1 + J_2))\\ ((J_0 \parallel J_1) \parallel J_2) \equiv_{alg} (J_0 \parallel (J_1 \parallel J_2)) & (J_0 + J_1) \equiv_{alg} (J_1 + J_0)\\ (J_0 \parallel J_1) \equiv_{alg} (J_1 \parallel J_0) & ((J_0 \# J_1) \# J_2) \equiv_{alg} (J_0 \# (J_1 \# J_2))\end{array}$$

具有 $J_0, J_1, J_2 \in J$ 三种 *Jackson* 类型。

当按照上述规则将所有括号移至右侧时，*Jackson*类型便处于标准形式 。◁

Jackson 网这一类网可通过从仅含一个库所的单元素网开始，递归应用生成规则得到。这些生成规则与 Murata [18] 定义的规则类似，且能保持可靠性 [14]。因此，任何 Jackson 网在构造上都是健全的。

定义 3（Jackson 网 [14]）。若一个 *WF-net* $N = (P, T, F, in, out)$ 可以从单个库所 p 出发，通过递归应用以下五条生成规则得到，则称其为Jackson 网：

$$\begin{array}{ll}J1{:}\ p \leftrightarrow (p_1; (t; p_2)) & J4{:}\ p \leftrightarrow (p_1 \parallel p_2)\\ J2{:}\ t \leftrightarrow (t_1; (p_1; t_2)) & J5{:}\ t \leftrightarrow (t_1 + t_2)\\ J3{:}\ p \leftrightarrow (p \# t) & \end{array}$$

我们称 N 由 p生成。◁

如 [14], 所示，Jackson 网完全由 Jackson 类型决定，反之亦然。

定理 1（Jackson 网与 Jackson 类型等价 [14]）。设N_1 和 N_2 是分别由 *Jackson* 类型 J_1 和J_2生成的两个 *Jackson* 网。当且仅当 $J_1 \equiv_{alg} J_2$时， N_1 和 N_2 同构。◁

3.2 带标识符的 Petri网

工作流网描述单个案例的所有可能执行过程，而系统通常由许多交互进程组成。后者可以用带标识符的带类型 Petri 网（简称 t-PNID）[23]建模。在这种形式体系中，每个对象都有类型和唯一标识符，以便对其进行引用。令牌携带标识符向量，用于关联对象。弧上的变量用于操纵这些标识符。

定义4（标识符、类型和变量）。令 I、Λ 和 V 分别表示标识符、类型标签和变量的可数无限集。我们定义：

– 域赋值函数 $I : \Lambda \to P(I)$，使得 $I(\lambda_1)$ 是无限集，并且 $I(\lambda_1) \cap I(\lambda_2) \neq \emptyset$ 蕴含 $\lambda_1 = \lambda_2$ 对所有 λ_1、$\lambda_2 \in \Lambda$ 均成立；– 标识符类型函数 $\mathtt{type}_{\mathcal{I}} : I \to \Lambda$，满足：若 $\mathtt{type}_{\mathcal{I}}(\mathtt{id}) = \lambda$，则 $\mathtt{id} \in I(\lambda)$；– 变量类型函数 $\mathtt{type}_{\mathcal{V}} : V \to \Lambda$，规定 $x \in V$ 中的变量只能替换为来自 $I(\mathtt{type}_{\mathcal{V}}(x))$ 的值。

根据上下文可以明确时，我们省略 $\mathtt{type}$ 的下标。我们将这些函数扩展到集合、向量和序列，方法是对其中的每个元素应用该函数。◁

在 t-PNID 中，每个库所都标注有一个标签，称为库所类型。库所类型是一个类型向量，表示该库所可以容纳的标识符令牌的类型。与 Jackson 类型类似，我们用 $[p, \lambda]$ 表示库所 p 的类型为$\alpha(p) = \lambda$。每条弧都标注有一个标识符向量多重集，使得每个变量的类型都与库所类型一致。如果标注为空或仅含一个元素，则省略括号。

定义5（带标识符的类型化 Petri 网）。一个带标识符的类型化 Petri 网（*t-PNID*）N 是一个元组（P，T，F，α，β），其中：

– (P, T, F) 是一个经典*Petri*网；– $\alpha : P \to \Lambda^*$ 是库所类型函数；– $\beta : F \to (V^*)^{\oplus}$ 为每条弧定义一个变量向量多重集，使得对于任意 $x \in supp\,(\beta((p, t)))$，$\alpha(p) = \mathtt{type}(x)$，且对于任意$y \in supp\,(\beta((t, p')))$，$\mathtt{type}(y) = \alpha(p')$，其中 $t \in T$, $p \in {}^\bullet t$, $p' \in t^\bullet$。◁

t-PNID 的标识是定义在库所集合上的令牌配置。库所中的每个令牌都应具有正确的类型，即令牌在库所中携带的标识符向量应与相应的库所类型匹配。集合 $\mathtt{C}(p)$ 定义了库所 p 可能携带的所有标识符向量。

定义 *6*（标识）。给定一个 *t-PNID* $N = (P, T, F, \alpha, \beta)$，以及库所 $p \in P$，其标识符集合为 $\mathtt{C}(p) = \prod_{1 \le i \le |\alpha(p)|} I(\alpha(p)(i))$。标识是一个函数 $m \in \mathbb{M}(N)$，其中 $\mathbb{M}(N) = P \to (\Lambda^*)^{\oplus}$，且对每个库所 $p \in P$，都有 $m(p) \in \mathtt{C}(p)^{\oplus}$。$m$ 中使用的标识符集合记作 $Id(m) = \bigcup_{p \in P} rng(supp\,(m(p)))$。有序对 (N, m) 称为标记 *t-PNID*。◁

要定义 t-PNID 的语义，需要为变量赋予标识符。

定义 7（变量集 [23]）。给定一个 *t-PNID* $N = (P, T, F, \alpha, \beta)$，$t \in T$和 $\lambda \in \Lambda$，我们定义以下变量集：

– 输入变量为 $\mathrm{In}(t) = \bigcup_{x \in \beta((p,t))} {}_{p \in {}^\bullet t} \mathrm{rng}(\mathrm{supp}(x))$; – 输出变量为 $\mathrm{Out}(t) = \bigcup_{x \in \beta((t,p))} {}_{p \in t^\bullet} \mathrm{rng}(\mathrm{supp}(x))$; – 变量为 $\mathrm{Var}(t) = \mathrm{In}(t) \cup \mathrm{Out}(t)$; – 发射变量为 $\mathrm{Emit}(t) = \mathrm{Out}(t) \setminus \mathrm{In}(t)$; – 收集变量为 $\mathrm{Collect}(t) = \mathrm{In}(t) \setminus \mathrm{Out}(t)$; – 发射变迁为 $E_N(\lambda) = \{t \mid \exists x \in \mathrm{Emit}(t) \wedge \mathtt{type}(x) = \lambda\}$; – 收集变迁为 $C_N(\lambda) = \{t \mid \exists x \in \mathrm{Collect}(t) \wedge \mathtt{type}(x) = \lambda\}$; – N中的类型为 $\mathtt{type}(N) = \{\vec{\lambda} \mid \exists p \in P\colon \vec{\lambda} \in \alpha(p)\}$。

◁

将变量赋值为标识符称为绑定。绑定用于通过发射标识符的变量（即仅出现在该变迁输出弧上的变量）向网中注入新的新鲜数据。请注意，在此定义中，标识符的新鲜性是相对于标识而言的，也就是说，已消失的标识符（那些通过收集变迁从网中完全移除的标识符）可以重复使用，因为这不会妨碍 t-PNID 的语义。

定义 *8*（*t-PNID* 的发射规则）。给定一个标记 *t-PNID* (N, m) *with* $N = (P, T, F, \alpha, \beta)$，变迁 $t \in T$的一个绑定是单射$\psi : V \to I$，满足 $\mathtt{type}(v) = \mathtt{type}(\psi(v))$，且 $\psi(v) \notin Id(m)$ 当且仅当 $v \in Emit(t)$。在绑定 ψ下，若变迁 t 在 (N, m) 中使能，记作 $(N, m)[t, \psi\rangle$，当且仅当对所有 $p \in {}^\bullet t$，$\rho_\psi(\beta(p, t)) \leq m(p)$。其发射产生标识 m'，记作 $(N, m)[t, \psi\rangle(N, m')$，使得 $m'(p) + \rho_\psi(\beta(p, t)) = m(p) + \rho_\psi(\beta(t, p))$。◁

发射规则以归纳方式扩展到序列。若存在 $\eta \in (T \times (V \to I))^*$ 使得 $(N, m)[\eta\rangle(N, m')$，则标识 m' 可从 m 到达。我们用 $R(N, m)$ 表示从 m 出发，对于 (N, m) 可达的所有标识的集合。我们用 $L(N, m)$ 表示 (N, m) 的所有可能发射序列，即 $L(N, m) = \{\eta \mid (N, m)[\eta\rangle\}$，且 $Id(\eta) = \bigcup_{(t,\psi) \in \eta} \mathrm{rng}(\psi)$，其中 $\mathrm{rng}(\psi)$ 是 η 中所用标识符的集合。t-PNID 的执行语义定义为一个标记迁移系统，它涵盖从给定初始标识出发的所有可能执行。若两个 t-PNID 所导出的迁移系统双模拟等价，则称这两个 t-PNID 双模拟等价。

定义 *9.* 给定一个带有 $N = (P, T, F, \alpha, \beta)$ 的标记 *t-PNID* (N, m_0)，其诱导的迁移系统为 $\Gamma_{N,m_0} = (\mathbb{M}(N), (T \times (V \to I)), m_0, \to)$，其中$m \xrightarrow{(t,\psi)} m'$ 当且仅当 $(N, m)\ [t, \psi\rangle(N, m')$。◁

工作流网的可靠性性质通常包括正确完成、弱终止性和准活性 [6]。将可靠性扩展到 t-PNID 后，便得到标识符健全性 [23]。在 t-PNID 中，给定类型的每个对象都会通过发射变迁“进入”系统，并与唯一标识符绑定。直观而言，标识符健全性表示对象应始终可以被移除（弱类型终止性）；并且，一旦某个对象的收集变迁发射，就不应再有令牌引用已移除的对象（正确类型完成）。

定义 10（标识符健全性 [23]）。设 (N, m_0) 为一个标记 *t-PNID*，且 $\lambda \in \Lambda$ 为某种类型。(N, m_0) 是 λ-健全的 当且仅当它

– 适当的 λ-完成性，即，对于所有 $t \in C_N(\lambda)$、绑定 $\psi : V \to I$ 以及标识 $m, m' \in R(N, m_0)$，若 $m[t, \psi\rangle m'$，则对于所有标识符 $\mathtt{id} \in rng(\psi|_{Collect(t)}) \cap Id(m)$ 和 $\mathtt{type}(\mathtt{id}) = \lambda$，均有 $\mathtt{id} \notin Id(m')$ [5]；– 弱 λ-终止性，即，对于每个 $m \in R(N, m_0)$ 和标识符 $\mathtt{id} \in I(\lambda)$，若满足 $\mathtt{id} \in Id(m)$，则存在一个标识 $m' \in R(N, m)$，使得 $\mathtt{id} \notin Id(m')$。

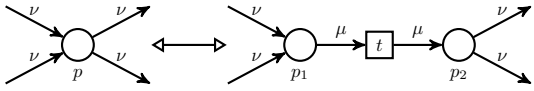
如果对于所有 $\lambda \in \mathtt{type}(N)$，它都是 λ-健全的，那么它就是标识符健全的。 ◁

3.3 类型化 Jackson 网

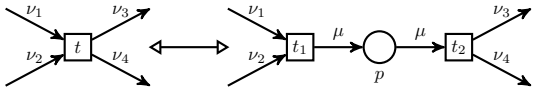
一般而言，对于 t-PNID，标识符可靠性是不可判定的 [23]。类似于 Jackson 网将工作流网限制为块，类型化 *Jackson* 网（t-JN）将 t-PNID 限制为块，同时保证标识符可靠性和活性。对于 t-JN，我们不允许弧和变量具有重数，即 $\beta(f)(v) \leq 1$ 对所有 $f \in F$ 和 $v \in \mathcal{V}$ 均成立，并且要求变量与标识符类型之间存在双射。这排除了诸如 $\lambda = \langle x, x\rangle$ 这样的库所类型。假设类型上存在一个类哥德尔数（参见 [14]），则库所类型和弧标注可以表示为集合。类似于 Jackson 类型用于描述 Jackson 网，我们采用基于 Jackson 类型的记法来表示类型化 Jackson 网。

定义 11（类型化 Jackson 网）。若 *t-PNID* N 可通过递归应用以下六条生成规则中的任意一条，由一组变迁 T' 生成，则称其为类型化 Jackson 网。如果 N 由单个变迁组成的集合生成（即 $|T'| = 1$），则称 N 为原子的。

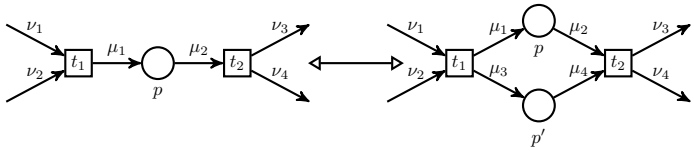
R1 库所扩展：$[p, \lambda] \leftrightarrow ([p_1, \lambda]\,;\,(t_1;\,[p_2, \lambda]))$



R2 变迁扩展：$t \leftrightarrow (t_1;\,([p, \lambda]\,;\,t_2))$，其中 $Var(t) \subseteq \lambda$

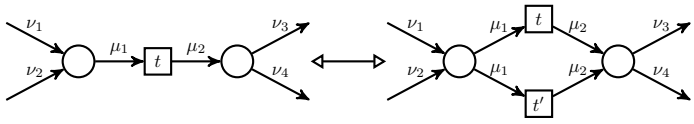


R3 库所复制：$(t_1; ([p, \lambda]\,; t_2)) \leftrightarrow (t_1; (([p, \lambda]\,k\,[p', \lambda'])\,; t_2))$，其中 $\lambda' \cap$ 发射$(p^\bullet) = \emptyset$

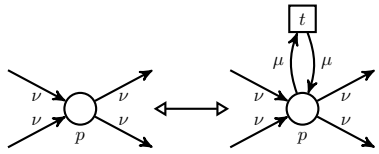


[5] 在此，我们将 ψ 限制为仅包含类型为 λ 且仅被消耗的对象。

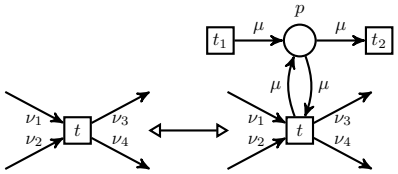
R4 变迁复制： $t \leftrightarrow (t + t')$



R5 自环添加*:* $[p, \lambda] \leftrightarrow ([p, \lambda] \# t)$



R6 标识符引入： $t \leftrightarrow (t \triangleleft (N_1, [p, \lambda], N_2))$，其中 $(N_1; ([p, \lambda]; N_2))$ 是一个 *t-JN*， $\lambda \cap Var(t) = \emptyset$



◁

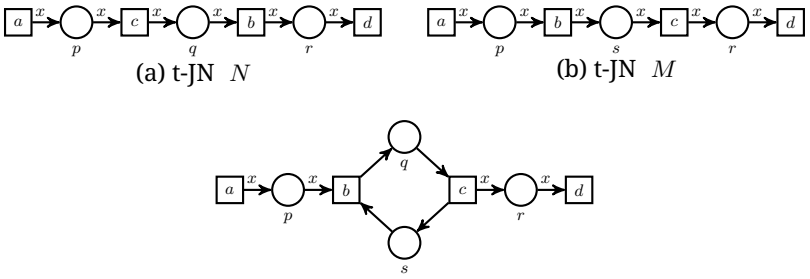
图1给出了一个 t-JN 示例。从产品过程开始，可以使用规则 $R2$ 化简变迁 C 和 D 。得到的变迁是一个自环变迁，可以使用 $R5$化简，得到块 ($E \triangleleft (A$, 乘积 $B)$)。使用 $R6$可以化简该块，留下变迁 E。变迁 E 再次是一个自环，可以使用 $R5$化简。通过反复应用规则 $R1$、 $R2$ 和 $R5$ ，包含变迁 H、 J、 L O、 N 和 K 的块可以化简为一个库所。剩余库所相对于库所 p是一个重复库所，可以使用 $R3$化简。在 G和 Z 上应用 $R2$ ，得到块 ($G \triangleleft (T$,客户 $V)$)，该块可以化简为变迁 G。因此，图1中的网是一个原子 t-JN。

定理 2（类型化 Jackson 网的标识符健全性 [23]）。给定一个 *t-JN* N，则 N 是标识符健全且活。◁

4 t-JN 的可分解性

t-PNID 指定了一类网，其中明确规定了单个系统内不同类型对象之间的交互。然而，有时人们可能只想关注给定对象类型集合所表现出的某些行为，并从原始类型化 Petri 网标识符定义模型中提取相应的网。下面我们将这一想法形式化。

定义 12（类型投影）。设 $N = (P_N, T_N, F_N, \alpha, \beta)$ 为一个 *t-PNID*， $\Upsilon \subseteq \Lambda$ 为一组标识符类型。 Υ 在 N 上的类型投影是一个 *t-PNID* $\pi_\Upsilon(N) = (P_\Upsilon, T_\Upsilon, F_\Upsilon, \alpha_\Upsilon, \beta_\Upsilon)$，其中：



(a) t-JN N (b) t-JN M

(c) t-PNID $N \uplus M$

图 4：尽管 N 和 M 都是 t-JN，但它们的组合并非 t-JN

– $P_\Upsilon = \{p \in P \mid \Upsilon \subseteq \alpha(p)\}$； – $T_\Upsilon = \{t \in T \mid (^\bullet t \cup t^\bullet) \cap P \neq \emptyset\}$； – $F_\Upsilon = F \cap ((P_\Upsilon \times T_\Upsilon) \cup (T_\Upsilon \times P_\Upsilon))$；– 对每个 $p \in P_\Upsilon$，$\alpha_\Upsilon(p) = \Upsilon$；– 对每个 $f \in ((P_\Upsilon \times T_\Upsilon) \cup (T_\Upsilon \times P_\Upsilon))$，$\beta_\Upsilon(f) = \beta(f)|_{\mathtt{type}^{-1}}$ $V(\Upsilon)$。 ◁

接下来的引理将探讨类型化 Jackson 网的一项性质。简言之，该性质表明 t-JN 在类型投影下是封闭的。这也间接说明，t-JN 为描述和操控由多个相互通信的组件构成的系统提供了一种合适的形式化方法。

引理 1。如果 $N = (P_N, T_N, F_N, \alpha, \beta)$ 是一个 *t-JN*，那么对于任意 $\Upsilon \subseteq \mathtt{type}_\Lambda(N)$，$\pi_\Upsilon(N)$ 也是一个 *t-JN*。◁

证明。（概要）为简便起见，假设 N 是原子的。然后，利用定义 11 中的规则，可以将 N 化简为单个变迁。从该变迁出发，可以按照定义 12 中的网图构造方法，使用相同的规则（但不使用标识符引入规则）构造一个 t-JN，前提是弧标注始终为 Υ 类型。接着，很容易验证所构造的网确实是 Υ 在 N 上的类型投影。 ■

接下来，我们定义如何组合 t-PNID，并说明 t-JN 在组合运算下不封闭。

定义 13（组合）。设 $N = (P_N, T_N, F_N, \alpha_N, \beta_N)$ 和 $M = (P_M, T_M, F_M, \alpha_M, \beta_M)$ 是两个 *t-PNID*。它们的组合定义如下：

$$N \uplus M = (P_N \cup P_M, T_N \cup T_M, F_N \cup F_M, \alpha_N \cup \alpha_M, \beta_N \cup \beta_M)$$

显然，两个 t-JN 的组合并不会自动产生一个 t-JN。请看图 4 中的网。显然，可以通过应用定义 11 中的 R2 得到 N 和 M。然而，无法通过连续应用定义 11 中的规则，将它们的组合化简为单个变迁。

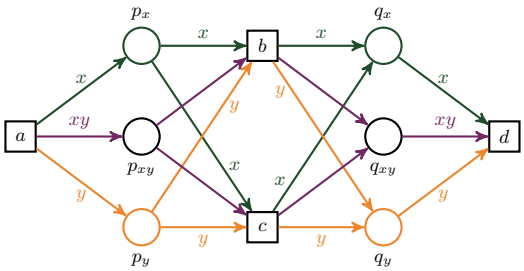


图5：在 t-JN $(a;\ [p, \mathrm{h}x, y\mathrm{i}];\ (b||c);\ [q, \mathrm{h}x, y\mathrm{i}];\ d)$ 上，$\{\lambda_1\}$、$\{\lambda_2\}$和$\{\lambda_1, \lambda_2\}$的投影的组合。这里，类型赋值如下：$\alpha(p_x) = \alpha(q_x) = \lambda_1$, $\alpha(p_y) = \alpha(q_y) = \lambda_2$ 以及 $\alpha(p) = \alpha(q) = \lambda_1\lambda_2$。

一个更令人意外的观察是，对 t-JN 的类型投影进行组合，可能不会得到 t-JN。以图 5 中的网为例。它在 $\{\lambda_1\}$ 和 $\{\lambda_2\}$ 上的投影都是 t-JN。然而，使用组合算子将它们合并后，得到的 t-PNID 并不是 t-JN：事实上，由于库所 p 的“副本”出现在三个库所中，且所有这些副本具有相同的前集和后集（并且仅类型各异），因此无法应用定义 11 中的标识符消除规则 $R6$。

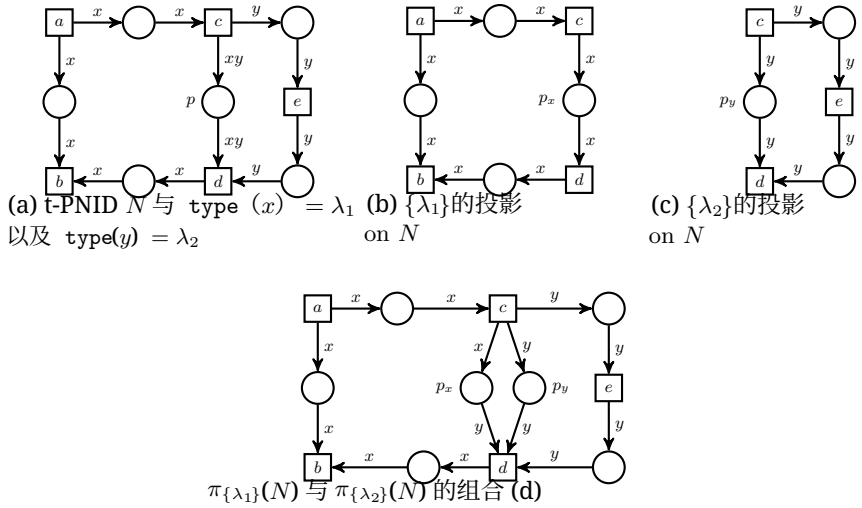
从上面的例子可以看出，$[p_{xy}, \langle \lambda_1, \lambda_2 \rangle]$ 与其副本 p_x 和 p_y 之间唯一的差异在于各自的类型，而 p_x 和 p_y 所携带的标识符始终包含在 p_{xy}中，因此，p_x 和 p_y 都可以视为相对于 p_{xy}的次要库所。我们用次要库所这一概念将这一观察形式化：如果库所 p 和库所 q 具有相同的前集和后集，且 q 的类型涵盖 p的类型，那么库所 p 相对于库所 q 是次要的。

定义 14（次要库所）。设 $N = (P_N, T_N, F_N, \alpha, \beta)$ 是一个 *t-PNID*。当且仅当满足以下条件时，库所 $p \in P$ 相对于库所 $q \in P$ 是次要的：
– ${}^\bullet p = {}^\bullet q$, $p^\bullet = q^\bullet$ 和 $\alpha(p) \subset \alpha(q)$; – $\beta((t, p)) = \beta((t, q))|_{\mathtt{type}^{-1}(\alpha(p))}$, 对每个 $t \in {}^\bullet p$ 都成立；– $\beta((p, t)) = \beta((q, t))|_{\mathtt{type}^{-1}(\alpha(p))}$, 对每个 $t \in p^\bullet$ 都成立。 ◁

接下来我们证明，添加或移除次要库所不会改变网的整体行为。

引理 2. 设 $N = (P, T, F, \alpha, \beta)$ 是一个初始标识为 m_0 的 *t-PNID*，满足 $m_0(p) = m_0(q) = \emptyset$，其中 $p, q \in P$，且 p 相对于 q 是次要的。设 $N' = (P \setminus \{p\}\ T, F \setminus (\{(p, t) | t \in p^\bullet\} \cup \{(t, p) | t \in {}^\bullet p\})\ \alpha, \beta)$ 是从 N 中消去库所 p 后得到的 *t-PNID*。那么 $\Gamma_{N, m_0} \sim^r \Gamma_{N', m_0}$. ◁

证明。（概要）只需定义一个关系 $Q \subseteq \mathcal{R}(N, m_0) \times \mathcal{R}(N', m_0)$，使得 $(m, m') \in Q$ 当且仅当 $m(r) = m'(r)$，其中 $r \in P \setminus \{p\}$；并且对于所有 $\mathtt{id} \in \mathtt{C}(p)$，$m(p)(\mathtt{id}) = m'(q)(\mathtt{id})$，且 $|m(p)| = |m'(q)|$。于是，根据 t-PNID 的变迁规则以及 p 和 q 的前集和后集相同这一点，可直接推出引理的结论。 ■



(a) t-PNID N 与 $\texttt{type}(x) = \lambda_1$ 以及 $\texttt{type}(y) = \lambda_2$ (b) $\{\lambda_1\}$的投影 on N (c) $\{\lambda_2\}$的投影 on N

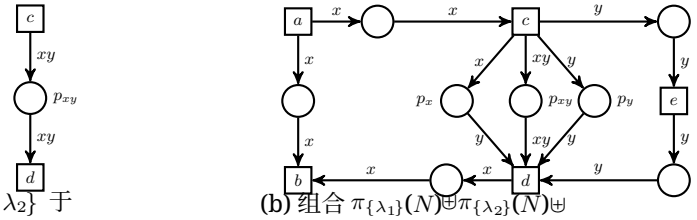
$\pi_{\{\lambda_1\}}(N)$ 与 $\pi_{\{\lambda_2\}}(N)$ 的组合 (d)

图 6：t-PNID N (6a)、其单例投影及其组合

现在来讨论可重构性。简而言之，如果一个网的所有类型投影组合后得到的仍是同一个网，那么这个网就是可重构的。要满足这一性质并非易事。例如，考虑图 6 中该网的单例投影（即，对每个 $\lambda \in \texttt{type}_\Lambda(N)$ 得到的投影 $\pi_{\{\lambda\}}(N)$）。不难看出，这类投影会“忽略”对象（或系统组件）之间的交互。因此，将图 6 中的单例投影 $\pi_{\{\lambda_1\}}(N)$ 和 $\pi_{\{\lambda_2\}}(N)$ 组合起来，并不会得到一个将 p_x 和 p_y 合并到同一库所的模型，因为组合算子无法识别这类投影之间的组件交互。图 6d 展示了这一点。

为了能够从原模型的投影中重构出原模型（或至少近似地重构），需要考虑一种反映组件间交互的投影。对于图 6a 中的网，其非单元素投影 $\pi_{\{\lambda_1,\lambda_2\}}(N)$ 如图 7a 所示。现在，利用该投影，我们可以得到一个与 N 非常相似的组合（见图 7b）。请注意，在这个组合中，交互库所的副本 p 分别以库所p_x、p_y 和 p_{xy}的形式出现了三次。同样容易看出，库所 p_x 和 p_y 是 p_{xy}的次要部分，而 $\alpha(p) = \alpha(p_{xy})$ 证明了 $\pi_{\{\lambda_1,\lambda_2\}}(N)$ 是定义在 N 的类型上的最大投影，使得 p 的正确类型得以“重构”。由此得到以下结果，该结果阐明了类型化 Jackson 网的可重构性。

定理 3. 设 $N = (P,\ T,\ F,\ \alpha,\ \beta)$ 是一个 ***t-JN***。那么 $\Gamma_{N,\ \emptyset} \sim^r \Gamma_{N',\ \emptyset}$，其中$N' = \biguplus_{\emptyset \subset \Upsilon \subseteq \texttt{type}_\Lambda(N)} \pi_\Upsilon(N)$。 ◁



(a) 投影 $\{\lambda_1, \lambda_2\}$ 于 N 图 6a

(b) 组合 $\pi_{\{\lambda_1\}}(N) \uplus \pi_{\{\lambda_2\}}(N) \uplus \pi_{\{\lambda_1,\lambda_2\}}(N)$（ N 图 6a）

图 7：将反映交互的投影 $\pi_{\{\lambda_1,\lambda_2\}}(N)$ 加入组合后，得到的结果与原始网 N 相同，但允许存在对 p 而言次要的库所（例如 p_x 和p_y）。

证明。（概要）证明可直接由以下观察得出。在所有可能的投影中，对每个库所 $p \in P$，都存在一个投影 $\pi_{\Upsilon}(N)$，使得 $\alpha(p) = \Upsilon$。这也意味着 $\pi_{\Upsilon}(N)$ 包含 p，而所有其他满足 $\Upsilon' \subset \Upsilon$ 的投影 $\pi_{\Upsilon'}(N)$ 至多包含 p 的次要部分。根据定义 13，很容易看出，所有投影的组合会得到一个与 N 相同的 t-JN，区别仅在于某些投影引入了额外的库所次要部分。证明所得的网与 N 双模拟等价，可以仿照引理 2 进行。

■

请注意，如果使用定义 11 中的化简规则移除所有额外的次要库所（即原本不存在于 N中的次要库所），则上述结果可以得到加强。为简单起见，给定一个 t-PNID N 及其库所集合 P，我们用 $\lfloor P \rfloor$ 表示其次要库所集合。

推论 1. 设 N为一个 *t-JN*，且 N'如定理 *3* 中所述。那么 $(N, \emptyset) \leftrightsquigarrow (N', \emptyset)$，若 $\lfloor P \rfloor = \lfloor P' \rfloor$，其中 P和 P'分别是 N和 N'的库所集合。◁

上述结果可通过补充定理 3 的证明得到：增加一个步骤，对 N' 中而不在 N中的所有次要库所，有限次应用 t-JN 化简规则。

5 可再发现性框架

在上一节中，我们证明了 t-JN 具有可重构性：给定一个 t-JN，将其所有 （真）类型投影进行组合，可得到一个与原网强双模拟等价的 t-JN。[6]

本节提出一种框架，用于重新发现依赖此属性的交互进程系统。该框架基于分而治之策略 [21]。该方法的第一步是将事件日志划分

[6] 若依次应用定义 11 中的化简规则，移除组合中的次要库所，则这些网也是同构的。

表1：图1中 t-PNID 的发射序列

变迁 x y z			
A	$p1$		
A	$p2$		
T			$c1$
G		$o1$	$c1$
C	$p1$		
E	$p2$	$o1$	

变迁 x y z			
T			$c2$
H		$o1$	
L		$o1$	
J		$o1$	
B	$p2$		
O		$o1$	

变迁 x y z			
D	$p1$		
V			$c2$
K		$o1$	
Z		$o1$	$c1$
V			$c1$
B	$p1$		

到所有可能的投影上。为此，我们将事件日志的概念拓展为交互系统的事件日志，并指出，如果这些事件日志由 t-JN 生成，那么对这些事件日志进行投影便具有一项特殊性质：投影后的事件日志可以由投影网重放。换言之，对事件日志进行投影，与由投影网生成该事件日志，两者并无区别。这一观察构成了所提出的可再发现性框架的基础。第二步是解决交互进程系统的发现性问题：首先为每个投影发现一个模型，然后将这些投影组合成原始系统。如果事件日志和发现算法能够保证所定义的性质，那么组合便能实现可再发现性。

5.1 事件日志与执行轨迹

在流程发现中，事件日志表示为事件序列（称为轨迹）的一个（多重）集合，其中每个序列都表示一个流程实例的执行历史。传统的流程发现假设流程是一个工作流网。因此，事件日志中的每条轨迹都应对应于该工作流网的一个变迁发射序列。若满足这一条件，则称该事件日志由该工作流网生成。我们将这一概念推广到有标记的 Petri 网。

定义 15（事件日志）。给定一个变迁集合 T，轨迹集合 $L \subseteq T^*$ 称为事件日志。若有标记的 *Petri* 网 (N, m) 生成事件日志 L，则 $N, m)[\sigma i$ 对所有 $\sigma \in L$ 均成立，即 $L \subseteq L(N, m_0)$。◁

单个流程事件日志中的每个序列都假定从工作流网的初始标识开始。而标记 t-PNID 表示一个持续执行的系统；对于给定的具体标识，该系统存在一次可记录在事件日志中的可观测执行。因此，事件日志是系统中更大范围执行的部分观测：某一类型的事件日志只捕获包含该类型标识符的相关事件，并按其执行顺序记录这些事件。由于每次变迁触发都由一个变迁和一个绑定组成，因此 t-PNID 的发射序列会为每个类型集 Υ诱导出一个事件日志。直观地说，这个诱导事件日志通过过滤过程构造而成。对于 Υ 的每个可能标识符向量，我们都保留一个发射序列。逐一检查每次变迁触发；如果其绑定满足 Υ的某个标识符向量，就将其加入相应的序列中。

定义 *16*（诱导事件日志）。设 (N, m_0) 为一个标记 *t-PNID*。给定一个非空类型集 $\Upsilon \subseteq \mathtt{type}_\Lambda(N)$，发射序列 $\eta \in L(N, m_0)$ 的 Υ-诱导事件日志定义为：$Log_\Upsilon(\eta) = \{\eta_{|i} \mid i \in (Id(\eta) \cap I(\Upsilon))^{|\Upsilon|}\}$，其中 $\eta_{|i}$ 由以下规则归纳定义：(1) $\epsilon_{|i} = \epsilon$；(2) $(h(t,\psi)i \cdot \eta)_{|i} = h(t,\psi)i \cdot \eta_{|i}$，若 $supp(i) \subseteq rng(\psi)$；以及 (3) $(h(t,\psi)i \cdot \eta)_{|i} = \eta_{|i}$，否则。$\triangleleft$

发射序列可以诱导出不同的事件日志。例如，考虑图1中的网的发射序列，如表 1 所示。由于无法从表 1 的发射序列中推断出每个变量的类型，我们假设变量与类型之间存在双射，即每个变量都由其类型唯一标识，反之亦然。这样，我们可以为每个变量创建一个诱导日志，因为类型和变量名可以互换。例如，x 诱导的事件日志为 $\mathrm{Log}_{\{x\}} = \{\langle A, E, B\rangle, \langle A, C, D, B\rangle\}$，而 z 诱导的事件日志为 $\mathrm{Log}_{\{z\}} = \{\langle T, G, Z, V\rangle, \langle T, V\rangle\}$。类似地，也可以针对类型组合诱导事件日志。在此例中，组合类型上唯一非空的诱导事件日志为 $\mathrm{Log}_{\{y,z\}} = \{\langle G, Z\rangle\}$ 和 $\mathrm{Log}_{\{x,y\}} = \{\langle E\rangle\}$。

如表 1 中的发射序列所示，变迁发射（因此也包括事件）只显示变量与标识符之间的绑定。例如，对于绑定为 $y \mapsto o1$ 和 $z \mapsto c1$ 的发射 G，无法直接从表中推导出已消耗和产生的令牌的类型。因此，我们对 t-PNID 的流程发现作出以下假设：

1. 不存在“黑色”令牌：所有库所都携带至少具有一种类型的令牌，并且每种类型在一个库所类型中至多出现一次，即所有库所都指向至少一个流程实例。
2. 变量与类型之间存在双射，即每种类型恰好使用一个变量。 3. 使用类哥德尔数 $\mathscr{G}$ 对库所类型中的类型排序，即对于任意库所 p，对于 $1 \le i < j \le |\alpha(p)|$ 和 $p \in P$，有 $\mathscr{G}(\alpha(p)(i)) < \mathscr{G}(\alpha(p)(j))$。

5.2 类型化 Jackson 网络的可再发现性

传统的流程发现方法将事件日志中的事件关联到单个对象：流程实例；而以对象为中心的方法可以将事件关联到多个对象 [11]。大多数以对象为中心的流程发现算法（例如 [5,17]）采用分治方法，其中“扁平化”是将事件数据拆分为较小事件日志的默认实现。扁平化操作为数据集中的每个对象创建一条轨迹，并将相同类型对象的轨迹合并到一个事件日志中。正如我们在第4节中所展示的，单例投影，即仅单独考虑各类型的投影，不足以重构导致该以对象为中心的事件日志的 t-JN。以对象为中心的流程发现也有类似的观察结果（参见 [3,5,7]）：将事件数据扁平化为事件日志会生成不准确的模型。相反，只有考虑类型的所有可能组合，才能实现可重构性。因此，对于分而治之策略，划分步骤应涉及类型的所有可能组合，也就是说，每种进程间交互

都需要各自的事件日志。在本节余下部分，我们将展示，只要考虑类型的所有组合，就可以进行扁平化，并且可以使用传统的流程发现算法重新发现交互进程系统。

对于一个交互进程系统，我们考虑执行轨迹，即从初始标识开始的发射序列。同样，由该发射序列可以导出特定类型或类型组合对应的事件日志。将系统投影到某个类型或类型组合上，得到的仍然是一个 t-JN。类似地，若将 t-JN N 的一条发射序列投影到类型集合 Υ 上，那么该投影就是 Υ-投影在 N 上的一条发射序列。这个性质直接源于如下结果：t-JN N 被其 Υ-投影弱模拟。

引理 3. 设 N 是一个 *t-JN*，且 $\Upsilon \subseteq \mathtt{type}_\Lambda(N)$。则 $\hat{}\ \mathtt{H}_U(\Gamma_{N,\emptyset}) \preccurlyeq^r \Gamma_{\pi_\Upsilon(N)\ \emptyset}$，其中 $U = T_N \setminus T_\Upsilon$。◁

证明。（概要）设 $N_\Upsilon = \Upsilon_{|N} = (P_\Upsilon, T_\Upsilon, F_\Upsilon, \alpha_\Upsilon, \beta_\Upsilon)$。我们可以定义一个关系 $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(\pi_\Upsilon(N))$，使得若 $p \in P_\Upsilon$，则 $Q(m)(p)(a_{|I(\Upsilon)}) = m(p)(a)$；否则，$Q(m)(p) = m(p)$。$Q$ 的根弱双模拟直接由 t-PNID 的变迁规则得出。■

正如引理所示，对发射序列进行投影后，会得到投影网的发射序列。模拟关系的直接推论是，无论是从原网的发射序列诱导事件日志，还是从投影后的发射序列诱导事件日志，所得事件日志都相同。

推论 2. 设 (N, m_0) 是一个标记 *t-PNID*。给定类型集 $\Upsilon \subseteq \mathtt{type}_\Lambda(N)$。则 $Log_\Upsilon(\eta) = Log_\Upsilon(\pi_\Upsilon(\eta))$。◁

因此，无法观察出某个诱导事件日志究竟源自原模型还是其投影。请注意，投影可能呈现更多行为，因此反向结论并不成立。一般而言，并非所有由投影诱导的事件日志都能由原模型诱导。

一般而言，投影不一定是原子 t-JN（即可以通过应用定义 11 中的规则约简为单个变迁的 t-JN）。然而，如果投影是原子的，那么其结构就是变迁边界 WF 网：这种 WF 网不含源库所和汇库所，而是包含一组起始变迁和结束变迁，且起始变迁的前集（分别地，结束变迁的后集）为空。变迁边界 WF 网的闭包通过添加一个新的源库所 i 构造，使每个起始变迁都从 i 消耗；再添加一个新的汇库所 f，使每个结束变迁都向 f 产生。

引理 *4.* 设 N 是一个 *t-JN*，且 $\pi_\Upsilon(N) = (P_\Upsilon, T_\Upsilon, F_\Upsilon, \alpha_\Upsilon, \beta_\Upsilon)$，其中某个 $\Upsilon \subseteq \mathtt{type}_\Lambda(N)$ 满足 $\pi_\Upsilon(N)$ 是原子的。设 $\eta \in L(N, \emptyset)$ 是一个发射序列。那么，$Log_\Upsilon(\eta)$ 由 $(N_\Upsilon, \emptyset)$ 生成，其中 $N_\Upsilon = (P_\Upsilon \cup \{i, f\}\ T_\Upsilon, F_\Upsilon \{(i, t)\ |\ {}^\bullet t = \emptyset\} \cup \{(t, f)\ |\ t^\bullet = \emptyset\})$。◁

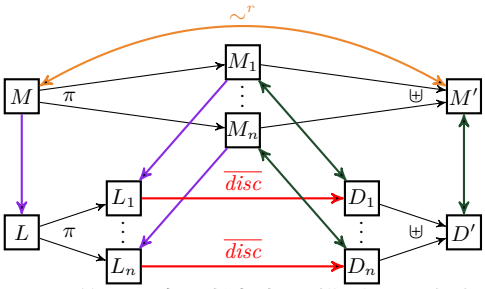


图 8：类型化 Jackson 网的可再发现性框架。模型 M 生成事件日志 L。日志投影 $L_1 \ldots L_n$ 由投影网 $M_1 \ldots M_n$生成。发现算法 $disc$得出网 $D_1 \ldots D_n$，它们与 $M_1 \ldots M_n$同构，并可在 D'中组合。 D' 与 M' 同构，因此也与 M同构。

证明。 （略）令 $\sigma \in Log_\Upsilon(\eta)$。根据构造，$Log_\Upsilon(\eta)$ 中的每个发射序列都有一个生成该序列的对应标识符向量。假设$\vec{v} \in \mathcal{I}^{|\Upsilon|}$ 是 σ的这样一个向量。

注意，对于任意变迁 $t \in T$，若 $^\bullet t = \emptyset$，则 Emit$(t) \cap \Upsilon \neq \emptyset$；类似地，若 $t^\bullet = \emptyset$，则 Collect$(t) \cap \Upsilon \neq \emptyset$。由于 N是标识符健全的，因此只有 $^\bullet\sigma(1) = \emptyset$和 $\sigma(|\sigma|)^\bullet = \emptyset$。定义关系 $R = \{(M, m) \mid \forall p \in P\colon M(p)(v) = m(p)\}$ 和$U = \{(t, \psi) \mid v \not\subseteq \text{rng}(\psi)\}$，即 U 包含所有不属于σ的变迁。于是 R是弱模拟，也就是说，$\hat{} \; \mathrm{H}_U(\Gamma_{N,\emptyset}) \preccurlyeq_R^r \Gamma_{N_\Upsilon,\emptyset}$，因此 $(N_\Upsilon, \emptyset)[\sigma\mathbf{i}_\circ$ ■

给定一个类型集 Υ，如果其投影是原子的，那么该投影可以转换为一个工作流网；对于原网的任意发射序列，这个工作流网都能生成由 Υ 诱导的事件日志。假设我们有一个发现算法 disc，能够重新发现模型；也就是说，给定由某个模型 M 生成的事件日志 L，disc 会返回原始模型。算法的可再发现性要求生成模型 M 满足某种性质 $P_{disc}(M)$，并且事件日志 L 相对于生成模型 M 的质量满足某种性质 $Q_{disc}(L, M)$。换言之， $P(M)$ 和 $Q(L, M)$ 是得出发现算法 disc 具有可再发现性的前提。例如， α-miner [22] 要求模型 M 满足 $P(M)$，即模型是良构的；同时要求事件日志 L 相对于模型 M 满足 $Q(L, M)$，即事件日志是直接后继完备的。类似地，Inductive Miner [16] 要求生成模型 M 是不含静默动作或自循环的过程树（$P(M)$），并要求事件日志 L 相对于原始模型 M 直接后继完备（$Q(L, M)$）。

定义 17（重新发现）。算法 *disc* 可以从事件日志 $L \subseteq T^*$ 中重新发现工作流网 $W = (P, T, F, in, out)$，如果 $P_{disc}(W)$ 和 $Q_{disc}(L, W)$ 蕴含 $disc(L) \leftrightsquigarrow W$。◁

因此，假设存在一个发现算法 $disc$，在条件 P 和 Q 下能够根据事件日志重构工作流模型。在

换言之，给定由某个模型 M生成的事件日志 L ，$disc$ 会返回一个与生成模型同构的模型。现在，假设我们有某个 t-JN N的发射序列 η，以及某个投影 Υ。那么，如果 $P(\pi_\Upsilon(N))$，且 Q(日志$_\Upsilon(\eta)$ $\pi_\Upsilon(N))$，则 $disc$会返回一个与 $\pi_\Upsilon(N)$的闭包同构的模型，因为 $disc$只返回工作流网。我们用 $disc$表示移除了源库所和汇库所后的模型，即 $disc \nleftrightsquigarrow \pi_\Upsilon(N)$。然后，如图 8 所示，如果针对每一种可能的类型组合（即所有类型组合构成的子集闭合集）进行发现，并得到一个与类型投影模型同构的模型，那么组合后得到的模型就与原始模型双模拟等价。

定理 *4*（类型化 *Jackson* 网的可再发现性）。设 N 是一个 *t-JN*，且 $\eta \in L(N, \emptyset)$ 不含次要库所。设 *disc* 是一个满足定义 *17* 中性质 P 和 Q 的发现算法。如果对于所有 $\emptyset \subset \Upsilon \subseteq \mathtt{type}_\Lambda(N)$，$\Upsilon$-投影都是原子的，并满足条件 $P(\pi_\Upsilon(N))$和 Q *(Log*$_\Upsilon(\eta))$ $\pi_\Upsilon(N))$，则$\Gamma_{N,\emptyset} \nleftrightsquigarrow \Gamma_{N',\emptyset}$ 满足$N' = \biguplus_{\emptyset \subset \Upsilon \subseteq \mathtt{type}_\Lambda(N)}$ *disc(Log*$_\Upsilon(\eta))$。

证明。（概要）设 $\emptyset \subset \Upsilon \subseteq \mathtt{type}_\Lambda(N)$是 N中的一个类型集。由于 $P(\pi_\Upsilon(N))$且 Q(Log$_\Upsilon(\eta))$ $\pi_\Upsilon(N))$，$\pi_\Upsilon(N)$的闭包与 disc(Log$_\Upsilon(\eta)$)同构。从该闭包中可得输入和输出库所，满足 $^\bullet\mathsf{in} = \emptyset = \mathsf{out}^\bullet$。由于这些网同构，因此有 $\Upsilon_{|N} \nleftrightsquigarrow$ disc(Log$_\Upsilon(\eta)$)。结合这些结果可得$\biguplus_{\emptyset \subset \Upsilon \subseteq \mathtt{type}_\Lambda(N)}$ disc(Log$_\Upsilon(\eta)$) $\nleftrightsquigarrow$ $\biguplus_{\emptyset \subset \Upsilon \subseteq \mathtt{type}_\Lambda(N)}$ $\pi_\Upsilon(N)$。结论直接由推论 1 得出。 ■

6 结论

本文研究了用于建模交互进程系统的类型化 Jackson 网，这是一类描述对象标识符操作的良结构过程模型。正如我们所展示的，这类网具有一项重要性质，即可重构性。换言之，对所有可能类型组合的投影进行组合，即可还原原始系统的模型。忽略进程之间的交互会导致模型不够准确，甚至错误。在发现交互进程系统时也会出现类似问题，例如以对象为中心的流程发现，其中事件日志会按对象展平。

本文为块结构网的组合提供了形式化基础，并以此构建了一个发现交互进程系统的框架。我们将流程发现中使用的事件日志概念与系统执行联系起来，并证明无法观察到事件日志究竟是由交互进程系统生成，还是由该系统的投影生成。这些性质构成了该框架的关键要素。我们阐明了在何种条件下，可以使用能够保证重新发现性的流程发现算法来发现单个进程及其交互，以及如何将这些进程及交互组合起来，重新发现一个与生成事件日志的原始系统双模拟等价的交互进程模型。

尽管类型化 Jackson 网的表达能力不如对象中心佩特里网 [5],过程小网 [10] 或交互工件 [17],，本文揭示了发现交互进程时的局限性与潜在陷阱。本研究旨在为以对象为中心的流程发现奠定形式化基础。下一步，我们计划实现该框架，并调整算法，以便从工业数据集中发现有用的模型。

致谢。 Artem Polyvyanyy 部分获得了澳大利亚研究委员会项目 DP220101516 的支撑集。

参考文献

1. van der Aalst, W.M.P.：《工作流验证：使用基于佩特里网的技术发现控制流错误》。载于：计算机科学讲义，第161–183页。Springer Berlin Heidelberg（2000）。https://doi.org/10.1007/3-540-45594-9_11 https://doi.org/10.1007/3-540-45594-9%5F11 , . van der Aalst, W.M.P.：《工作流网的验证》。载于：Petri网。LNCS，第1248， 407 426 1997 10 1007 3 540 63139 9_48 卷，第–3页。Springer（）。https://doi.org/。/- - -。van der Aalst, W.M.P.：《对象中心流程挖掘：处理事件数据中的发散与收敛》。载于：SEFM 2019，第3–25页。LNCS第11724号，Springer（2019）4. van der Aalst, W.M.P.：《流程发现基础》。载于：流程挖掘手册。LNBIP，第448卷。Springer（2022）5. van der Aalst, W.M.P., Berti, A.：《发现对象中心佩特里网》。Fundamenta Informaticae **1**–**4**(175), 1–40（2020）6. van der Aalst, W.M.P., van Hee, K.M., ter Hofstede, A.H.M., Sidorova, N., Verbeek, H.M.W., Voorhoeve, M., Wynn, M.T.：《工作流网的健全性：分类、可判定性与分析》。Formal Asp. Comput. **23**(3), 333–363（2011）7. Adams, J.N., Park, G., Levich, S., Schuster, D., van der Aalst, W.M.P.：《从对象中心事件数据中提取和编码特征的框架》。载于：ICSOC 2022，第36–53页。LNCS第13740号，Springer（2022）8. Berti, A., van der Aalst, W.M.P.：《OC-PM：分析对象中心事件日志和流程模型》。International Journal on Software Tools for Technology Transfer（2022年9月）。https://doi.org/10.1007/s10009-022-00668-w，https://doi.org/10.1007/s10009-022-00668-w9. Best, E., Devillers, R., Koutny, M.：《Box Algebra、Petri网与过程表达式》。Information and Computation **178**(1), 44–100（2002年10月）。https://doi.org/10.1006/inco.2002.3117，https://doi.org/10.1006/inco.2002.3117 10. Fahland, D.：《用多对多交互描述流程行为》。载于：Petri网与并发的应用和理论，第3–24页。Springer International Publishing（2019）。https://doi.org/10.1007/978-3-030-21571-2_1, https://doi.org/10.1007/978-3-030-21571-2%5F1 11 . Ghahfarokhi, A.F., Park, G., Berti, A., van der Aalst, W.M.P.：《OCEL：对象中心事件日志的标准》。载于：ADBIS 2021。CCIS，第1450卷，第169–175页。Springer（2021）12. van Glabbeek, R.J.：《线性时间—分支时间谱 II》。载于：CONCUR'93，第66–81页。Springer Berlin Heidelberg（1993）。https://doi.org/10.1007/3-540-57208-2_6, https://doi.org/10.1007/3-540-57208-2%5F6

13. van Hee, K., Sidorova, N., Voorhoeve, M.: 《逐步细化方法下工作流网的可靠性与可分离性》。载于：Petri网的应用与理论 2003，第337–356页。Springer Berlin Heidelberg（2003）。https://doi.org/10.1007/3-540 44919 1_22 https://doi.org/10.1007/3-540-44919-1%5F22 --, 14. van Hee, K.M., Hidders, J., Houben, G.J., Paredaens, J., Thiran, P.: 《工作流模型与文档类型之间的关系》。信息系统 **34**(1), 178–208（2009年3月）。https://doi.org/10.1016/j.is.2008.06.003 https://doi.org/10.1016/j.is.2008.06.003 15. Kopp, O., Martin, D., Wutke, D., Leyman, F.: 《基于图的业务流程建模语言与块结构业务流程建模语言之间的差异》。EMISAJ **4**(1), 3–13（2009）16. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: 《从事件日志中发现块结构流程模型——一种构造性方法》。载于：Petri网与并发的应用和理论，第311–329页。Springer Berlin Heidelberg（2013）。https://doi.org/10.1007/978-3-642-38697-8_17, https://doi.org/10.1007/978-3-642-38697-8%5F1717. Lu, X., Nagelkerke, M., van de Wiel, D., Fahland, D.: 《从ERP系统中发现交互工件》。IEEE Trans. Serv. Comput. **8**(6), 861–873（2015）18. Murata, T.: 《Petri网：性质、分析与应用》。IEEE会刊 **77**(4), 541–580（1989年4月）。https://doi.org/10.1109/5.24143, https://doi.org/10.1109/5.24143 19. Polyvyanyy, A., van der Werf, J.M.E.M., Overbeek, S., Brouwers, R.: 《信息系统建模：语言、验证与工具支持》。载于：CAiSE。计算机科学讲义，第11483卷，第194–212页。Springer（2019）20. Tour, A., Polyvyanyy, A., Kalenkova, A.A.: 《智能体系统挖掘：愿景、优势与挑战》。IEEE Access **9**, 99480–99494（2021）21. Tour, A., Polyvyanyy, A., Kalenkova, A.A., Senderovich, A.: 《智能体挖掘器：一种从事件数据中发现智能体系统的算法》。CoRR **abs/2212.01454**（2022）22. van der Aalst, W., Weijters, T., Maruster, L.: 《工作流挖掘：从事件日志中发现流程模型》。知识与数据工程 **16**(9), 1128–1142（2004）23. van der Werf, J.M.E.M., Rivkin, A., Polyvyanyy, A., Montali, M.: 《数据与流程共振》。载于：Petri网与并发的应用和理论，第369–392页。Springer International Publishing（2022）。https://doi.org/10.1007/978-3-031-06653-5_19, https://doi.org/10.1007/978-3-031-06653-5%5F19