arXiv:2305.05113v1 [cs.DB] 9 May 2023

对象中心对齐

Lukas Liss、Jan Niklas Adams 和 Wil M.P. van der Aalst

德国亚琛，亚琛工业大学流程与数据科学讲席 lukas.liss@rwth-aachen.de，{niklas.adams,wvdaalst}@pads.rwth-aachen.de

摘要。 流程往往会与其他流程交互，并处理不同类型的各种对象。这些对象可能相互影响，从而在子流程之间形成依赖关系。分析此类复杂流程的符合性，对传统的符合性检查方法提出了挑战，因为这些方法假设一个流程只有一个案例标识符。要创建单一案例标识符，就必须将复杂流程扁平化。当分离那些通过某些对象发生交互的流程时，这会导致信息丢失。本文介绍了一种可直接应用于此类对象中心流程的对齐方法。我们提出了对齐，可据此了解基于行为的洞见：流程生成的事件数据与对象中心佩特里网所规定的行为之间有多大关联。本文的贡献包括对象中心对齐的定义、用于计算对象中心对齐的算法、公开可用的实现，以及定性和定量评估。定性评估表明，对象中心对齐能够更深入地揭示对象中心流程，因为它们能够正确考虑对象间依赖。定量评估的结果显示，运行时间会随对象数量、流程执行长度以及对齐成本呈指数增长。评估结果促使未来研究致力于缩短运行时间，使对象中心对齐更适用于规模更大的流程。

关键词： 流程挖掘 · 对象中心过程挖掘 · 对齐。

1 引言

流程挖掘通过分析流程生成的事件数据，为流程提供洞见。在分析流程时，一种标准流程包括提取数据、发现流程模型，以及检查流程是否符合规范 [13]。本文提出了一种计算对齐的方法，用于检查对象中心流程的符合性，弥补传统符合性检查方法无法提供正确洞见的不足。

传统流程挖掘方法依赖于这样一种假设：流程由单一案例概念定义，即为一个对象创建的所有操作共同构成一次流程执行。现实世界中的流程往往并不符合这一

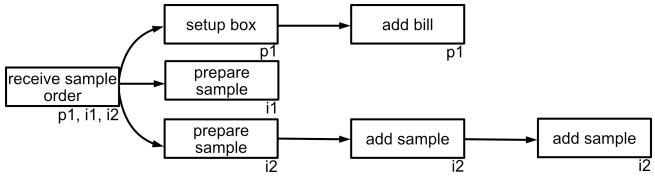

图1. 贯穿示例中的一次流程执行。事件与包类型（前缀 p）或项目类型（前缀 i）的对象相关联。该流程执行描述了由各个对象所引发的事件偏序。

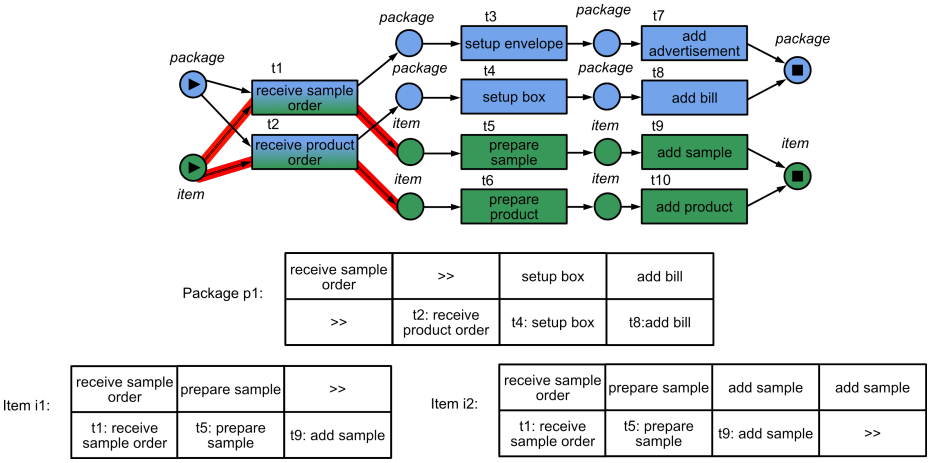

图2. 上：作为对象中心佩特里网的规范模型。模型中的变量弧以红色标出。下：图1中的流程执行被展平为各个对象，并与规范模型中对应对象类型的子网进行对齐。

假设。典型的供应链流程就是一个例子。供应流程涉及原材料，生产流程涉及原材料和产品，发货流程涉及产品和订单，而支付流程则涉及订单和客户。一次供应链执行并非由单个对象定义。现实世界的流程由多个子流程构成，这些子流程作用于多个不同类型的对象实例。不同对象之间的这些子流程可能存在同步点和长期依赖。

近年来，人们提出了一些方法来泛化流程的概念，以便描述这些复杂流程。这些方法统称为对象中心过程挖掘 [20]。对象中心过程挖掘对传统流程挖掘技术进行了泛化，使得分析不再是沿着一个流程跟踪一个对象，而是沿着多个相互连接的子流程跟踪多个对象。这种泛化增加了复杂性。在传统流程挖掘中，一次流程执行称为案例，并由事件序列定义 [1]。对象中心的流程执行是一个图，用于描

述不同子流程中事件之间的偏序 [7]。迄今为止，针对对象中心一致性检查，人们提出了一种基于重放的拟合度概念，但该方法仅适用于可重放的行为[5]。然而，流程所有者通常关注的是将观察到的行为与建模行为对齐，并找出流程执行中的偏差，即使用对齐 [8]。迄今为止，对象中心过程挖掘中对齐的概念、计算方法、实现及可行性分析仍付阙如。

本文使用的贯穿示例描述了一个具有跨对象依赖关系的包装流程。在我们的示例中，一次流程执行涉及多个项目和一个包。图1将一次流程执行描绘为事件图，展示项目对象与包对象之间的偏序。图2展示了包装流程的对象中心佩特里网 [4] 。对象中心佩特里网与传统 Petri 网的区别在于引入了库所类型和可变弧。令牌具有类型，同一类型的库所只能容纳相同类型的令牌。可变弧可以消耗任意数量的令牌。库所按项目（绿色）或包（蓝色）进行类型划分。此外，为了清晰起见，我们用参与某个变迁的类型对应的颜色标示该变迁，并用红色突出显示可变弧。在给定的流程中，包的路径取决于项目的路径，反之亦然。包和项目可以属于样品订单或产品订单，但不能同时属于两者，具体取决于是否触发接收样品订单 或接收产品订单 。尽管这些对象在开始后不再共享任何事件，但它们允许的行为仍取决于它们属于样品订单还是产品订单。

如果流程所有者如今想要发现其对象中心流程中的偏差，就需要扁平化 [2] 观察到的流程执行，并将传统对齐应用于对象中心佩特里网中类型相同的子网。我们在图2中展示了图1所示贯穿示例流程执行的这一过程。若将其扁平化为每个对象一条轨迹，得到的三条轨迹会分别与该类型的子网对齐，但这种对齐方式在组合模型中并不可行。如前所述，活动接收样品订单和接收产品订单不可能同时发生在一次流程执行中，因为规范模型要求在产品订单和样品订单之间作出选择。但扁平化后的对齐结果对应该发生哪种活动并不一致。$p1$ 的对齐在模型部分包含接收产品订单，而$i1$ 和$i2$ 的对齐则在模型部分包含接收样品订单。贯穿示例表明，对对象中心流程进行对齐，需要的不只是分别为每个对象寻找对齐。通过尊重对象之间的依赖关系来对齐所有对象的子流程，会形成一个计算复杂的问题，本文将对此进行探讨。

本文提出了四项贡献，以支持并研究以对象为中心的对齐。首先，我们将对齐的概念推广到以对象为中心的流程。其次，我们提出了一种用于计算最优以对象为中心的对齐的算法。第三，我们实现了该算法，并将其作为一个开源项目公开提供[1] ，该项目基于开源的以对象为中心的

[1] https://github.com/LukasLiss/object-centric-alignments

流程挖掘库 OPCA [6]。第四，我们在真实世界的事件数据上评估对象中心对齐的质量和计算时间。由此，我们深入了解该方法的可扩展性和适用性。

本文结构如下。第 2 节介绍相关工作，第 3 节介绍预备知识。随后，我们在第 4 节定义以对象为中心的对齐。我们用于计算对齐的算法由两部分组成：构造同步积网（第 5 节），以及在同步积网中寻找最优对齐（第 6 节）。第 7 节介绍定性和定量评估，并在第 8 节总结全文。

2 相关工作

流程挖掘包括业务流程的发现、一致性检查和增强 [1]。我们的方法属于一致性检查范畴，即将事件日志中的行为与规范模型所规定的允许行为进行比较 [13]。对于传统流程，已有多种一致性检查方法 [19]。其中大多数采用基于令牌的重放方法 [25]，或对齐方法 [8]。这两种方法都曾用于推导精确度 [9] 和拟合度 [10] 等质量指标。与基于令牌的重放不同，对齐不依赖规范模型的结构 [8]。与我们的方法类似，Adriansyah 等人定义的传统对齐计算采用两步算法来计算对齐 [8]。Adriansyah 等人的方法构造一个同步积网，使得寻找最优对齐可转化为在该网中寻找最短路径。这是一个经过充分研究的问题，可以用迪杰斯特拉 [16] 算法或 A^* [14] 算法求解。研究者已探索多种加快计算的方法 [18][27]。然而，该对齐算法假设流程只有一个案例标识符，因此无法用于由多个对象参与的流程组合。

多年来，人们对传统对齐算法进行了多种扩展，将工作流维度与其他维度一并纳入考虑 [11][12]。为此，这些方法使用高阶网来表示附加维度。de Leoni 等人提出的数据与资源感知一致性检查方法使用数据佩特里网 [23]。Felli 等人将数据佩特里网与可满足性模理论结合起来，计算数据感知对齐[21]。Sommers 等人构建了一种 ν-佩特里网，用于计算受资源约束的对齐 [26]。然而，上述所有方法都假设流程只有一个案例标识符。

有一些方法能够拓展这一概括，并将流程建模为相互作用的子流程。多智能体流程模型通过组合 Petri 网来描述智能体的行为及其交互 [24]。对象中心流程挖掘 [2] 扩展了流程的概念，使其能够相互交互，并操作不同类型的对象。Adams 等人将对象中心流程中的案例和变体定义为事件图，而非事件序列 [7]。所定义的流程执行可作为我们计算对齐的输入。我们使用的另一种输入数据类型是对象中心 Petri 网

[4] 它能够描述允许的行为。可以使用 van der Aalst 和 Berti 提出的发现算法，从对象中心事件日志中发现对象中心 Petri 网 [4]。近期，有人提出了用于评估模型质量的精确度和拟合度指标 [5]。然而，目前仍缺少检查流程是否符合规范模型并发现偏差的技术，例如对象中心对齐。

3 预备知识

对象中心过程挖掘研究的是作用于多种不同类型对象的事件。事件是指在某个时间戳发生、涉及若干不同类型对象的活动。 $\mathbb{U}_{event}$ 是事件标识符的全集。全集 $\mathbb{U}_{act}$ 包含所有可见活动。 $\mathbb{U}_{typ}$ 是所有对象类型的全集。对象全集为 $\mathbb{U}_{obj}$。每个对象恰好关联一种类型 $\pi_{type} : \mathbb{U}_{obj} \to \mathbb{U}_{typ}$。 $\mathbb{U}_{time}$ 是所有时间戳的全集。

定义 1（事件日志）。 $L = (E, O, OT, \pi_{act}, \pi_{obj}, \pi_{time}, \pi_{trace})$ 是一个事件日志，其包含:

– $E \subseteq \mathbb{U}_{event}$ 是事件集合， $O \subseteq \mathbb{U}_{obj}$ 是对象集合，– $OT = \{\pi_{type}(o) \mid o \in O\}$ 是对象类型集合，– $\pi_{act} : E \to \mathbb{U}_{act}$ 将每个事件映射到一个活动，– $\pi_{obj} : E \to \mathcal{P}(\mathbb{U}_{obj}) \setminus \{\emptyset\}$ 将每个事件映射到至少一个对象，– $\pi_{time} : E \to \mathbb{U}_{time}$ 将每个事件映射到一个时间戳，且– $\pi_{trace} : O \to E^*$ 将每个对象映射到一个事件序列，使得$\pi_{trace}(o) = \langle e_1, ..., e_n \rangle$ 与 $\{e_1, ..., e_n\} = \{e \in E | o \in \pi_{obj}(e)\}$ 以及 $\forall_{i \in \{1,...,n-1\}} \pi_{time}(e_i) \leq \pi_{time}(e_{i+1})$

事件日志可能包含来自多个流程执行的事件。分析行为时，我们希望提取出一个流程执行。

定义 2（流程执行）。设 $L = (E, O, OT, \pi_{act}, \pi_{obj}, \pi_{time}, \pi_{trace})$ 是一个对象中心事件日志。对象图 $OG_L = (O, I)$ 中的 $I = \{\{o, o'\} \exists_{e \in E} \{o, o'\} \subseteq \pi_{obj}(e) \wedge o \neq o'\}$ 连接了共享事件的对象。对象图 $OG_L\}$ 中的连通分量 $con(L) = \{X \subseteq O | X$ 是相互依赖对象的集合。每个集合 $X \in con(L)$ 定义了 L 的一个流程执行。流程执行是一个图 $P_X = (E_X, D_X)$，其节点为 $E_X = \{e \in E | X \cap \pi_{obj}(e) \neq \emptyset\}$，边为 $D_X = \{(e, e') \in E_X \times E_X | \exists_{o \in X, 1 \leq i < n} \langle e_1, ..., e_n \rangle = \pi_{trace}(o) \wedge e = e_i \wedge e' = e_{i+1}\}$。集合 $px(L) = \{P_X | X \in con(L)\}$ 包含事件日志 L 的所有流程执行。

图1展示了贯穿示例的流程执行。对象中心 Petri 网使用类型来描述对象中心行为，例如彩色 Petri 网 [22]。

$\mathcal{B}(A)$ 用于表示集合 A 的所有多重集。给定集合 A 的多重集 M，元素 $a \in A$ 在 M 中的实例数为 $M(a)$。我们重载符号 $M = [a^k | a \in A]$，表示多重集 M 中元素 a 有 k 个实例。

定义 3（对象中心佩特里网 [4]）。对象中心佩特里网是一个元组 $ON = (N, pt, F_{var})$，其中 $N = (P, T, F, l)$ 是一个带标签的佩特里网，包含库所 P 和变迁 T。$F \in \mathcal{B}((P \times T) \cup (T \times P))$ 是库所与变迁之间的弧多重集。变迁标记为活动，或由 $l: T \to \mathbb{U}_{act} \cup \{\tau\}$ 标记为不可见活动 $\tau \notin \mathbb{U}_{act}$。$pt: P \to \mathbb{U}_{typ}$ 将库所映射到对象类型，而 $F_{var} \leq F$ 是变量弧的子多重集。

注意，我们使用函数 l 将所有变迁标记为活动或 τ。对象中心佩特里网的其他常见定义将 l 定义为偏函数。通过假设对于所有没有标签的 t，$l(t) = \tau$，即可将其转换为我们的定义。我们为对象中心佩特里网定义以下派生记法

- $\bullet t = \{p \in P | (p, t) \in F\}$ 是变迁 $t \in T$ 的前集。- $t\bullet = \{p \in P | (t, p) \in F\}$ 是变迁 $t \in T$ 的后集。- $pl(t) = \bullet t \cup t\bullet$ 是 $t \in T$ 的输入和输出库所，$pl_{var}(t) = \{p \in P | \{(p, t), (t, p)\} \cap F_{var} \neq \emptyset\}$ 是通过变量弧连接的库所，而 $pl_{nv}(t) = \{p \in P | \{(p, t), (t, p)\} \cap (F \setminus F_{var}) \neq \emptyset\}$ 是通过非可变弧连接的库所。- $tpl(t) = \{pt(p) | p \in pl(t)\}$，$tpl_{var}(t) = \{pt(p) | p \in pl_{var}(t)\}$，以及 $tpl_{nv}(t) = \{pt(p) | p \in pl_{nv}(t)\}$ 是与变迁相关的对象类型。

图2展示了贯穿示例中的对象中心佩特里网。它包含两种对象类型，以及用于变迁接收样品订单 和接收产品订单的变量弧。

定义 4（良构面向对象 Petri 网 [4]）。设 $ON = (N, pt, F_{var})$ 是一个对象中心佩特里网，其 $N = (P, T, F, l)$。当且仅当对于每个变迁 $t \in T$：$tpl_{var}(t) \cap tpl_{nv}(t) = \emptyset$ 时，ON 是良构的。

在良构的对象中心佩特里网中，与同一变迁相连且属于相同对象类型的库所的弧，要么全部是变量弧，要么全部都不是变量弧。以下我们假设所用的所有对象中心佩特里网都是良构的。与彩色 Petri 网类似，对象中心佩特里网使用标识和绑定的概念来描述佩特里网的语义。

定义 5（对象中心 Petri 网的标识 [4]）。设 $ON = (N, pt, F_{var})$ 是一个对象中心佩特里网，其 $N = (P, T, F, l)$。$\mathcal{Q}_{ON} = \{(p, o) \in P \times \mathbb{U}_{obj} | pt(p) = \pi_{type}(o)\}$ 是所有可能令牌的集合。ON 的标识 M 是一个令牌多重集 $M \in \mathcal{B}(\mathcal{Q}_{ON})$。

绑定描述了哪个变迁会触发，以及每种对象类型会消耗和产生哪些对象实例。

定义 6（对象中心佩特里网 [4] 的绑定）。设 $ON = (N, pt, F_{var})$ 是一个对象中心佩特里网，其具有 $N = (P, T, F, l)$。所有可能绑定的集合为 $B = \{(t, b) \in T \times (\mathbb{U}_{type} \not\to \mathcal{P}(\mathbb{U}_{obj})) | dom(b) = tpl(t) \wedge \forall_{ot \in tpl_{nv}(t)} \forall_{p \in pl_{nv}(t)\ pt(p) = ot} | b(ot)| = F(p, t)\}$。绑定 $(t, b) \in B$ 对应于在佩特里网 ON 中触发变迁 t。对象映射 b 描述了哪些对象实例被消耗和产生。给定绑定的已消耗令牌多重集

绑定 $(t, b) \in B$ 为 $cons(t, b) = [(p, o) \in \mathcal{Q}_{ON} | p \in \bullet t \wedge o \in b(pt(p))]$。给定绑定 $(t, b) \in B$ 时，已产生令牌的多重集为 $prod(t, b) = [(p, o) \in \mathcal{Q}_{ON} | p \in t\bullet \wedge o \in b(pt(p))]$。

如果 $cons(t, b) \leq M$，则在标识 $M \in \mathcal{B}(\mathcal{Q}_{ON})$ 中启用绑定 $(t, b) \in B$。在标识 M 中应用绑定 (t, b)，会得到新标识 $M' = M - cons(t, b) + prod(t, b)$。我们用记法 $M \xrightarrow{(t,b)} M'$ 表示在 M 中应用 (t, b)。这意味着 (t, b) 在 M 中已启用，且 M' 是在 M 中应用 (t, b) 的结果。

此记法可扩展到绑定序列 $\sigma = \langle (t_1, b_1), (t_2, b_2) ..., (t_n, b_n) \rangle \in B^*$，使得 $M_0 (t_1, b_1) \longrightarrow M_1 (t_2, b_2) \longrightarrow M_2 ... (t_n, b_n) \longrightarrow M_n$。我们用记法 $M \xrightarrow{\sigma} M'$ 表示，通过依次应用 σ 中的绑定，可以从 M 到达 M'。变迁可通过标签函数 l 映射到活动。由此得到可见绑定序列 $\sigma_v = \langle (l(t_1) \ b_1), (l(t_2) \ b_2) ..., (l(t_n) \ b_n) \rangle$；若 $l(t_i) = \tau$，则省略 $(l(t_i) \ b_i)$。

定义 7（接受型对象中心佩特里网 [4]）。接受型对象中心佩特里网是一个元组 $AN = (ON, M_{init}, M_{final})$，其中 $ON = (N, pt, F_{var})$是一个良构面向对象 *Petri* 网。$M_{init} \in \mathcal{B}(\mathcal{Q}_{ON})$ 和 $M_{final} \in \mathcal{B}(\mathcal{Q}_{ON})$分别表示该网的初始标识和最终标识。

接受型对象中心佩特里网接受某些绑定序列，但不接受另一些。所有被接受的绑定序列构成一个语言。

定义 8（接受型 Petri 网的语言 [4]）。接受型对象中心 *Petri* 网 $AN = (ON, M_{init}, M_{final})$ 的语言$\phi(AN) = \{\sigma_v | M_{init} \xrightarrow{\sigma} M_{final}\}$ 包含所有以 M_{init} 开始并以 M_{final} 结束的可见绑定序列。

4 对齐

对齐展示了流程执行与规范模型允许的行为之间的关系。我们使用移动来表示某个事件发生在流程执行中、规范模型中，还是两者中。请注意，我们不允许对齐改变对象集合。我们假设对象集合是固定的。

定义9（移动）。设 $L = (E, O, OT, \pi_{act}, \pi_{obj}, \pi_{times}, \pi_{trace})$ 为一个对象-中心化事件日志和 $P_X = (E_X, D_X) \in px(L)$ 一次流程执行。设 $AN = (((P, T, F, l) pt, F_{var}) M_{init}, M_{final})$ 为一个接受型对象中心 *Petri* 网。所有移动的集合为 $moves(P_X, AN) \subseteq (\{\pi_{act}(e) | e \in E_X\} \cup \{\gg\}) \times \mathcal{P}(O) \times (T \cup \{\gg\}) \times \mathcal{P}(O)$，其中跳过符号为 $\gg \notin \mathbb{U}_{act} \cup T$。移动 $(a_{log}, o_{log}, t_{mod}, o_{mod}) \in moves(P_X, AN)$ 属于以下三种类型之一：

日志移动 - 对于 $e \in E_X$：$a_{log} = \pi_{act}(e)$、$o_{log} = \pi_{obj}(e)$、$t_{mod} = \gg$ 和$o_{mod} = \emptyset$。

模型移动 - 针对 $(t, b) \in \sigma$，包含 $\sigma_v \in \phi(AN)$：$t_{mod} = t$, $o_{mod} = \bigcup_{o \in range(b)} o$、$a_{log} = \gg$和 $o_{log} = \emptyset$。

同步移动 - 对于 $e \in E_X$和 $(t, b) \in \sigma$，其中包含 $\sigma_v \in \phi(AN)$：$a_{log} = \pi_{act}(e) = l(t)$、$t_{mod} = t$，以及 $o_{log} = o_{mod} = \pi_{obj}(e) = \bigcup_{o \in range(b)} o$。

图3展示了每种移动类型。对于同步移动，模型部分和日志部分的活动及对象必须完全相同。对于日志移动和模型移动，只有一部分包含活动和对象，而另一部分则被跳过。跳过符号 $\gg$ 表示该部分中没有发生任何事情。上方部分是日志部分 a_{log} 和 o_{log}。下方的区块是模型部分，其中包含 t_{mod}、其标记的活动 l（t_{mod}）以及 o_{mod}。

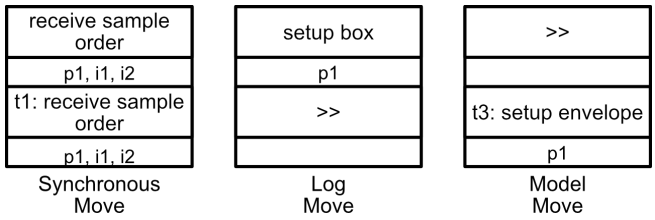

图3. 图2中的接受型对象中心 Petri 网与图1中的流程执行示例所对应的一个同步移动、一个日志移动和一个模型移动。

我们定义如下移动投影。

定义10（移动投影）。

给定一个移动 $m = (a_{log}, o_{log}, t_{mod}, o_{mod}) \in moves(P_X, AN)$，其流程执行为 P_X，接受型对象中心佩特里网为 $AN = (((P, T, F, l)\ pt, F_{var}), M_{init}, M_{final})$。我们使用以下投影将移动映射到其属性：$\pi_{la}(m) = a_{log}$ 将移动映射到其日志活动。$\pi_{lo}(m) = o_{log}$ 将移动映射到其日志对象。$\pi_{mt}(m) = t_{mod}$ 将移动映射到其模型变迁。$\pi_{ma}(m) = l(t_{mod})$ 将移动映射到该变迁所标记的活动。$\pi_{mo}(m) = o_{mod}$ 将移动映射到其模型对象。

在分别推理模型或日志的行为时，我们希望忽略该部分中被跳过的行为。对齐（我们在定义 12 中给出定义）是由移动构成的有向无环图。为了定义对齐，我们需要分别推理模型和日志的行为。因此，我们引入以下约简：从由移动构成的有向无环图中移除给定部分中包含被跳过行为的移动，同时保留该无环图所定义的偏序。

定义11（约简到日志部分和模型部分）。令 $MG = (M, C)$ 为一个有向无环图，其顶点为 $M \subseteq moves(P_X, AN)$，边为 $C \subseteq M \times M$，流程执行为 P_X，接受型对象中心 *Petri* 网为 AN。对日志部分中具有可见活动的移动进行约简，得到 $MG_{\downarrow log} = (M_{\downarrow log}, C_{\downarrow log})$，其中：

– $M_{\downarrow log} = \{m \in M | \pi_{la}(m) \neq \gg\}$ 同步移动和日志移动。– $C_{\downarrow log} = \{(m_1, m_n) \in M_{\downarrow log} \times M_{\downarrow log} | \exists_{<m_1,\ldots,m_n> \in M^*} \forall_{1 \leq i < n} (m_i, m_{i+1}) \in C \wedge \forall_{1<i<n} \pi_{la}(m_i) = \gg\}$ 同步移动和日志移动之间的边，以及移除模型移动后新增的边。

模型部分中具有可见活动的移动的约简为 $MG_{\downarrow mod} = (M_{\downarrow mod}, C_{\downarrow mod})$，其中包括：

– $M_{\downarrow mod} = \{m \in M | \pi_{ma}(m) \neq \gg\}$ 同步移动和模型移动– $C_{\downarrow mod} = \{(m_1, m_n) \in M_{\downarrow mod} \times M_{\downarrow mod} | \exists_{<m_1,\ldots,m_n> \in M^*} \forall_{1 \leq i < n} (m_i, m_{i+1}) \in C \wedge \forall_{1<i<n} \pi_{ma}(m_i) = \gg\}$ 同步移动和模型移动之间的边，以及移除日志移动后新增的边。

图 4 同时展示了任意移动有向无环图中的 $MG_{\downarrow log}$和 $MG_{\downarrow model}$。$MG_{\downarrow log}$ 描述了移除所有模型移动及相关边后的有向无环图。当移动图中的两个移动原本通过被移除的模型移动相连时，会添加新的边。$MG_{\downarrow model}$ 在模型部分中也会同时发挥作用。

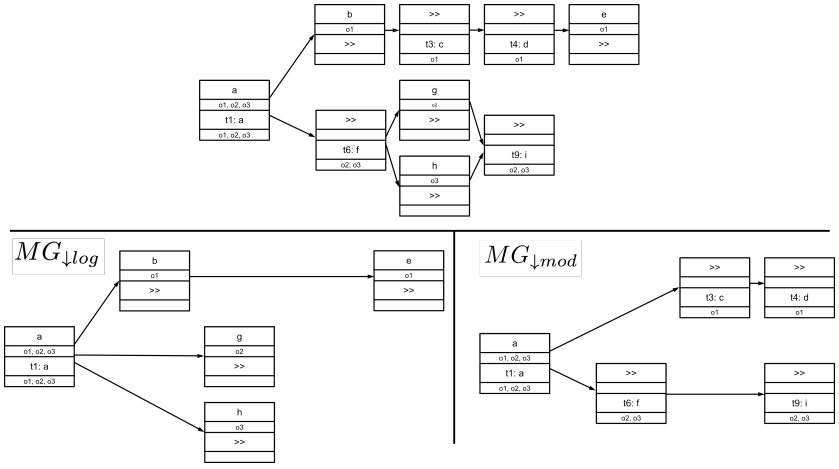

图 4. 移动图的有向无环图的约简 $MG_{\downarrow log}$ 和 $MG_{\downarrow model}$。

对齐是由移动构成的有向无环图，它要求日志部分包含流程执行，而模型部分使用规范模型的语言。

定义*12*（对齐）。设 $L = (E, O, OT, \pi_{act}, \pi_{obj}, \pi_{times}, \pi_{trace})$ 是一个对象中心事件日志，$P_X = (E_X, D_X) \in px(L)$ 是一次流程执行。设$AN = (((P, T, F, l)\ pt, F_{var})M_{init}, M_{final})$ 是一个接受型对象中心 *Petri* 网。对齐 $AL_{P_X,AN} = (M, C)$ 是 $M \subseteq moves(P_X, AN)$ 上的一个有向无环图，满足：

对齐在日志部分中包含流程执行的行为： P_X 与 $AL_{P_X,\ AN_{\downarrow log}}$ 同构，且存在双射 f： $E_X \to M_{\downarrow log}$，使得 $\forall_{e \in E_X} \pi_{act}(e) = \pi_{la}(f(e)) \wedge \pi_{obj}(e) = \pi_{lo}(f(e))$。

该对齐包含模型部分中被佩特里网接受的行为：存在一个绑定序列 $\sigma = \langle (t_1, b_1), (t_2, b_2) ..., (t_n, b_n) \rangle \in B^*$，其中 $\sigma_v \in \phi(AN)$，以及一个双射函数 f': $B \to M_{\downarrow mod}$，满足：

- $\forall_{(t,b) \in \sigma} t = \pi_{mt}(f'(t,b)) \wedge \bigcup_{o \in range(b)} o = \pi_{mo}(f'(t,b))$
- $\forall_{m_1, m_2 \in M_{\downarrow mod}} (m_1, m_2) \in C_{\downarrow mod} \Rightarrow \exists_{1 \le i < j \le n} m_1 = f'(t_i, b_i) \wedge m_2 = f'(t_j, b_j)$

对于一次流程执行和一个接受型对象中心 *Petri* 网，可能存在多个对齐。 $al(P_X, AN)$ 是所有这些对齐的集合。

贯穿示例的一种对齐如图 5 所示。移动图是有向无环的，由此形成移动的偏序。将该图约化为日志移动和同步移动后，它与图1中的给定流程执行同构。这确保了给定流程执行的行为包含在对齐中。约化到模型部分后，对应于一个必须属于接受型对象中心 Petri 网语言的绑定序列。这确保了模型部分描述的是模型所接受的行为。请注意，对齐对模型部分没有提出任何其他要求，只需该部分为模型所允许即可。因此，对齐也可能包含与流程执行非常不同的模型行为。这类对齐最终会包含更多模型移动和日志移动，以及更少的同步移动。移动的成本函数使我们能够搜索模型中希望与之对齐的特定行为。大多数时候，我们寻找的是模型所允许的、与给定流程执行最相似的行为。这正是标准成本函数所形式化的内容。

定义 13（移动函数的标准成本）。
设 $AL_{P_X,\ AN} = (M, C) \in al(P_X, AN)$ 是一个流程执行 P_X 与接受型对象中心 *Petri* 网 AN 之间的对齐。成本函数 $cost_{move} : moves(P_X, AN) \to \mathbb{R}$ 定义如下：

$$cost_{move}(m) = \begin{cases} 0 & \text{若 } m \text{ 是同步移动，} |\pi_{lo}(m) \cup \pi_{mo}(m)| \text{ 若 } m \text{ 是模型移动或日志移动} \wedge \pi_{ma}(m) \neq \tau, \\ \varepsilon & \text{若 } \pi_{ma}() = \tau \wedge a_{log} = \gg, \\ m & \text{否则} \end{cases} +\infty$$

其中 ε 是一个正的极小数。完整对齐的成本是所有对齐移动的成本之和：
$cost_{alignment}(AL_{P_X,\ AN}) = \sum_{m \in M} cost_{move}(m)$。

图 5 中对齐的成本为 6，因为其中有 3 个模型移动和 3 个日志移动，且每个移动都涉及一个对象。成本越低，模型部分与日志部分的匹配越好，因为同步移动的成本低于模型移动和日志移动。我们将成本最低的对齐之一称为最优对齐。

定义 14（最优对齐）。设 L 为对象中心事件日志， $P_X \in px(L)$ 为流程执行。设 AN 为一个接受型对象中心

佩特里网。如果对齐 $AL_{P_X,\ AN} = (M, C) \in al(P_X, AN)$ 满足 $\forall_{a \in al(P_X, AN)}$ $cost_{alignment}(AL_{P_X,\ AN}) \leq cost_{alignment}(a)$，则称其为最优对齐。

请注意，对于给定的流程执行和规范佩特里网，仍可能存在多个最优对齐，只要它们与该流程执行的相似程度相同。

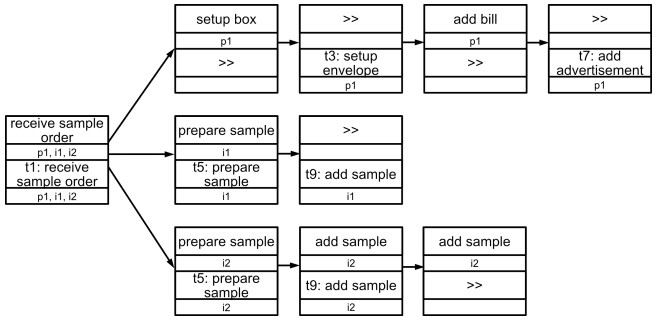

图 5. 图1中的流程执行与图2中的接受型对象中心 Petri 网之间的对齐。

贯穿示例的一个最优对象中心对齐见图 5。对象中心对齐遵循图2中的对象中心佩特里网所定义的对象间依赖。例如，对象中心对齐为$p1$、$i1$和$i2$采用了同一个共享起始活动，从而使对齐符合对象间依赖。这一区别将对象中心对齐与可能违反此要求的传统对齐区分开来。

5 对象中心同步积网

我们计算最优对齐方法的第一步，是为给定的流程执行和接受型对象中心佩特里网创建一个同步积网。该同步积网旨在生成所有可能的对齐。由于对齐可能包含模型移动、日志移动和同步移动，同步积网由与这些移动直接对应的三个部分组成。图 8 所示的同步积网中标出了这三个部分。首先，我们根据流程执行构建日志部分。然后，对规范模型进行预处理，最后将两者合并为同步积网，并添加同步部分。

5.1 流程执行网的构建

流程执行网是同步积网的一部分，用于保证流程执行包含在对齐中。流程执行网的构造与佩特里网运行 [15] 和因果关系

网 [3] 有关。对于每个对象，流程执行中的每条边都为一个事件定义了一个前置条件。因此，图 6 中的流程执行网为流程执行中定义的每个条件都设置了一个库所。

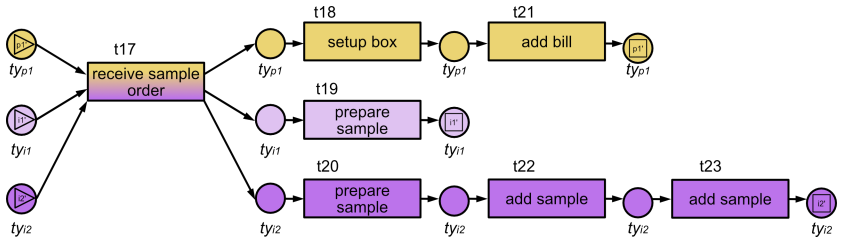

图6. 流程执行网

由于对象中心 Petri 网的定义不够充分，而且我们将把创建的网与给定的接受型对象中心 Petri 网合并，因此日志部分和模型部分不能使用相同的对象和类型。流程执行包含对象实例层面的直接跟随 关系。例如，图1中的贯穿示例流程执行表明，在项目*i1* 上从未发生添加样本，而在项目$i2$ 上，添加样本活动发生了两次。虽然它们具有相同的对象类型，但流程执行会对它们加以区分。在这方面，接受型对象中心 Petri 网的定义不够充分，因为它只按类型区分库所。显然，将对象实例分开很重要，否则，一个对象实例中的偏差可能会被另一个对象实例中的相反偏差抵消。例如，如果不严格区分每个事件发生在哪个对象实例上，那么*i1* 上缺失的添加样本 活动就可能被*i2* 上额外的添加样本活动抵消。为了使接受型对象中心 Petri 网能够区分对象实例，我们必须为流程执行中的每个对象实例创建一个新的独立类型。由于每个对象实例必须恰好具有一种类型，因此我们需要在流程执行网中用新的独立占位对象实例替换原有对象实例。

定义 *15*（新对象和类型的生成）。设 $L = (E, O, OT, \pi_{act}, \pi_{obj}, \pi_{times}, \pi_{trace})$ 是一个对象中心事件日志，且 $P_X = (E_X, D_X) \in px(L)$ 是一个包含对象 X 的流程执行。设 $O_{new}(P_X) \in \mathbb{U}_{obj}$ *with* $O_{new}(P_X) \cap O = \emptyset$ *and* $|O_{new}(P_X)| = |X|$ 是一组新的对象实例，并且 $OT_{new}(P_X) \in \mathbb{U}_{type}$ *with* $OT_{new}(P_X) \cap \{\pi_{type}(o) | o \in X\} = \emptyset$ *and* $|OT_{new}(P_X)| = |X|$ 是一组新的类型。

令 new_{obj} ： $X \to O_{new}$（P_X）为一个双射函数，它将对象重命名为 O_{new}（P_X）中的未使用对象。令 new_{type} ： $X \to OT_{new}$（P_X）为一个双射函数，将每个对象关联到其唯一的新类型。

我们定义以下派生概念：$orob$ ： $O_{new}(P_X) \to X$ 使用 $orob = new_{obj}^{-1}$ 返回原始对象。

$orty$ ： $OT_{new}(P_X) \rightarrow \{\pi_{type}(o)|o \in X\}$ 使用 $orty(ot) = \pi_{type}(new^{-1}_{type}(ot))$ 返回与给定新类型关联的对象的原始类型。

我们使用新的对象和类型创建流程执行网。流程执行中定义的条件是每个对象的直接跟随关系。佩特里网为每个条件包含一个库所。此外，还为每个对象添加了起始库所和结束库所。变迁对应于流程执行中的事件。

定义 16（流程执行网）。令 $L = (E, O, OT, \pi_{act}, \pi_{obj}, \pi_{time}, \pi_{trace})$ 为一个对象中心事件日志。令 $P_X = (E_X, D_X) \in px(L)$ 为该事件日志中的一次流程执行。流程执行网 $PX_{net} = (((P, T, F, l)\ pt, F_{var})\ M_{init}, M_{final})$ 是一个具有以下特征的接受型对象中心 *Petri* 网：

- $P = \{p_{o;i}|o \in X \wedge \pi_{trace}(o) = \langle e_1, ..., e_n\rangle \wedge 1 \leq i \leq n-1\}$
  $\cup \{p_{o;s}|o \in X\} \cup \{p_{o;e}|o \in X\}$
- $T = \{t_e|e \in E_X\}$
- $F = [(t_e, p_{o;i}) \in T \times P|\pi_{trace}(o) = \langle e_1, ..., e_n\rangle \wedge \exists_{1 \leq i \leq n-1} e_i = e]$
  $\cup\, [(p_{o;i}, t_e) \in P \times T|\pi_{trace}(o) = \langle e_1, ..., e_n\rangle \wedge \exists_{1 \leq i \leq n-1} e_{i+1} = e]$
  $\cup\, [(p_{o;s}, t_e) \in P \times T|\pi_{trace}(o) = \langle e_1, ..., e_n\rangle \wedge e_1 = e]$
  $\cup\, [(t_e, p_{o;e}) \in T \times P|\pi_{trace}(o) = \langle e_1, ..., e_n\rangle \wedge e_n = e]$
- $l(t_e) = \pi_{act}(e)$; $pt(p_{o;i}) = new_{type}(o)$; $F_{var} = \emptyset$
- $M_{init} = \{(p_{o;s}, new_{obj}(o))|o \in X\}$; $M_{final} = \{(p_{o;e}, o')|new_{obj}(o) \in X\}$

5.2 对象中心佩特里网的预处理

法定行为已经通过一个接受型对象中心 Petri 网给出，这与我们创建同步积网所需的形式非常接近。但是，接受型对象中心 Petri 网可能包含可变弧。图2中的贯穿示例 Petri 网包含分别对应于 t_1 和 t_2 的可变弧。对于带有可变弧的变迁，我们无法预先知道它们将使用多少个对象。当我们尝试寻找流程执行网与法定网之间的同步移动时，这就成了一个问题。只有当两个变迁使用相同的对象实例时，它们才能同步，这意味着它们使用的对象实例数量也必须相同。因此，我们需要预先知道一个变迁将使用多少个对象实例。

如第4节所述，我们假设用于对齐的对象集合不可变，且由流程执行定义。对于预先定义的对象实例集合，可变弧能够使用的对象数量是有限的。因此，我们可以将带有可变弧的变迁替换为一组不带可变弧的变迁。对于可变弧可能消耗的每一种对象数量组合，我们都可以在佩特里网中添加一个新变迁，使其恰好使用相应数量的对象，同时将可变弧替换为若干条非可变弧。对图2中的佩特里网进行这一处理后，得到的就是图 7 中不带可变弧的佩特里网。

定义17（接受型对象中心*Petri*网预处理）。设 $L = (E, O, OT, \pi_{act}, \pi_{obj}, \pi_{times}, \pi_{trace})$ 是一个对象中心事件日志，$P_X = (E_X, D_X) \in px(L)$ 是一次流程执行。设 $AN = (((P, T, F, l)\ pt, F_{var}), M_{init}, M_{final})$ 是一个接受型对象中心 *Petri* 网。设类型 $ot_1, ..., ot_n \in OT$ 的顺序任意但固定。预处理后的接受型对象中心 *Petri* 网为 $DJ_{net} = (((P', T', F', l')\ pt', F'_{var}) M'_{init}, M'_{final})$，其中：

– $P' = P$, $pt' = pt, F'_{var} = \emptyset$, – $T' = \{t_{c_1,...,c_n} | t \in T \wedge \forall_{1 \le i \le n} 0 \le c_i \le |\{o \in X | \pi_{type}(o) = ot_i\}|$ 如果 $ot_i \in tpl_{var}(t)$ 否则 $c_i = 0\}$– $F' = [(p, t_{c_1,...,c_n}) \in P' \times T' | (p, t) \in F \wedge (p, t) \notin F_{var})] \cup [(t_{c_1,...,c_n}, p) \in T' \times P' | (t, p) \in F \wedge (t, p) \notin F_{var})] \cup [(p, t_{c_1,...,c_n})^k \in P' \times T' | (p, t) \in F_{var} \wedge \exists_{1 \le i \le n} pt(p) = ot_i \wedge c_i = k] \cup [(t_{c_1,...,c_n}, p)^k \in T' \times P' | (t, p) \in F_{var} \wedge \exists_{1 \le i \le n} pt(p) = ot_i \wedge c_i = k]$– $l'(t_{c_1,...,c_n}) = l(t)$ – $M'_{init} = [(p, o) \in P' \times X | \forall_{t \in T} (t, p) \notin F \wedge pt(p) = \pi_{type}(o)]$– $M'_{final} = [(p, o) \in P' \times X | \forall_{t \in T} (p, t) \notin F \wedge pt(p) = \pi_{type}(o)]$

我们将预处理后的接受型对象中心 Petri 网称为法定网。贯穿示例的法定网见图 7。

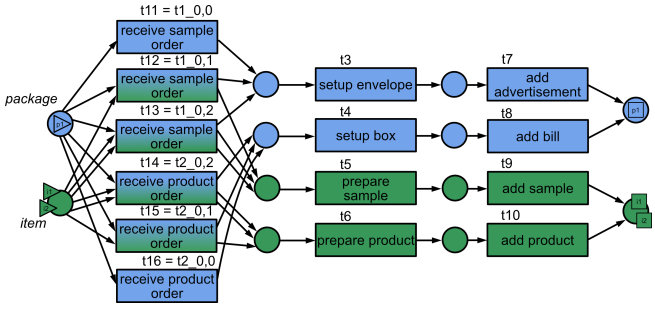

图 7. 预处理后的不含变量弧的接受型对象中心 Petri 网

5.3 创建同步积网

现在，我们已经有了两个不含变量弧的接受型对象中心 Petri 网：一个描述流程执行，另一个描述法定行为。我们可以将它们组合起来，创建同步积网。同步积网应当能够直接关联到可能的对齐。在一次对齐中，流程执行部分和法定行为部分要么分别向前推进，要么同步向前推进。若要使活动在流程执行部分和法定行为部分同时发生，它们必须是同一个活动，并且作用于相同的对象实例。对象中心 Petri 网只能区分对象

类型之间，而非对象实例之间。因此，我们使用 ν-网扩展对象中心佩特里网的要求，使其能够区分不同的对象实例。正如同步积网在图 8 中所示，我们将日志、模型和同步变迁分别解码为 $(t_{log}, \gg)$,$(\gg, t_{mod})$, 和 (t_{log}, t_{mod})。

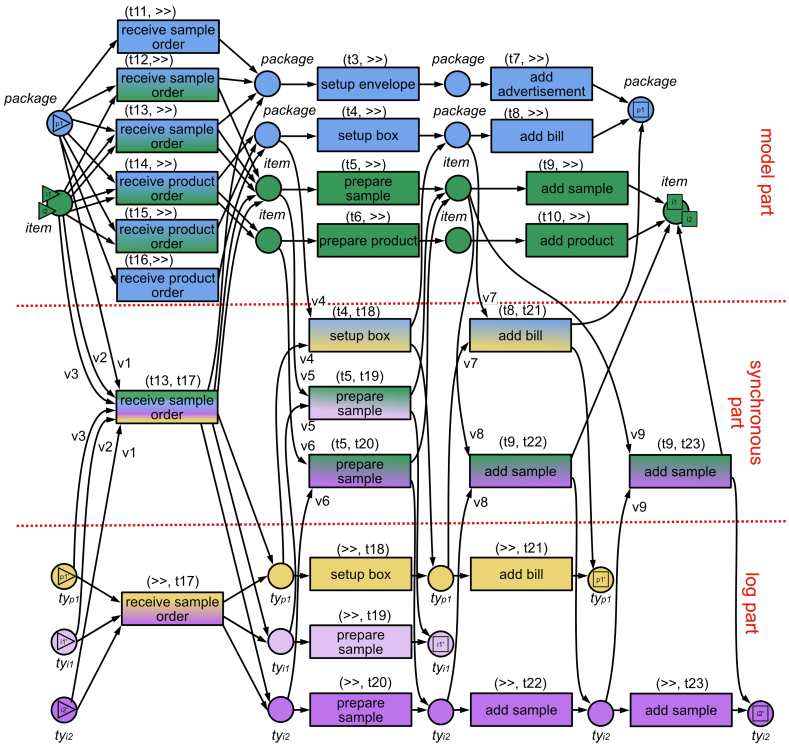

图 8. 图 6 中流程执行网与图 7 中接受型对象中心 Petri 网的同步积网

定义 18（同步积网）。
设 $PX_{net} = (((P_{PX}, T_{PX}, F_{PX}, l_{PX})\ pt_{PX}, F_{var_{PX}} = \emptyset)\ M_{init_{PX}}, M_{final_{PX}})$为一个流程执行网。设 $DJ_{net} = (((P_{DJ}, T_{DJ}, F_{DJ}, l_{DJ})\ pt_{DJ}, F_{var_{DJ}} = \emptyset)\ M_{init_{DJ}}, M_{final_{DJ}}$为一个法定网。设 Var 为一组未使用的变量名称。

如果两个变迁标记了相同的活动，并且消耗相同的令牌多重集，则它们可以同步。这要求它们关联到相同的类型，并且消耗和产生相同数量的令牌。

$T_{syn} = \{(t_{PX}, t_{DJ}) \in T_{PX} \times T_{DJ} | l_{PX}(t_{PX}) = l_{DJ}(t_{DJ}) \wedge \{orty(ty) | ty \in tpl(t_{PX})\} = tpl(t_{DJ}) \wedge \forall_{p_{PX} \in \bullet t_{PX}, p_{DJ} \in \bullet t_{DJ}} (orty(pt_{PX}(p_{PX})) = pt_{DJ}(p_{DJ})) \Rightarrow F_{PX}((p_{PX}, t_{PX})) = F_{DJ}((p_{DJ}, t_{DJ}))) \wedge \forall_{p_{PX} \in t_{PX}\bullet, p_{DJ} \in t_{DJ}\bullet} (orty(pt_{PX}(p_{PX})) = pt_{DJ}(p_{DJ})) \Rightarrow (F_{PX}((t_{PX}, p_{PX})) = F_{DJ}((t_{DJ}, p_{DJ})))\}$是同步变迁的集合。

同步积网是接受对象中心化 ν-佩特里网 $SP_{net} = (ON_{SP}, M_{init_{SP}}, M_{final_{SP}}, \nu)$，其中包含 $ON_{SP} = (N_{SP}, pt_{SP}, F_{var_{SP}} = \emptyset)$ 以及 $N_{SP} = (P_{SP}, T_{SP}, F_{SP}, l_{SP})$，满足：

– $P_{SP} = P_{PX} \cup P_{DJ}$是库所与 $pt_{SP} = pt_{PX} \cup pt_{DJ}$的并集– $T_{SP} \subset ($ $T_{PX}^{\gg} \times T_{DJ}^{\gg}) = \{(t_{PX}, \gg) | t_{PX} \in T_{PX}\} \cup \{(\gg, t_{DJ}) | t_{DJ} \in T_{DJ}\} \cup T_{syn}$是变迁与额外同步变迁的并集。– $F_{SP} = [(p_{PX}, (t_{PX}, \gg)) | (p_{PX}, t_{PX}) \in F_{PX}] \cup [(p_{DJ}, (\gg, t_{DJ})) | (p_{DJ}, t_{DJ}) \in F_{DJ}] \cup [((t_{PX}, \gg)\ p_{PX}) | (t_{PX}, p_{PX}) \in F_{PX}] \cup [((\gg, t_{DJ})\ p_{DJ}) | (t_{DJ}, p_{DJ}) \in F_{DJ}] \cup [(p_{PX}, (t_{PX}, t_{DJ})) | (p_{PX}, t_{PX}) \in F_{PX} \wedge (t_{PX}, t_{DJ}) \in T_{SP}] \cup [(p_{DJ}, (t_{PX}, t_{DJ})) | (p_{DJ}, t_{DJ}) \in F_{DJ} \wedge (t_{PX}, t_{DJ}) \in T_{SP}] \cup [((t_{PX}, t_{DJ})\ p_{PX}) | (t_{PX}, p_{PX}) \in F_{PX} \wedge (t_{PX}, t_{DJ}) \in T_{SP}] \cup [((t_{PX}, t_{DJ})\ p_{DJ}) | (t_{DJ}, p_{DJ}) \in F_{DJ} \wedge (t_{PX}, t_{DJ}) \in T_{SP}]$– $l_{SP} = \{((t_{PX}, \gg)\ a) | (t_{PX}, a) \in l_{PX}\} \cup \{(\gg, t_{DJ})\ a) | (t_{DJ}, a) \in l_{DJ}\} \cup \{((t_{PX}, t_{DJ})\ a) | (t_{PX}, a) \in l_{PX}, (t_{DJ}, a) \in l_{DJ}\}$– $M_{init_{SP}} = M_{init_{PX}} \cup M_{init_{DJ}}$和 $M_{final_{SP}} = M_{final_{PX}} \cup M_{final_{DJ}}$– ν：$P_{SP} \times T_{SP} \not\to Var$ 满足：$\forall_{(t_{PX}, t_{DJ}) \in T_{PX} \times T_{DJ}} \forall_{p_{PX} \in \bullet(t_{PX}, t_{DJ})} \exists_{(p_{DJ}, (t_{PX}, t_{DJ})) \in F_{SP}} (p_{DJ} \in P_{DJ} \wedge pt(p_{DJ}) = orty(pt(p_{PX}))) \Rightarrow (\nu((p_{DJ}, (t_PX, t_{DJ}))) = \nu((p_{PX}, (t_{PX}, t_{DJ}))))$

在流程执行部分中，我们使用 $orob$函数比较模型部分和日志部分中的对象。ν 网需求函数为同步变迁的入弧分配变量。对于来自流程执行网的每条入弧，都必须存在一条来自每个具有相同原始类型的库所的弧，以便这些弧具有由 Var分配的相同唯一变量。这要求同步移动在流程执行网和法定网中使用同一个对象实例。这些变量确保变迁对于具有相同变量的弧只能消耗同一个对象实例。这里指的是原始对象，而不是流程执行网中新创建的对象。

定义 19（同步积网中的有效绑定）。令 $SP_{net} = (((P_{SP}, T_{SP}, F_{SP}, l_{SP})\ pt_{SP}, F_{var_{SP}})\ M_{init_{SP}}, M_{final_{SP}}, \nu)$ 为同步积网。令 $SP_{net \backslash \nu}$ 为不含 ν的同步积网。绑定序列 $\sigma = \langle (t_1, b_2) \ldots, (t_n, b_n) \rangle \in B^*$ 对于 SP_{net} 是有效的，当且仅当它对于 $SP_{net \backslash \nu}$ 有效，并且对于每个与由 ν 分配了变量的弧相连的 $(t_i, b_i) \in \sigma$ （其中 $t_i \in T_{sny}$）而言，具有相同变量的弧消耗相同的对象：

$\forall_{p_{DJ} \in P_{DJ},\ p_{PX} \in P_{PX},\ \{p_{DJ}, p_{PX}\} \subseteq \bullet(t_i)} (\nu((p_{PX},\ t_i)) = \nu((p_{DJ},\ t_i)) \Rightarrow \{orob(o) | o \in b_i(pt((p_{PX},\ t_i)))\} \cap b(pt((p_{DJ},\ t_i))) \neq \emptyset)$。

贯穿示例对应的同步佩特里网如图 8 所示。

6 同步积网的对齐

本节介绍我们寻找对象中心对齐方法的第二部分。这部分的输入是一个同步积网，其中每个变迁都对应对齐中的一次移动。同步积网中从初始标识到最终标识的所有绑定序列都对应一个对齐。本节要做的是根据移动的成本函数，寻找一个最优对齐。同步积网中的每个绑定都直接对应一次移动。绑定定义了变迁和所用的对象。正如第 5 节所述，变迁已描述了移动的类型：$(\gg, t)$ 表示模型移动，$(t, \gg)$ 表示日志移动，(t_{PX}, t_{DJ}) 表示同步移动。绑定中的变迁也定义了活动。绑定中使用的对象就是该移动涉及的对象。

因此，寻找最优对齐与寻找从同步积网的初始标识到最终标识的最优绑定序列相关。将标识视为节点、将绑定视为标识之间的边，我们便可将搜索空间构建为一个图。边的成本则是将给定成本函数应用于与该绑定相关的移动所得的值。这是一个定义明确的搜索问题，我们可以使用标准搜索算法，在图中寻找成本最低或最短的路径。

定义 20（加权状态空间图）。
设 $SP_{net} = (((P_{SP}, T_{SP}, F_{SP}, l_{SP})\ pt_{SP}, F_{var_{SP}})\ M_{init_{SP}}, M_{final_{SP}}, \nu)$ 为一个同步积网。设 B_{SP} 为 SP_{net} 中的所有可能绑定。该佩特里网的加权搜索空间图为 $G_{wg} = (V_{wg}, E_{wg}, w_{wg})$，其中 $V_{wg} \subseteq \mathcal{B}(\mathcal{Q}_{ON_{SP}})$、$E_{wg} \subseteq \mathcal{B}(\mathcal{Q}_{ON_{SP}}) \times \mathcal{B}(\mathcal{Q}_{ON_{SP}})$ 以及 w_{wg}：$E_{wg} \to \mathbb{R}$ 满足：

- $V_{wg} = \{M \in \mathcal{B}(\mathcal{Q}_{ON_{SP}}) | \exists_{\sigma \in B^*_{SP}} M_{init_{SP}} \xrightarrow{\sigma} M\}$ 是同步积网中所有可达标识的集合。
- $E_{wg} = \{(M, M') \in \mathcal{B}(\mathcal{Q}_{ON_{SP}}) \times \mathcal{B}(\mathcal{Q}_{ON_{SP}}) | \exists_{(t,b)\ in B_{SP}} M (t,b) \longrightarrow M'\}$ 将每个标识连接到其可直接到达的标识。
- $w_{wg}(M, M') = cost_{move}(a_{log}, o_{log}, t_{model}, o_{model})$，其中 $(t, b) \in B_{SP}$ 与 $M (t,b) \longrightarrow M'$ 以及 $t = (t_{PX}, t_{DJ}) \in T_{SP}$ 满足：
  - $a_{log} = l_{SP}(t_{PX})$；$t_{model} = t_{DJ}$
  - $o_{log} = range(b)$ 如果 $a_{log} \neq \gg$，否则 $o_{log} = \emptyset$
  - $o_{model} = range(b)$ 如果 $a_{model} \neq \gg$ 否则 $o_{model} = \emptyset$

定理 1。设 P_X 是一个包含有限数量对象和事件的流程执行，并设 $DJ_{net} = (((P_{DJ}, T_{DJ}, F_{DJ}, l_{DJ})\ pt_{DJ}, F_{var_{DJ}} = \emptyset)\ M_{init_{DJ}}, M_{final_{DJ}})$ 是一个从初始标识出发具有有限数量可达标识的法定网。P_X 与 DJ_{net} 的同步积网 SP_{net} 具有有限数量的可达标识。

证明（有限搜索空间）。设 $P_X = (E_X, D_X)$ 是一个流程执行，其中包含有限数量的对象 $o = |X|$ 和事件 $e = |E_X|$。设 $DJ_{net} = (((P_{DJ}, T_{DJ},$

F_{DJ}， l_{DJ}） pt_{DJ}， $F_{var_{DJ}} = \emptyset$） $M_{init_{DJ}}$， $M_{final_{DJ}}$）是具有有限数量可达标识的流程执行规范模型 $m_{DJ} = |\{M|\exists_{\sigma \in B^*_{DJ}} M_{init_{DJ}\,(PX)} \xrightarrow{\sigma} M\}|$，其中 B_{DJ} 是 DJ_{net} 中可能绑定的集合。

我们方法的第 1 步根据流程执行构建流程执行网 PX_{net} =((((P_{PX}， T_{PX}， F_{PX}， l_{PX}) pt_{PX}， $F_{var_{PX}}$) $M_{init_{PX}}$， $M_{final_{PX}}$)。令B_{PX}为 PX_{net} 中所有绑定的集合。对于每个对象，每个事件至多添加一个库所，并为每个对象额外添加一个起始库所。由于事件和对象的数量有限， PX_{net}中的库所数量也是有限的。每个变迁消耗多少令牌，就会产生多少令牌，因为每个输入库所都对应一个输出库所。因此，令牌数量始终保持不变。由此， PX_{net} 中的可达标识 $m_{PX} = |\{M|\exists_{\sigma \in B^*_{PX}} M_{init_{PX}} \xrightarrow{\sigma} M\}| \leq (e+1)^o$ 数量有限。

我们方法的第 2 步对 DJ_{net} 进行预处理。此过程中不会添加库所。带有可变弧的变迁会被替换为一组变迁，这些变迁所能到达的标识也仅限于原变迁所能到达的标识。因此，可达标识的数量保持不变，故为有限。

在步骤 3 中， DJ_{net} 和 PX_{net} 合并在一起，形成同步积网。不会添加任何库所，而添加的同步变迁也不会产生新的可达标识，因为总是存在一个来自 DJ_{net} 的变迁和一个来自 PX_{net} 的变迁，它们共同对网中标识产生相同的影响。因此，对于 P_X 和预处理后的 DJ_{net} ，同步积网 SP_{net} 中的标识数量是两个网中标识数量的乘积: $m_{SP} = |\{M|\exists_{\sigma \in B^*_{SP}} M_{init_{SP}} \xrightarrow{\sigma} M\}| = m_{DJ} * m_{PX}$，其中 B_{SP}是 SP_{net}中所有可能绑定的集合。由于 m_{DJ} 和 m_{PX} 都是有限的，因此同步积网具有有限数量的可达标识。

如定理 1所述，且有限搜索空间证明所示，只要事件日志和规范模型的规模有限，加权搜索空间图就是有限的。对于有限加权图，可以判定是否存在代价最低路径；如果存在，Dijkstra 算法等算法[14] 就能找到它。唯一不存在代价最低路径的情况是完全不存在路径。这意味着同步积网无法从初始状态到达最终状态。只有当规范模型无法针对流程执行中的对象集合完成执行时，才会出现这种情况。换句话说，规范模型不包含与该流程执行相关的任何允许行为。在这种情况下，不存在对齐，并且系统会告知用户规范模型与该流程执行不匹配。这类似于规范模型无法完成执行时的传统对齐。在通常情况下，如果规范模型描述了与给定流程执行相关的行为，就可以借助最短路径确定最优对齐。

所得最短路径是从初始标识到最终标识的绑定序列。它直接对应于给定流程执行与规范模型的对象中心对齐。使用图 8 中的同步积网搜索最优对齐，可得到图 5 中的对齐。在此过程中遵循了包 与项目之间的对象间依赖。此外，对象

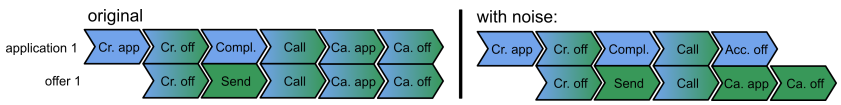

图 9. [17] 中的一次流程执行及其带有一些噪声的变体。

实例按照 ν-网的要求正确分离，因此不同对象实例的偏差不会相互抵消。从图 5 中可以看出，*i1* 缺少的添加样本 活动和$i2$ 多出的添加样本 活动均被识别为偏差。

可能存在多个成本相同的绑定序列。可以保证找到其中一个成本最低的绑定序列；但如果存在多个成本相同的绑定序列，具体找到哪一个取决于实现方式。

7 评估

我们使用 BPI2017 挑战赛[17]中的真实世界数据进行了评估。评估分为定性部分和运行时间的定量评估。

7.1 定性

定性评估旨在评估，与现有方法相比，所提出的方法是否确实能让我们更深入地了解复杂流程。接下来，我们将把所介绍的对象中心对齐方法与展平后的事件日志 [8] 上的传统对齐进行比较 [4]。

我们使用 BPI2017 [17] 数据进行真实场景评估。它是为数不多可公开获取、且能够转换为对象中心事件日志的事件日志之一。该流程包含两种对象类型：申请和报价。所有变体都恰好包含一个申请，且至少包含一个报价。申请可以被取消，也可以接受报价。

在此次评估中，我们仅选取了占流程执行总数 19.6% 的 4 个最主要变体。为简洁明了起见，我们移除了活动提交、完成和接受，因为它们只出现在申请与报价流程开始交互之前。我们使用 Python 库 ocpa [6] 从这 4 个变体中发现了法定佩特里网；该库实现了 van der Aalst 和 Berti [4]提出的发现方法。我们的方法也采用了 ocpa [6]实现。接下来，我们通过删除和替换给定流程执行中的事件，在日志中引入噪声。

原始的已接受流程执行和带噪声的流程执行见图 9。在此示例中，活动取消申请 从申请跟踪中删除，活动 取消报价也被删除。对于申请*1*，记录的活动不是取消报价 ，而是接受报价 。在报价*1*的跟踪中，没有任何变化。此类错误可能由此引入。

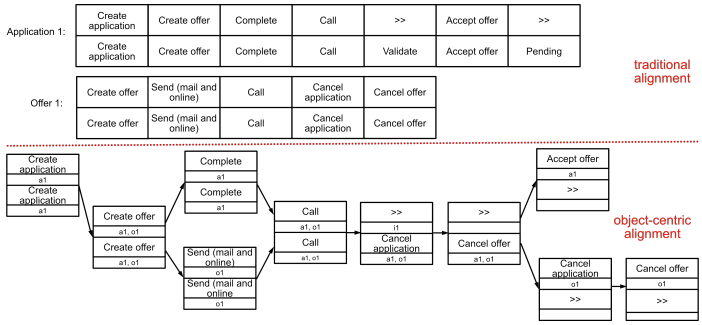

图10. 含噪声变体与评估规范模型的传统对齐和对象中心对齐。

由于人为错误而偏离了真实流程。这一变更造成了多个偏离，违背了规范模型所描述的期望行为。例如，无法在未接受报价的情况下接受申请。这是一种对象间依赖关系，我们希望对齐能够检测到。此外，事件 取消申请 和 取消报价 现在只记录在报价 上，却未与申请关联。因此，这也代表了不期望的行为。

我们在将对象中心流程执行和对象中心佩特里网 [4] 扁平化 [8] 后，应用了传统对齐方法。这产生了两种对齐。我们还将对象中心对齐方法应用于所描述的评估数据。传统对齐和对象中心对齐均可见于图10。

比较这两种方法，可以发现诊断结果存在很大差异。最大的差异在于，传统对齐未能检测出申请*1*本不应被接受。原因是，采用传统方法时，申请*1*的对齐是在不了解任何报价的情况下计算的。因此，它不得不假设活动接受报价是有效活动，然而实际上并没有可供接受的匹配报价。由于缺少这一信息，传统对齐会提示日志中缺少验证和待处理活动，进一步强化了对该申请本应获批的理解。我们的方法考虑了对象之间的所有依赖关系，因此将接受报价识别为日志移动。此外，传统对齐没有指出报价*1*存在任何控制流偏差。这导致报价*1*和申请*1*的对齐结果相互矛盾，因为申请不可能既被接受又被取消。对象中心对齐不会出现这类矛盾，因为它会一次性考虑整个流程执行。

7.2 定量分析

为评估我们方法的可扩展性，我们对运行时间进行了定量分析。

评估设置 我们使用 BPI2017 事件数据作为数据源 [17]。仅使用了出现频率最高的 50% 活动，所有其他活动均被过滤掉。随后，日志包含 755 个变体。我们使用了一个经过设计的佩特里网，使给定日志包含一些未对齐的流程执行。所使用的佩特里网可在 GitHub[1] 上获取。该网络包含 6 个可见变迁和 4 个静默变迁。网络中共有 10 个库所。我们将全部 755 个变体与模型进行了对齐，并记录了它们的属性以及所得的对齐计算时间。该评估的原始结果可在 GitHub[1] 上找到。评估在一台配备 8GB 内存、主频为 3.1 GHz 的双核英特尔酷睿 i5 计算机上进行。

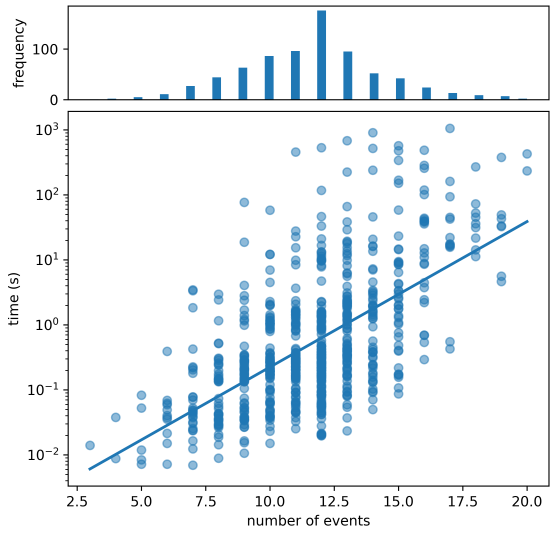

图11。 对数刻度下，执行时间随事件数量的变化。

结果 在计算出的755个对齐中，成本从0到最高5不等。事件数量从3个事件到20个事件不等，大多数流程执行包含11到14个事件。涉及的对象实例数量从2个增加到7个。这些属性的分布可见于图11、图12和图13。最短计算时间为0.007秒，最长为1051.8秒。可以

可以看到，对于所有属性的取值范围接近边界的情况，数据点都比较少，因此这些取值对应的结果更容易受到异常值的影响。我们计算了每对属性之间的相关系数。事件数量与对象数量呈正相关，相关系数为 0.59；而事件数量与成本呈负相关，相关系数为 -0.38。尤其是后者令人意外，因为人们通常会认为，事件更多的流程变体会有更多偏差，因此成本也会更高。

我们绘制了计算时间随三个维度变化的曲线：事件数量（图11）、对象数量 （图12）和成本（图13）。为减小负相关的影响，我们按照事件数量对图13中的数据点进行了分组。请注意，三个图中的时间尺度均为对数刻度，因此，线性增长代表指数增长。在这三个维度上，计算时间均呈指数增长。

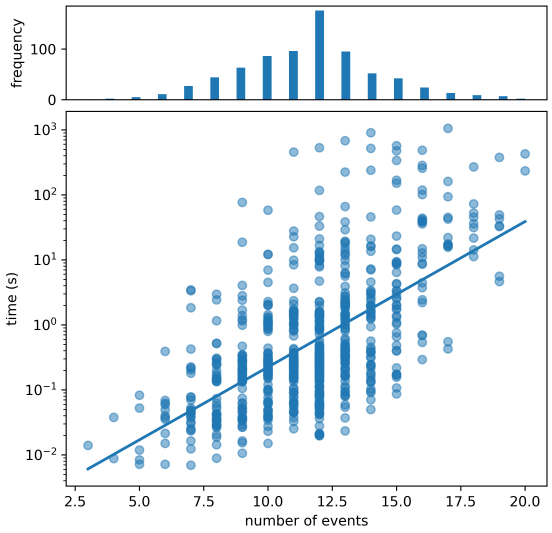

图12。 对象数量与执行时间的对数刻度关系。

这一结果可以通过搜索空间的结构来解释。更多事件会增大流程执行网的规模，也可能增加同步积网中的同步变迁数量。因此，可能的标识更多，搜索空间也随之扩大。更多对象同样会增大流程执行网的规模，并带来大量并行行为，从而扩大搜索空间。Dijkstra 算法的运行时间平均而言与搜索空间的大小呈指数关系 [16]。因此，随着事件和对象数量的增加，计算时间会呈指数增长。

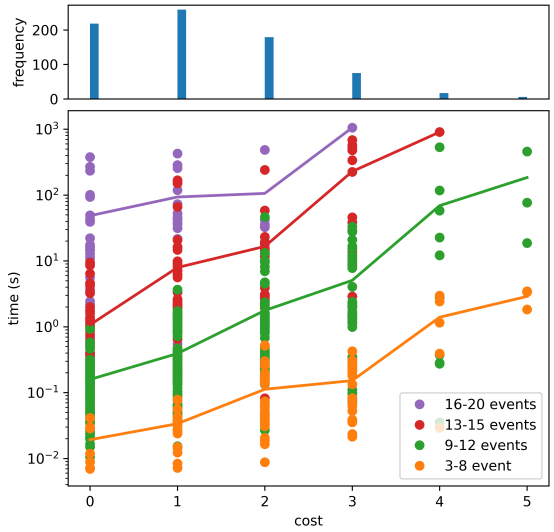

图13. 按事件数量分组的对齐成本对数刻度执行时间。

对于成本更高的对齐，探索的搜索空间范围更大，因为迪杰斯特拉会探索所有可通过一条成本低于到达最终标识的代价最低路径的路径抵达的顶点。因此，对齐的成本越高，运行时间就会呈指数增长。

8 结论

本文针对对象中心流程挖掘中的符合性检查提出了四项贡献。首先，我们定义了对象中心对齐，将传统对齐推广到移动图。其次，我们提供了一种算法，用于计算对象中心日志与佩特里网之间的最优对齐。该两步方法首先构建一个同步积网，然后搜索从初始标识到最终标识的最优绑定序列。第三，我们使用开源库 OCPA [6] 实现了该算法，并在 GitHub[1] 上公开发布。最后，我们对所提出的方法进行了评估。定性评估表明，对象中心对齐在偏差诊断方面具有优势。在扁平化事件数据时，复杂的对象间依赖会丢失，从而导致对齐中出现矛盾。使用对象中心对齐，我们能够保留这些依赖关系并避免矛盾。定量评估表明，计算时间会随着对象实例数量和对齐成本的增加而呈指数级增长。这表明，将整个对象中心事件日志与一个中等符合程度的模型进行对齐可能过于耗时。

耗时较长。在这些情况下，可以使用对象中心对齐，以获取针对单个流程执行或变体的具体诊断信息。

基于对象中心对齐的未来工作有两个方向。一方面，可以研究放宽当前方法的限制。例如，可以取消固定对象集这一假设，从而允许该方法引入可能缺失的全新对象。另一方面，可以致力于降低复杂度和运行时间：启发式方法、使用 A^* 算法，以及定义问题的松弛，都是降低计算时间的有前景方向。

参考文献

1. van der Aalst, W.M.：流程挖掘。《ACM通讯》**55**(8), 76–83 (2012 2240257 2 10 1145 2240236). https://doi.org/ . / . . van der Aalst, W.M.：以对象为中心的流程挖掘：处理事件数据中的分歧与汇合。载于：软件工程与形式化方法。第 3 1 1 3 25 2019 10 1007 978 3 030 30446 – 页。Springer ( ). https://doi.org/ . / - - - - . van der Aalst, W.M., Adriansyah, A., van Dongen, B.：因果网：面向流程发现的建模语言。载于：CONCUR 2011——并发理论，第 6901 卷，第 28–42 页。Springer (2011)。https://doi.org/10.1007/978-3-642- 23217-6 3 4. van der Aalst, W.M., Berti, A.：发现以对象为中心的佩特里网。《Fundamenta Informaticae》**175**1946 5 1 4 1 40 2020 10 3233 2020 (-), – ( ). https://doi.org/ . /FI- - . Adams, J.N., van der Aalst, W.M.：对象中心过程挖掘中的精确度与拟合度。载于：2021年第3届国际流程挖掘会议 (ICPM)。第 128– 135 9576886 6 2021 10 1109 53251 2021 页。IEEE ( ). https://doi.org/ . / ICPM . . . Adams, J.N., Park, G., van der Aalst, W.M.：ocpa：用于以对象为中心的流程分析的 Python 库。Software Impacts **14**, 100438 (2022)。https://doi.org/10 100438 7 1016 2022 . /j.simpa. . . Adams, J.N., Schuster, D., Schmitz, S., Schuh, G., van der Aalst, W.M.：为以对象为中心的事件数据定义案例与变体。载于：2022年第4届国际流程挖掘会议 (ICPM)。第 128–135 页。IEEE (2022)。https://doi.org/10 9980730 8 1109 57379 2022 . /ICPM . . . Adriansyah, A.：对齐观察到的行为与建模行为 (2014)。https://doi.org/10.6100/IR770080，出版方：埃因霍温理工大学 9. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.：基于对齐的精确度检查。载于：业务流程管理研讨会。第 137–149 页。Springer (2013)。https://doi.org/10.1007/978- 3-642-36285-9 15 10. Adriansyah, A., Sidorova, N., van Dongen, B.：一致性检查中的基于成本的拟合度。载于：2011年第十一届并发在系统设计中的应用国际会议。第 57 19 11 66 2011 10 1109 2011 – ( ). https://doi.org/ . /ACSD. . . Borrego, D., Barba, I.：数据感知场景中声明式业务流程模型的一致性检查与诊断。《应用专家系统》**41**(11), 5340 010 12 5352 2014 10 1016 2014 03 – ( ). https://doi.org/ . /j.eswa. . . . Burattin, A., Maggi, F.M., Sperduti, A.：基于多视角声明式流程模型的一致性检查。《应用专家系统》**65**(C), 194–211 (2016)。https://doi.org/10.1016/j.eswa.2016.08.040

13. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.：《一致性检查：关联流程与模型》。Springer (2018)。https://doi.org/10.1007/978-3- 319-99414-7 14. Dechter, R., Pearl, J.：广义最佳优先搜索策略及 A* 的最优性。ACM学报**32**3830 15 3 505 536 1985 10 1145 3828 ( ), – ( ). https://doi.org/ . / . . Desel, J., Reisig, W.：库所/变迁 Petri 网。载于：Petri 网讲义 I：基本模型，第 122 6 15 16 173 1998 10 1007 3 540 65306 – 页。Springer ( ). https://doi.org/ . / - - - . Dijkstra, E.W.：关于图的两个问题的注记。《数值数学》101386390 17 1 269 271 1959 10 1007 2017 ( ), – ( ). https://doi.org/ . /BF . van Dongen, B.：BPI 挑战赛 (2017)。https://doi.org/10.4121/UUID:5F3067DF-F10B-45DA-B98B-86AE4C7A310B 18. van Dongen, B.F.：使用扩展标识方程高效计算对齐。载于：业务流程管理——第16届国际会议，BPM 2018 7 12 19 197 214 2018 10 1007 978 3 319 98648 . 第 – 页。Springer ( )。https://doi.org/ . / - - - - . Dunzer, S., Stierle, M., Matzner, M., Baier, S.：一致性检查：前沿文献综述。载于：第11届面向主体的业务流程管理国际会议论文集。第 1–10 页。S-BPM ONE '19，计算机协会 (2019)。https://doi.org/10 3329014 20 1145 3329007 . / . . Fahland, D.：利用事件知识图谱对多个行为维度进行流程挖掘。载于：《流程挖掘手册》，第 274–319 页。Springer (2022)。https://doi.org/10 3 9 21 1007 978 3 031 08848 . / - - - - . Felli, P., Gianola, A., Montali, M., Rivkin, A., Winkler, S.：Cocomot：通过可满足性模理论对多视角流程进行一致性检查。载于：Polyvyanyy, A., Wynn, M.T., Looy, A.V., Reichert, M. (编)：业务流程管理——第19届国际会议，BPM 2021，意大利罗马，2021年9月6日至10日，论文集。《计算机科学讲义》，第 12875 卷，第 217–234 页。Springer (2021)。https://doi.org/10 0 15 22 1007 978 3 030 85469 . / - - - - . Jensen, K., Kristensen, L.M.：有色 Petri 网：用于并发系统形式化建模与验证的图形语言。《ACM通讯》**58**(6), 61–70 (2015 2663340 23 10 1145)。https://doi.org/ . / . de Leoni, M., van der Aalst, W.M., van Dongen, B.F.：业务流程的数据与资源感知型一致性检查。载于：业务信息系统——第15届国际会议，BIS 2012。第 48–59 页。Springer (2012)。https://doi.org/10 3 5 24 1007 978 3 642 30359 . / - - - - . Nesterov, R., Bernardinello, L., Lomazova, I., Pomello, L.：发现面向架构且可靠的多智能体系统流程模型：一种组合式方法。《软件与系统建模》(2022年5月)。https://doi.org/10.1007/s10270-022-01008-x 25. Rozinat, A., van der Aalst, W.M.：基于实际行为监控的流程一致性检查。《信息系统》**33**(1), 64–95 (2008)。https://doi.org/10 001 26 1016 2007 07 . /j.is. . . . Sommers, D., Sidorova, N., van Dongen, B.：将事件日志与资源受限的 ν-Petri 网对齐。载于：Petri 网与并发的应用与理论，第 325 5 17 27 345 2022 10 1007 978 3 031 06653 – 页。Springer ( )。https://doi.org/ . / - - - - . Song, W., Xia, X., Jacobsen, H.A., Zhang, P., Hu, H.：事件日志与流程模型之间的高效对齐。《IEEE服务计算汇刊》**10**(1), 136–149 (2017)。https://doi.org/10.1109/TSC.2016.2601094