Towards Standardized Modeling of Collaboration Processes in Collaboration Process Discovery

Janik-Vasily Benzin[1] (✉) and Stefanie Rinderle-Ma[1]

Technical University of Munich, TUM School of Computation, Information and Technology, Garching, Germany
{janik.benzin,stefanie.rinderle-ma}@tum.de

Abstract. Collaboration processes represent behavior of collaborating cases within multiple process orchestrations that interact via collaboration concepts such as organizations, agents, objects, and services. The heterogeneity of collaboration concepts and types such as message exchange and synchronous collaboration has led to different models targeted by collaboration process discovery (CPD) techniques, but a standard model class is lacking. In this paper, in order to reduce heterogeneity among model classes and to reveal similarities between CPD techniques, we prove that the synchronous collaboration type simulates message exchanges, but not vice versa. This constitutes a step towards a standard CPD model class that achieves comparability between CPD techniques, enables approach and property transfer, and is a condition for a standardized collaboration mining pipeline similar to process mining.

Keywords: Collaboration Process Discovery · Collaboration Process Models · Standardization of Nets · Bisimulation · Collaboration Mining

1 Introduction

Business processes define the control-flow of business activities, i.e., what work has to be done in what order [14]. Process discovery [2], so far, has mostly discovered process orchestrations represented by, e.g., Petri nets, from a set of process instances correlated by similar cases [7]. *Collaboration processes* define the control-flow for similar *collaborating cases* [4]. As a collaboration process consists of multiple process orchestrations that collaborate to achieve a common business goal, its collaborating cases consist of multiple cases each corresponding to one of the process orchestrations. *Collaboration process discovery* (CPD) techniques such as [3,6,17] aim at discovering a process model from a set of process instances grouped by collaborating cases.

Collaboration between cases can be classified into four *collaboration types* Υ: υ_m is the *message exchange*, υ_h is the *handover-of-work*, υ_r is the *resource sharing*, and υ_s is the *synchronous collaboration* [27,16,4]. Each collaboration occurs via various types of *collaboration concepts* such as a hospital's departments [17] or a company's agents [24]. Hence, a collaboration process consists of multiple collaboration concepts whose behavior is modeled as a process orchestration

that collaborate with each other. Although different proposals exist to represent the discovered collaboration processes, e.g., *composed RM_WF_nets* [16,17] and *BPMN collaboration diagrams* [21,5], a standard model class is missing [4]. Similar to the de-facto standard of *workflow nets* to model process orchestrations in process mining, a standard for collaboration processes achieves comparability between techniques, enables transfer of approaches and properties, and lays the foundation for a standardized and modular collaboration mining pipeline. To delineate a potential standard model class for CPD techniques, we consider existing model classes and aim to prove similarities between them. Hence, our research question is: **How can we reduce heterogeneity among existing model classes of collaboration process discovery by proving similarities towards standardization?**

In this work, we use Petri nets as notation, as either Petri nets or BPMN diagrams are targeted by CPD techniques and BPMN diagrams can be transformed into an equivalent Petri net, e.g., [5]. We repeat basic definitions and notations in Sect. 2. In Sect. 3, we introduce model classes for collaboration processes that are discovered by CPD techniques. In Sect. 4, we show that models with message exchanges are similar to models with synchronous collaboration and present the impact of our result. Lastly, we conclude and give an outlook in Sect. 5.

2 Labeled Petri Nets, Reachability, Language, WF-Net

In the following, we shortly repeat definitions and notation.

A *labeled Petri net* is a 5-tuple $N = (P, T, F, l, \Lambda_{\{\tau\}})$, where P is the set of *places*, T is the set of *transitions* with $P \cap T = \emptyset$, $F \subseteq ((P \times T) \cup (T \times P))$ is the *flow* relation, $l : T \to \Lambda_{\{\tau\}}$ is the *labeling* function, and Λ a finite set of *activity labels* with the *silent activity* τ. We define the *preset* of $x \in P \cup T$ by $\bullet x = \{y \mid (y, x) \in F\}$ and the *postset* of x by $x\bullet = \{y \mid (x, y) \in F\}$. A multiset of places $m \in \mathcal{B}(P)$ is called a *marking*. Given a marking m, $m(p)$ specifies the number of tokens in place p. The tuple (N, m) is called a *marked Petri net*. The *transition enabling* $(N, m)[t\rangle$ for $t \in T$ is defined by $(N, m)[t\rangle$ iff $m(p) \geq 1$ for all $p \in \bullet t$. An enabled transition $(N, m)[t\rangle$ can fire, which removes a token from each of it's input places, adds a token to each of it's output places, and executes an activity $\alpha \in \Lambda_{\{\tau\}}$ represented by $l(t)$. We define this behavior in the *firing rule*: $(N, m) \xrightarrow{l(t)} (N, m')$ iff $(N, m)[t\rangle$ and $m' + \bullet t = m + t\bullet$[1]. We omit the labelled Petri net N, if the context is clear. A trace $\sigma = \langle \alpha_1, \ldots, \alpha_n \rangle \in {\Lambda^*_{\{\tau\}}}$[2] is a *firing trace* of (N, m_0) iff $m_0 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} m'$, in short $m_0 \xrightarrow{\sigma} m'$. We define the set of *reachable markings* by $\mathcal{R}(N, m_0) = \left\{ m' \mid \exists_{\sigma \in \Lambda^*_{\{\tau\}}} m_0 \xrightarrow{\sigma} m' \right\}$ and the *language* given final marking $m_f \in \mathcal{B}(P)$ by $\mathcal{L}(N, m_0, m_f) = \left\{ \sigma \in \Lambda^*_{\{\tau\}} \mid m_0 \xrightarrow{\sigma} m_f \right\}$. N is a *workflow net* (WF-net) iff (i) there exists a single source place $i \in P$ such that its preset is empty: $\bullet i = \emptyset$; (ii) there exists a single sink place $o \in P$ such

[1] Addition, subtraction etc. are lifted to multisets in the common way.
[2] The universe of traces over alphabet $\Lambda_{\{\tau\}}$ is denoted as $\Lambda^*_{\{\tau\}}$.

that its postset is empty: $o\bullet = \emptyset$; and (iii) every node $x \in P \cup T$ is on a directed path from i to o.

3 Model Classes in Collaboration Process Discovery

This section gives an overview on existing model classes targeted by CPD techniques. In the past 16 years, research on collaboration mining has proposed 15 CPD techniques as depicted in Tab. 1. Each of the 15 CPD techniques target a different model class to represent the discovered collaboration process model. 12 techniques target Petri nets and 3 techniques target BPMN models. The four different collaboration types (Υ): message exchange (υ_m), handover-of-work (υ_h), resource sharing (υ_r), and synchronous collaboration (υ_s) are discovered by CPD techniques with varying degree. Most techniques discover message exchange and synchronous collaboration. Note that if a CPD technique discovers υ_m, it also discovers υ_h, because υ_h is a special case of υ_m. υ_h corresponds to the "message start" BPMN element, i.e., handover-of-work is a message exchange in which the sent message enables the "first" activity in the receivers process [16,17,5].

Table 1. Overview of existing CPD techniques and their model classes.

CPD	Year	Model	Col. Types	Comm. Model	Net
[11]	2008	WF-nets	υ_m	-	Labeled
[27]	2013	Integrated RM_WF_nets	$\upsilon_m, \upsilon_r, \upsilon_s$	P2P	Labeled
[20,22]	2013/15	Artifact-centric models	υ_s	-	NA
[23]	2019	Communication nets	υ_m, υ_h	P2P	Higher
[1]	2020	Object-centric Petri nets	υ_s	-	Higher
[26]	2020	Top-level process model	υ_m, υ_h	P2P	Higher
[13]	2021	BPMN Choreography	υ_m, υ_h	P2P	Higher
[15]	2022	Industry net	υ_m, υ_h	P2P	Labeled
[10]	2022	System net	$\upsilon_m, \upsilon_h, \upsilon_s$	P2P	Higher
[16,17]	2023	Composed RM_WF_nets	Υ	P2P	Labeled
[24]	2023	Multi-agent system net	υ_h	P2P	Labeled
[3]	2023	Typed Jackson nets	υ_s	-	Higher
[19]	2023	Generalized WF-nets	$\upsilon_m, \upsilon_h, \upsilon_s$	P2P	Labeled
[21]	2023	BPMN collab. diagram without signals	υ_m, υ_h	P2P	Labeled
[5,6]	2024	BPMN collab. diagram	υ_m, υ_h	P2P, Pub/Sub	Higher

Interestingly, almost all CPD techniques represent a message exchange by a place, a shared resource by a place, and a synchronous collaboration by a (fused via equal labels) transition. The only exception for message exchanges is [5] due to supporting a *Pub/Sub* communication model. In contrast to a *point-to-point* (P2P) communication in which a single message is only sent and received once, a Pub/Sub communication model allows that a sent message msg is received as often as there a receivers *subscribed* (Sub) to a *published* (Pub) message msg. In Tab. 1, any extension or change to the labeled Petri net definition in Sect. 2 is

considered a *higher* Petri net. For example, the Pub/Sub communication cannot be represented in a labeled Petri net [12] such that the Petri net that is equivalent to the BPMN collab. diagram of [5] is classified as a higher Petri net. In the next section, we show how the different model classes targeted by CPD techniques can be brought closer together.

4 Simulating Collaboration Types

In the following, we bring the various model classes closer together by showing that synchronous collaboration suffices to also model message exchanges, but not vice versa. To that end, we state prerequisites of our proof in Sect. 4.1. Next, we introduce a simulation relation on collaboration types to state and prove our claim in Sect. 4.2. Lastly, we discuss the impact of our results in Sect. 4.3.

4.1 Prerequisites

We focus on labeled Petri nets for two reasons. First, quality metrics that measure the quality of discovered models are mostly defined on this "basic" formalism [8] such that a standard model class within labeled Petri nets enables use of the majority of quality metrics. Second, labeled Petri nets allow us to balance the complexity of the proof in the next section and the extent to which our results impact CPD (there exists a 50/50 split between labeled and higher in Tab. 1). As a consequence, we restrict message exchanges to the P2P communication model.

Next, we introduce the *collaboration composition* (CC) as an abstract model class that concisely represents the process orchestrations of n collaboration concepts collaborating via message exchange, handover-of-work, and synchronous collaboration in a collaboration process.

Definition 1 (Collaboration Composition). *Let $V = \{N_c \mid c \in \{1, \ldots, n\}\}$ be a set of n disjoint*[3] *WF-nets and let P_{AC} be a set of message types with sending* $\text{send} : P_{AC} \to \mathcal{P}(T_{\not\tau}^n)$[4] *and receiving transitions* $\text{rec} : P_{AC} \to \mathcal{P}(T_{\not\tau}^n)$ *for $T_{\not\tau}^n = T^n \setminus \{t \in T^n \mid l(t) = \tau\}, T^n = \bigcup_{c \in \{1,\ldots,n\}} T_c$. The collaboration composition (CC) is a labeled Petri net $\mathcal{C}(V, P_{AC}, \text{send}, \text{rec}) = (P, T, F, l, \Lambda_{\{\tau\}})$ defined by:*

1. $P = \bigcup_{c \in \{1,\ldots,n\}} P_c$,
2. $l' : T^n \to \Lambda_{\{\tau\}}, l'(t) = l_c(t)$ *with* $c \in \{1, \ldots, n\}$ *and* $t \in T_c$,
3. $T = \bigcup_{c \in \{1,\ldots,n\}} r(T_c)$, *with r a renaming function: $r(x) = t_s$ if there exists $T_s \in ET$ such that $x \in T_s$ and $t_s \in T_s$ a fixed transition with $ET = \{T_s \subseteq T^n \mid \forall_{t,t' \in T_s} l'(t) = l'(t') \wedge l'(t) \neq \tau\}$ the set of equally-labeled (synchronous) transition subsets, otherwise $r(x) = x$,*

[3] Two WF-nets N_1, N_2 are *disjoint* iff their place names $P_1 \cap P_2 = \emptyset$ and transition names $T_1 \cap T_2 = \emptyset$ are disjoint.

[4] Given set X, $\mathcal{P}(X) = \{X' \mid X' \subseteq X \wedge X' \neq \emptyset\}$.

4. $F = \{(r(x), r(y)) \mid (x, y) \in \bigcup_{c \in \{1,\ldots,n\}} F_c\} \cup$
   $\{(r(t), p_{ac}), (p_{ac}, r(t')) \mid p_{ac} \in P_{AC} \wedge t \in \text{send}(p_{ac}) \wedge t' \in \text{rec}(p_{ac})\}$, *and*
5. $l(t) = l'(t)$ *for* $t \in T$.

A CC $\mathcal{C}(V, P_{AC}, \text{send}, \text{rec})$ *can only have initial markings* $m_0 \in \mathcal{B}(P)$ *that specify non-zero tokens* $m_0(p) \neq 0$ *for source places* $p = i_c, c \in \{1, \ldots, n\}$ *of its WF-nets* N_c.

Fig. 1 depicts a marked CC $(\mathcal{C}(V, P_{AC}, \text{send}, \text{rec}), m_0)$ with $V = \{N_1, N_2\}$, $P_{AC} = \{p_{ac,1}, p_{ac,2}\}$, $\text{send} = \{(p_{ac,1}, \{t_4\}), (p_{ac,2}, \{t_2\})\}$, and $\text{rec} = \{(p_{ac,1}, \{t_1\}), (p_{ac,2}, \{t_5\})\}$. Note that t_3 in Fig. 1 embodies the fused transition as defined in (3), i.e., $t_3, t_6 \in T^2$ with $l_1(t_3) = l_2(t_6) = \text{c}$.

Since resource sharing υ_r is modeled as marked *self-loop* places by CPD techniques [27,16,17], we deliberately left υ_r out of a CC. Self-loop places marked with a token can be removed without changing the behavior of labeled Petri nets [18], e.g., the language does not change. As υ_h is a special case of υ_m (cf. Sect. 3 and Fig. 1), we do not need to represent it in a CC on its own. Overall, we focus on collaboration types $\Upsilon_{ms} = \{\upsilon_m, \upsilon_s\}$. We say collaboration type υ_s is *contained* in $\mathcal{C}(V, P_{AC}, \text{send}, \text{rec})$ iff $ET \neq \emptyset$. Likewise, we say collaboration type υ_m is *contained* in $\mathcal{C}(V, P_{AC}, \text{send}, \text{rec})$ iff $P_{AC} \neq \emptyset$. For example, the CC in Fig. 1 contains both υ_m and υ_s.

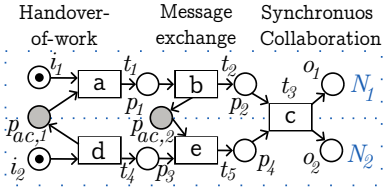

Fig. 1. Collaboration composition $\mathcal{C}(V, P_{AC}, \text{send}, \text{rec})$ with initial marking and two collaboration concepts collaborating via υ_h, υ_m, and υ_s.

In the next section, we apply the *weak bisimulation* equivalence [25] on collaboration types Υ_{ms} (cf. Def. 2) to show that message exchanges are "syntactic sugar" for synchronous collaborations (cf. Theorem 1), but not vice versa (cf. Theorem 2). Because the silent activity is typically "disregarded" for the purpose of analyzing a discovered Petri net, e.g., by conformance checking, the weak bisimulation equivalence that also "disregards" silent activities comes with a suitable granularity of differentiating labeled Petri nets. We define weak bisimulation on labeled Petri nets as follows.

A *labeled transition system* (LTS) is a 4-tuple $\Gamma = (S, \Lambda_{\{\tau\}}, s_0, \longrightarrow)$, where S is a set of states, $\Lambda_{\{\tau\}}$ is the set of labels, $s_0 \in S$ is the *initial state*, and $\longrightarrow \subseteq S \times \Lambda_{\{\tau\}} \times S$ the set of labeled edges. Note that we overload the notation of $\longrightarrow$ similar to the firing rule in Sect. 2. We write $s_1 \xrightarrow{\sigma} s_{n+1}$ iff there exists $\sigma = \langle \alpha_1, \ldots, \alpha_n \rangle \in \Lambda^*_{\{\tau\}}$ and $s_1, \ldots, s_{n+1} \in S$ such that $s_1 \xrightarrow{\alpha_1} s_2 \ldots s_n \xrightarrow{\alpha_n} s_{n+1}$. We define the *weak transition relation* $\stackrel{\alpha}{\Longrightarrow} \subseteq S \times \Lambda_{\{\tau\}} \times S$ with $\alpha \in \Lambda_{\{\tau\}}$ by (i) $s \, (\xrightarrow{\tau})^* \, s_1 \xrightarrow{\alpha} s_2 \, (\xrightarrow{\tau})^* \, s'$, if $\alpha \neq \tau$, and (ii) $s \, (\xrightarrow{\tau})^* \, s'$, if $\alpha = \tau$, where $(\xrightarrow{\tau})^*$ is the reflexive, transitive closure of $\xrightarrow{\tau}$. We lift the notation of the weak transition relation to traces in the same manner as for $\longrightarrow$. We define the *weak bisimulation equivalence* on the set of all LTS with labels $\Lambda_{\{\tau\}}$. Let $\Gamma_1 = (S_1, \Lambda_{\{\tau\}}, s_{0,1}, \longrightarrow_1)$ and $\Gamma_2 = (S_2, \Lambda_{\{\tau\}}, s_{0,2}, \longrightarrow_2)$ be two LTSs. A relation $R \subseteq S_1 \times S_2$ is a *weak simulation*, denoted by $\Gamma_1 \preceq_R \Gamma_2$, iff

i $(s_{0,1}, s_{0,2}) \in R$, i.e., the initial states are related; and
ii for every $(p, q) \in R$ and $\alpha \in \Lambda_{\{\tau\}}$ it holds that: if $p \xrightarrow{\alpha}_1 p'$, then $\alpha = \tau$ and $(p', q) \in R$, or there exists $q' \in S_2$ such that $q \overset{\alpha}{\Longrightarrow}_2 q'$ and $(p', q') \in R$.

If R is symmetric, it is a *weak bisimulation equivalence*, written $\Gamma_1 \approx_R \Gamma_2$. We also say Γ_1 is weakly bisimilar to Γ_2. Two marked Petri nets $(N, m_0), (N', m_0')$ are weakly bisimilar, written $(N, m_0) \approx (N', m_0')$, iff the two LTSs $\Gamma_{N,m_0} = \left(\mathcal{B}(P), \Lambda_{\{\tau\}}, m_0, \longrightarrow\right), \Gamma_{N',m_0'} = \left(\mathcal{B}(P'), \Lambda_{\{\tau\}}, m_0', \longrightarrow'\right)$ are weakly bisimilar[5].

4.2 Synchronous Collaboration simulates Asynchronous Collaboration

By means of the weak bisimulation equivalence, we state the two theorems of this paper as follows. First, any CC that only contains υ_m is weakly bisimilar to another CC that only contains υ_s (cf. Theorem 1). Second, any CC that only contains υ_s does not have a CC that only contains υ_m such that both are weakly bisimilar (cf. Theorem 2). To formalize both statements, we introduce our notion of *simulating* collaboration types.

Definition 2 (Simulating Collaboration Types). *A type $\upsilon_1 \in \Upsilon_{ms}$ simulates type $\upsilon_2 \in \Upsilon_{ms}$ with $\upsilon_1 \neq \upsilon_2$ iff for any marked collaboration composition $(\mathcal{C}(V, P_{AC}, \text{send}, \text{rec}), m_0) = (N, m_0)$ that only contains type υ_2, there exists a marked collaboration composition $(\mathcal{C}(V', P'_{AC}, \text{send}', \text{rec}'), m_0') = (N', m_0')$ that only contains type υ_1 such that $V' = V \cup \{N_c \mid c \in \{n+1, \ldots\}\}$ and $(N, m_0) \approx (N', m_0')$.*

In the construction of a (N', m_0') that simulates (N, m_0), it is important to prohibit changes to the WF-nets V except than extending to V', as other changes to V would mean that we allow simulating collaboration types by a collaboration concept's internal WF-net and not by some other collaboration type. In the following, we show that synchronous collaboration υ_s simulates message exchanges υ_m , under three conditions: neither *τ-labeled skipping* nor *τ-labeled loop* for sending messages of type $p_{ac} \in P_{AC}$ exist and there does not exist a *token generator* [9] that can infinitely often send a message of type p_{ac}. A *τ-labeled skipping* for sending p_{ac} exists iff $\sigma \in \Lambda^*_{\{\tau\}}$, $m_1 \in \mathcal{R}(N, m_0)$ exist such that $m_1 \xrightarrow{\sigma} m_2$, $m_1 \overset{\tau}{\Longrightarrow} m_2$, and sending activity $l(t)$ occurs in σ, i.e., $l(t) \in \sigma$[6] for $t \in \text{send}(p_{ac})$. A *$\tau$-labeled loop* in (N, m_0) for sending messages of type p_{ac} exists iff $m_1 \xrightarrow{l(t)} m_2$, $m_1 \in \mathcal{R}(N, m_0)$ and $m_2 \overset{\tau}{\Longrightarrow} m_3$ such that $(N, m_3)[t'\rangle$ for some $t, t' \in \text{send}(p_{ac})$. A token generator results in unbounded states of the coverability graph (ω-states) [18] and is characterized by: There exists $m \in \mathcal{R}(N, m_0), \sigma \in \Lambda^*_{\{\tau\}}, m' \in \mathcal{B}(P)$ such that $|m'| > 0$ and $m \xrightarrow{\sigma} m + m'$.

Theorem 1 (Synchronous execution simulates message exchange). *υ_s simulates υ_m, if for all $p_{ac} \in P_{AC}$ in $\mathcal{C}(V, P_{AC}, \text{send}, \text{rec})$: no τ-labeled skipping, no τ-labeled loop, and no token generator exist.*

[5] $\longrightarrow$ is the firing rule for (N, m_0) and $\longrightarrow'$ the firing rule for (N', m_0') (cf. Sect. 2).
[6] For $\sigma = \langle \alpha_1, \ldots, \alpha_n \rangle \in \Lambda^*_{\{\tau\}}$ and $\alpha \in \Lambda_{\{\tau\}}$, we write $\alpha \in \sigma$ iff $\exists_{i \in \{1,\ldots,n\}}\ \alpha_i = \alpha$.

Proof. Let $(\mathcal{C}(V, P_{AC}, \text{send}, \text{rec}), m_0) = (N, m_0)$ be a marked CC that only contains message exchanges υ_m. We prove the statement by constructing a new WF-net $N_{p_{ac}}$ for each message type $p_{ac} \in P_{AC}$ such that the new marked CC $(\mathcal{C}(V', \emptyset, \emptyset, \emptyset), m_0') = (N', m_0')$ with $V' = V \cup \{N_{p_{ac}} \mid p_{ac} \in P_{AC}\}$ is weakly bisimilar to the CC (N, m_0). Our construction is structured into four steps: Distinguishing three cases of how message exchanges occur (**step 1**), determining respective transition sets for two of the cases through analyzing the coverability graph (**step 2**), constructing a new WF-net $N_{p_{ac}}$ given the transition sets (**step 3**), and defining a weak bisimulation $Q \subseteq \mathcal{R}(N, m_0) \times \mathcal{R}(N', m_0')$ to complete the proof (**step 4**).

Step 1. Given $p_{ac} \in P_{AC}$, either (1) message type p_{ac} is *dead*, (2) p_{ac} is *optional*, or (3) p_{ac} is *compulsory*. First, message type p_{ac} is *dead* iff for all $t \in \text{send}(p_{ac})$ there is no reachable marking $m \in \mathcal{R}(N, m_0)$ such that $(N, m)[t\rangle$, i.e., the message exchange via type p_{ac} can never occur. Second, p_{ac} is *optional* iff there exist at least two firing traces $\sigma_\neg, \sigma \in \mathcal{L}(N, m_0, m_f)$ such that $\forall_{\alpha \in l(\text{send}(p_{ac}))}\ \alpha \notin \sigma_\neg$ (i.e., no sending activity occurs in $\sigma_\neg$) and $\exists_{\alpha \in l(\text{send}(p_{ac}))}\ \alpha \in \sigma$ (i.e., a sending activity occurs in σ). The final marking m_f specifies that as many tokens are on a WF-nets N_c's sink place o_c, as the initial marking specifies for a N_c's source place i_c, i.e., for $c \in \{1, \ldots, n\}$: if $m_0(i_c) = x$, then $m_f(o_c) = x$. The final marking specifies $m_f(p) = 0$, if $p \in P$ is not a sink place, i.e., $p\bullet \neq \emptyset$. Third, p_{ac} is *compulsory* iff for every firing trace $\sigma \in \mathcal{L}(N, m_0, m_f)$ it holds that $\exists_{\alpha \in l(\text{send}(p_{ac}))}\ \alpha \in \sigma$ (i.e., at least one sending activity always occurs).

Step 2. In Fig. 2, the WF-net $N_{p_{ac}}$ that is constructed in this and the next step is depicted. For (2), we ensure that the WF-net $N_{p_{ac}}$ that "simulates" optional message type p_{ac} with synchronous collaboration is able to skip all transitions with sending activities, as otherwise the corresponding markings would be missing in $\mathcal{R}(N', m_0')$. Skipping has to occur synchronously with transitions $T_{p_{ac},\times}$ "after" which sending cannot occur anymore (cf. Fig. 2). Also, sending activities may be repeated in the firing trace σ (cf. (2) and (3)) by traversing a loop. Hence, our construction must ensure that the transitions $T_{p_{ac},\circlearrowleft}$ "after" which sending occurs again and $T_{p_{ac},\emptyset,\times}$ "after" which repeated sending stops are also copied to $N_{p_{ac}}$ (cf. Fig. 2). In the following, we determine the necessary transition sets by sets of markings in the coverability graph.

Determine by breadth-first search for each $p_{ac} \in P_{AC}$ a unique set of markings $M_\times \subseteq \mathcal{R}(N, m_0)$ and set of markings $M_\circlearrowleft \subseteq \mathcal{R}(N, m_0)$ in the coverability graph. $M_\times$ is characterized by the following formula that identifies the markings from which activities "decide" whether a message is sent or never sent. For each $m_\times \in M_\times$ it holds that $m_\times$ is reachable by paths in the coverability graph in which no sending activities $\alpha \in l(\text{send}(p_{ac}))$ have occurred and formula $\gamma(m_\times)$ holds. $\gamma(m)$ holds iff there exist two sets of activities $A_0, A_1 \subseteq \Lambda$ with $m \overset{\alpha_0}{\Longrightarrow} m_\neg$, $\alpha_0 \in A_0$ such that for all $m' \in \mathcal{R}(N, m_\neg)$ sending activities cannot occur $\neg(N, m')[t\rangle$ for all $t \in \text{send}(p_{ac})$ and $m \overset{\alpha_1}{\Longrightarrow} m_1$, $\alpha_1 \in A_1$ such that there exists $m' \in \mathcal{R}(N, m_1)$ that enables sending activities $(N, m')[t\rangle$ for some $t \in \text{send}(p_{ac})$. Note that A_0, A_1 exist iff (2) holds or (3) with repeated sending, as sending and receiving transitions cannot be labeled with the silent activity (cf. Def. 1) and no

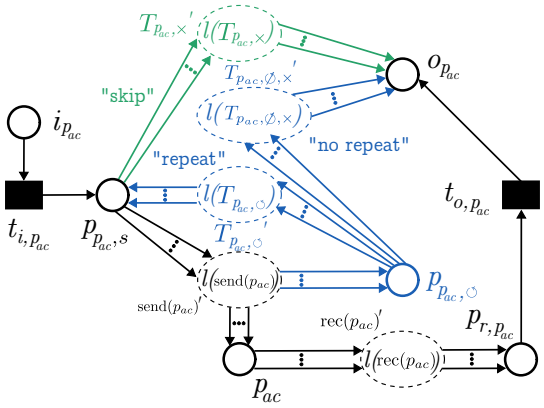

Fig. 2. WF-net $N_{p_{ac}}$ constructed for message type $p_{ac} \in P_{AC}$ and its sending $\text{send}(p_{ac})$ and receiving activities $\text{rec}(p_{ac})$. Conditional parts of the WF-net that are only constructed in certain cases are highlighted in blue and green.

τ-labeled skipping exists. Let $A_\times = \bigcup_{m_\times \in M_\times} A_{0,m_\times}$ be the union of activities $A_{0,m_\times}$ that are computed to satisfy $\gamma(m_\times)$. Define $T_{p_{ac},\times} = \{t \in T^n \mid l(t) \in A_\times\}$. Note that l is a bijection for $l(t) \in \Lambda$, i.e., transitions are uniquely determined by their observable label. If a transition in $T_{p_{ac},\times}$ fires, no sending activity for messages of type p_{ac} can occur.

To "simulate" loops, determine $M_\circlearrowleft$ whose markings are characterized by the following formula that identifies the markings from which activities "decide" whether a message will be sent again or not. For each $m_\circlearrowleft \in M_\circlearrowleft$ it holds that $m_\circlearrowleft$ is reachable by paths in the coverability graph in which some sending activity $\alpha \in l(\text{send}(p_{ac}))$ has occurred and $\gamma(m_\circlearrowleft)$ holds. If a sending activity is repeated, the set $M_\circlearrowleft$ cannot be empty, since the CC does not contain any *τ-labelled loops* for all $p_{ac} \in P_{AC}$. Let $A_\circlearrowleft = \bigcup_{m_\circlearrowleft \in M_\circlearrowleft} A_{1,m_\circlearrowleft}$ be the union of activities $A_{1,m_\circlearrowleft}$ that are computed to satisfy $\gamma(m_\circlearrowleft)$. These activities indicate that sending activities are executed again. Let $T_{p_{ac},\circlearrowleft} = \{t \in T \mid l(t) \in A_\circlearrowleft\}$ be the set of transitions for asynchronous collaboration p_{ac} that, if executed, result in firing traces with repeated sending activities. Observe that $T_{p_{ac},\circlearrowleft} = \emptyset$, if there are no loops involving sending activities. We refer to $T_{p_{ac},\circlearrowleft} = \emptyset$ with $\textcircled{\emptyset}$ and to the opposite with $\textcircled{\circlearrowleft}$. Since $T_{p_{ac},\times}$ only covers transitions that indicate no message of type p_{ac} is sent at all for $\textcircled{2}$, it misses transitions that indicate no message of type p_{ac} is sent after a message has been sent already, i.e., a loop of repeated sending activities is not "traversed" again. Let $A_{0,m_\circlearrowleft}$ for $m_\circlearrowleft \in M_\circlearrowleft$ be the sets of activities that were computed to satisfy $\gamma(m_\circlearrowleft)$, i.e., an activity $\alpha \in A_{0,m_\circlearrowleft}$ indicates that after some sending activity has occurred, it will not occur again. Next, $A_{\emptyset,\times} = \bigcup_{m_\circlearrowleft \in M_\circlearrowleft} A_{0,m_\circlearrowleft}$, and similarly: $T_{p_{ac},\emptyset,\times} = \{t \in T \mid l(t) \in A_{\emptyset,\times}\}$. Observe that $T_{p_{ac},\emptyset,\times} \neq \emptyset$ if $\textcircled{\circlearrowleft}$, because no token generator exists, no place $p \in \{p \mid p \in {\bullet t} \wedge t \in T_{p_{ac},\emptyset,\times}\}$ is a sink place, and N is composed of WF-nets.

Step 3. If either (2) or (3), construct WF-net $N_{p_{ac}} = (P_{p_{ac}}, T_{p_{ac}}, F_{p_{ac}}, l_{p_{ac}}, \Lambda_{\{\tau\}})$ as depicted in Fig. 2. Note that if we say $N_{p_{ac}}$ contains a part of Fig. 2, the renamed transitions depicted by a dotted oval with their labeling depicted inside the oval are added to $T_{p_{ac}}$ and $l_{p_{ac}}$ along with the places in their depicted pre- and postset to $P_{p_{ac}}$ and depicted arcs to $F_{p_{ac}}$. For example, the green part in Fig. 2 consists of renamed transitions $T'_{p_{ac},\times}$ with respective labels $l(T'_{p_{ac},\times})$, i.e., $T_{p_{ac}} = T_{p_{ac}} \cup T'_{p_{ac},\times}$ and $l_{p_{ac}}(t') = l(t)$ for $t' \in T'_{p_{ac},\times}$. Also, the green part adds the depicted arcs to $F_{p_{ac}}$. We distinguish four cases from the combination of (2) and (3) with (∅) and (↻). All four constructed WF-nets include the black parts in Fig. 2 that represent a WF-net with sending and receiving transitions of message type p_{ac}.
Case (2), (∅): In addition to the black part in Fig. 2, $N_{p_{ac}}$ contains the green part to skip sending messages.
Case (2), (↻): In addition to the black part in Fig. 2, $N_{p_{ac}}$ contains the green parts to skip sending messages and the blue part to repeat and stop repeating sending messages. Hence, $N_{p_{ac}}$ is equal to all depicted parts in Fig. 2.
Case (3), (∅): $N_{p_{ac}}$ contains the black part in Fig. 2 only.
Case (3), (↻): In addition to the black part in Fig. 2, $N_{p_{ac}}$ contains the blue part to repeat and stop repeating sending messages.
Step 4. Define $V' = V \cup \{N_{p_{ac}} \mid p_{ac} \in P_{AC}\}$ with $N_{p_{ac}}$ as constructed in **step 3**. Then, $(\mathcal{C}(V', \emptyset, \emptyset, \emptyset), m'_0) = (N', m'_0)$. Define m'_0 similar to m_0 for source places i_c of WF-nets $N_c \in V$, additionally $m'_0(p) = 1$ if $p = i_{p_{ac}}$, and $m'_0(p) = 0$ otherwise. Observe that the set of equally-labeled transitions ET' includes for each p_{ac} the following transitions:
Case (3), (∅): Because $N_{p_{ac}}$ only contains the black part in Fig. 2 in this case, ET' includes only renamed, equally-labeled sending and receiving transitions. Formally, for each $t \in ET'$ such that $t \in \text{send}(p_{ac}), t \in T^n$ another $t' \in ET'$ exists such that $t' \in \text{send}(p_{ac})', t' \in T_{p_{ac}}$ with $l(t) = l_{p_{ac}}(t')$. Similarly, ET' includes for each $t \in \text{rec}(p_{ac}), t \in T^n$ another $t' \in \text{rec}(p_{ac})', t' \in T_{p_{ac}}$ with $l(t) = l_{p_{ac}}(t')$. The remaining three cases are analogous.

Define relation $Q \subseteq \mathcal{R}(N, m_0) \times \mathcal{R}(N', m'_0)$ such that $(m, m') \in Q$ iff $m(p') \leq m'(p')$ for all places $p' \in P'$ of N'[7]. Then, $(N, m_0) \approx_Q (N', m'_0)$, since with the exception of new τ-labelled transitions in $N_{p_{ac}}$ for a message type p_{ac}, all transitions of the constructed WF-nets $N_{p_{ac}}$ are fused with their original counterparts in N, exactly the same flow relation is encoded in $\bullet p_{ac}$ and $p_{ac}\bullet$ as is defined by (4) in Def. 1, and optional and repeating behavior is exactly encoded. ■

To illustrate the construction in the last theorem's proof, Fig. 3 depicts the marked CC $(\mathcal{C}(V \cup \{N_{p_{ac,1}}, N_{p_{ac,2}}\}, \emptyset, \emptyset, \emptyset), m'_0)$ that only contains synchronous collaboration and that is weakly bisimilar to the CC in Fig. 1. Except for fused transitions, $N_{p_{ac,1}}$ and $N_{p_{ac,2}}$ are highlighted in grey in Fig. 3.

Although υ_s simulates υ_m under three conditions, all three conditions are not realistic in a real-world collaboration process, as τ-labeled skipping would mean that a collaboration concept can skip a collaboration without notifying the other

[7] If $p' \notin P$, $m(p) = 0$.

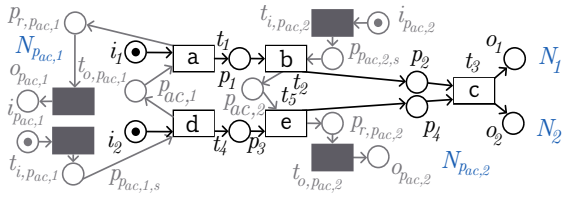

Fig. 3. Marked collaboration composition $(\mathcal{C}(V \cup \{N_{p_{ac,1}}, N_{p_{ac,2}}\}, \emptyset, \emptyset, \emptyset), m_0')$ constructed to simulate υ_m in $\mathcal{C}(V, P_{AC}, \text{send}, \text{rec})$ (Fig. 1) by υ_s as shown in Theorem 1.

collaborating concepts, the τ-labelled loop condition excludes arbitrary, non-observable sending of messages, and the existence of a token generator excludes arbitrarily creating collaboration concept instances.

The next theorem proves that synchronous collaboration υ_s cannot be simulated by message exchange υ_m.

Theorem 2 (Synchronous collaboration cannot be simulated). *υ_m does not simulate υ_s.*

Proof. We prove by contradiction. Assume υ_m simulates υ_s. Let $(\mathcal{C}(V, \emptyset, \emptyset, \emptyset), m_0) = (N, m_0)$ be a marked CC that contains only υ_s and $(\mathcal{C}(V', P_{AC}, \text{send}, \text{rec}), m_o') = (N', m_0')$ be a marked CC that contains only υ_m such that $(N, m_0) \approx (N', m_0')$. Observe that collaboration type υ_m is represented by P_{AC}, send, and rec. From Def. 1 (3) and (4), it follows that no transitions with different labels are merged or changed for exchanging messages of type $p_{ac} \in P_{AC}$. Hence, from Def. 2 it follows that all transitions $t \in T_s, T_s \in ET$ are still merged in (N', m_0') (changing V is prohibited) such that $ET \neq \emptyset$ and υ_s is contained in (N', m_0')[8]. ■

Overall, our theory on collaboration types in labeled Petri nets demonstrates that modulo weak bisimilarity, asynchronous collaboration can be simulated by synchronous collaboration, but not vice versa. In the next section, we elaborate on the implications.

4.3 Impact on Collaboration Process Discovery

The impact of the two theoretical results on simulating collaboration types on collaboration process discovery is twofold.
1. Since all CPD techniques in Tab. 1 that discover υ_s share the design choice to represent υ_s by fusing transitions similar to a CC (cf. Def. 1 (3)), the following statement holds at least for CPD techniques [15,24,21] that target labeled Petri

[8] Although the contradiction follows from a technicality, the statement still holds, if we enable synchronizing arbitrary transitions in a CC, which is rarely done. Nevertheless, asynchronous collaboration does not merge transitions and requires sending and receiving transitions that are not labeled with the silent activity, so we can never construct a single label out of at least a sending and receiving label.

net-based classes. If a CPD technique targets a model class that does not represent synchronous collaboration υ_s, the CPD technique can only be advanced to discover υ_s by both a structural change to the targeted model class and an algorithmic change to the technique. The statement is conjectured to also hold for CPD techniques [23,26,13,6,5] that target higher Petri net-based classes (cf. next section). Moreover, CPD techniques that can only discover υ_s can be tweaked to also discover υ_m by mirroring the construction of Theorem 1 in the event log, i.e., these techniques do not need to be changed.
2. Considering labeled Petri nets for a start, the standard model class for CPD must represent synchronous collaboration, but does not have to represent message exchanges, as it is "syntactic sugar". Also, the two theorems indicate that the model classes of [27,10,16,17] (incl. υ_s) are indistinguishable modulo weak bisimilarity. Additionally, models in these classes (incl. υ_s) can weakly simulate models from model classes [15,24,21] (excl. υ_s), but not vice versa.

All in all, the apparent heterogeneity in model classes among CPD techniques is misleading, as their models have more in common than is obvious.

5 Conclusion and Outlook

In this paper, we prove two statements that show to what extent heterogeneity in modeling collaboration by CPD techniques can be reduced to achieve a more standardized model class in collaboration mining. The first statement means that message exchanges are non-essential for discovering collaboration processes, while the second statement means that synchronous collaborations are essential. Hence, both a standard model class and a standard collaboration mining pipeline must be built with synchronous collaboration at their core.

In the future, we will analyze how we can transfer the two statements to also hold for higher Petri nets with the aim of delineating a standard model class for all CPD techniques. Furthermore, we will apply the theoretical results in designing a standard benchmark for CPD techniques. Moreover, we will extend the scope of standardizing collaboration mining towards quality metrics as proposed in conformance checking.

References

1. van der Aalst, W.M.P., Berti, A.: Discovering object-centric Petri nets. Fundamenta informaticae **175**(1-4), 1–40 (2020)
2. Augusto, A., Conforti, R., Dumas, M., Rosa et al., M.L.: Automated Discovery of Process Models from Event Logs: Review and Benchmark. IEEE Trans Knowl Data Eng **31**(4), 686–705 (2019)
3. Barenholz, D., Montali, M., Polyvyanyy, A., Reijers et al., H.A.: There and Back Again. In: PETRI NETS 2023. pp. 37–58 (2023)
4. Benzin, J.V., Rinderle-Ma, S.: Petri Net Classes for Collaboration Mining: Assessment and Design Guidelines. In: Process Mining Workshops (2024)
5. Corradini, F., Pettinari, S., Re, B., Rossi et al., L.: A technique for discovering BPMN collaboration diagrams. SoSyM (2024)

6. Corradini, F., Re, B., Rossi, L., Tiezzi, F.: A Technique for Collaboration Discovery. In: Enterprise, Business-Process and Inf, Syst. Modeling. pp. 63–78 (2022)
7. Diba, K., Batoulis, K., Weidlich, M., Weske, M.: Extraction, correlation, and abstraction of event data for process mining. Data Min. Knowl. Discov. **10**(3) (2020)
8. Dunzer, S., Stierle, M., Matzner, M., Baier, S.: Conformance checking: a state-of-the-art literature review. pp. 1–10. S-BPM ONE '19, ACM (2019)
9. Esparza, J., Nielsen, M.: Decidability Issues for Petri Nets. BRICS Report Series **1**(8) (1994)
10. Fettke, P., Reisig, W.: Systems Mining with Heraklit: The Next Step (2022), arXiv:2202.01289 [cs]
11. Gaaloul, W., Baïna, K., Godart, C.: Log-based mining techniques applied to Web service composition reengineering. Serv. Oriented Comp. Appl. **2**(2), 93–110 (2008)
12. Gutnik, G., Kaminka, G.: A Scalable Petri Net Representation of Interaction Protocols for Overhearing. In: Agent Communication. pp. 50–64. Springer (2005)
13. Hernandez-Resendiz, J.D., Tello-Leal, E., Marin-Castro, H.M., Ramirez-Alcocer et al., U.M.: Merging Event Logs for Inter-organizational Process Mining. pp. 3–26. Springer (2021)
14. Jablonski, S., Bussler, C.: Workflow management: modeling concepts, architecture and implementation. ITP New Media (1996)
15. Kwantes, P., Kleijn, J.: Distributed Synthesis of Asynchronously Communicating Distributed Process Models. In: ToPNoC, pp. 49–72 (2022)
16. Liu, C., Li, H., Zeng, Q., Lu et al., T.: Cross-Organization Emergency Response Process Mining: An Approach Based on Petri Nets. Math. Probl. Eng. **2020**, e8836007 (2020)
17. Liu, C., Li, H., Zhang, S., Cheng et al., L.: Cross-Department Collaborative Healthcare Process Model Discovery From Event Logs. IEEE Trans. Autom. Sci. Eng. **20**(3), 2115–2125 (Jul 2023)
18. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989)
19. Nesterov, R., Bernardinello, L., Lomazova, I., Pomello, L.: Discovering architecture-aware and sound process models of multi-agent systems: a compositional approach. SoSyM (1), 351–375 (2023)
20. Nooijen, E.H.J., van Dongen, B.F., Fahland, D.: Automatic Discovery of Data-Centric and Artifact-Centric Processes. In: BPM Workshops. pp. 316–327 (2013)
21. Peña, L., Andrade, D., Delgado, A., Calegari, D.: An Approach for Discovering Inter-organizational Collaborative Business Processes in BPMN 2.0. In: PM Workshops. pp. 487–498. Springer (2024)
22. Popova, V., Fahland, D., Dumas, M.: Artifact Lifecycle Discovery. Int. J. Coop. Inf. Syst. **24**(01), 1550001 (2015)
23. Stroiński, A., Dwornikowski, D., Brzeziński, J.: A Distributed Discovery of Communicating Resource Systems Models. Trans. Serv. Comput. **12**(2), 172–185 (2019)
24. Tour, A., Polyvyanyy, A., Kalenkova, A., Senderovich, A.: Agent Miner: An Algorithm for Discovering Agent Systems from Event Data. In: BPM. pp. 284–302 (2023)
25. Van Glabbeek, R.J., Weijland, W.P.: Branching time and abstraction in bisimulation semantics. Journal of the ACM **43**(3), 555–600 (May 1996)
26. Zeng, Q., Duan, H., Liu, C.: Top-Down Process Mining From Multi-Source Running Logs Based on Refinement of Petri Nets. IEEE Access **8**, 61355–61369 (2020)
27. Zeng, Q., Sun, S., Duan, H., Liu et al., C.: Cross-organizational collaborative workflow mining from a multi-source log. Decis Support Syst **54**, 1280–1301 (Feb 2013)