Collaboration Petri Nets: Verification, Equivalence, and Discovery (Extended Version)

Janik-Vasily Benzin\(^1\) and Stefanie Rinderle-Ma\(^2\)

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

Abstract. Process modeling and discovery techniques aim to construct sound and valid process models for different types of processes, i.e., process orchestrations and collaboration processes. Orchestrations represent behavior of cases within one process. 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 resource sharing has led to different representations and discovery techniques for collaboration process models, but a standard model class is lacking. We propose collaboration Petri nets (cPN) to achieve comparability between techniques, to enable approach and property transfer, and to build a standardized collaboration mining pipeline similar to process mining. For cPN, we require desirable modeling power, decision power, modeling convenience, and relations to existing model classes. We show the representation of collaboration types, structural characterization as workflow nets, automatic verification of soundness, bisimulation equivalence to existing model classes, and application in a general discovery framework. As empirical evidence to discover cPN, we conduct a comparative evaluation between three discovery techniques on a set of existing collaboration event logs.

Keywords: Collaboration Mining · Collaboration Process Models · Standardisation of Nets · Process Discovery

1 Introduction

Business processes are vital to many domains such as healthcare and manufacturing \cite{12} and define the control-flow of business activities, i.e., what work has to be done in what order \cite{45}. Languages for capturing process logic in process models \cite{45,36} and techniques to discover (business) process models from process executions recorded in an event log \cite{10,13} are are both essential to document, understand, and improve processes \cite{25} such that the business benefits significantly \cite{46}. Among the set of process modeling languages, Petri nets are well-suited due to their formal semantics, graphical nature, expressiveness, analysis techniques, and tool support \cite{61,64}. Moreover, different modeling languages can be transformed to Petri nets \cite{64} and metrics measuring the quality
of discovered models with respect to the event log are only defined for Petri nets [13], which means that the theory of Petri nets is central to process modeling and discovery.

Depending on the type of process, modeling and discovery techniques employ distinct approaches. Process orchestrations define the control-flow for similar cases [10, 13]. Process mining techniques [42] mostly deal with process orchestrations, i.e., process discovery [13] aims at discovering a Petri net from a set of process instances correlated by cases [30] to model a process orchestration. Collaboration processes define the control-flow for similar collaborating cases [17]. 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. Hence, collaboration process discovery (CPD) techniques such as [15, 24, 53] aim at discovering a Petri net from a set of process instances grouped by collaborating cases. A divide-and-conquer approach on the case notion is typical for CPD techniques to apply process discovery on projected event logs.

Collaboration between cases can be heterogeneous [17] and occurs via various types of collaboration concepts such as a hospital’s departments [53], a company’s agents [72], a shop’s objects like orders or packages [5], or the services in a web service composition [69]. Hence, a collaboration process consists of multiple collaboration concepts whose intra-process behavior is modeled as a process orchestration, while their collaborations constitute the inter-process behavior. Although different proposals exist to model and discover collaboration processes in various domains, e.g., composed RM_WF nets for healthcare [53], object-centric Petri nets for commerce [5], and interaction Petri nets for supply-chains and logistics [77, 29, 37], a standard Petri net class is missing [17]. 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.

Advances in collaboration mining are recently focused on CPD [53, 15, 72]. Following this, the requirements for a standard are: (i) sufficient modeling power to represent the control-flow of collaboration processes across domains and types of collaboration concepts, balanced with (ii) enough decision power to decide relevant problems like soundness, (iii) adequate modeling convenience such that models are relatively simple, concise, and particularly suitable for discovery, and (iv) desirable relations to existing model classes. Hence, our research question is: How can we define a standard Petri net class for collaboration process discovery that meets the requirements for a standard?

We propose collaboration Petri nets (cPN) as a standard to model and discover collaboration processes. To illustrate the proposal, Fig. 1 depicts a hospital’s patient treatment collaboration process [77, 76, 53] modeled as a cPN. Collaboration concepts comprise departments “emergency”, “X-ray”, “surgical”, and “cardiovascular”. Patients are initially treated in “emergency”, which collaborates via the shared resource type “charging system” (see p_r,1 in Fig. 1) with
“X-ray” and a handover-of-work message over message channel $p_{ac,1}$ with “cardiovascular”. Other collaborations (inter-process) are a second resource sharing of the two diagnosis rooms $p_{r,2}$, message exchanges via channels $p_{ac,2}, \ldots, p_{ac,7}$, and synchronous activity execution in transitions $t_{sc,1}$ and $t_{sc,2}$ where doctors of “surgical” and “cardiovascular” consult and prescribe together. By focusing on the intra-process behavior of the four departments, four process orchestrations modeled as workflow nets are visible.

![Fig. 1. cPN of treatment collaboration process (simplified from [77,53]).](image)

Section 2 introduces preliminaries. In Sect. 3, we define $cPN$, present initial evidence for their modeling power and convenience, and show decidability of soundness. The discovery framework for $cPN$ in Sect. 4 proves that by applying existing CPD techniques on collaboration event logs in the framework, we discover Petri nets that are weakly bisimilar to $cPN$. To underline favourable properties of $cPN$, we present empirical results that compare three publicly available CPD techniques on an existing dataset in Sect. 5. In Sect. 6, related work is discussed. Lastly, we conclude and give an outlook in Sect. 7.
2 Preliminaries

Let \( X \) be a set. \( \mathcal{P}(X) = \{X' \mid X' \subseteq X\} \) denotes the powerset of \( X \), and \( \mathcal{P}^+ (X) = \mathcal{P}(X) \setminus \emptyset \) (with \( \emptyset \) the empty set) denotes the set of all non-empty subsets of \( X \). Let \( Y \) be another set. A relation \( R \subseteq X \times Y \) with \( X \times Y = \{(x, y) \mid x \in X \land y \in Y\} \) the cartesian product of \( X \) and \( Y \) is a subset of all ordered pairs in the cartesian product. \( \text{DOM}(R) = \{x \in X \mid \exists y \in Y (x, y) \in R\} \) defines the domain of relation \( R \) and \( \text{RNG}(R) = \{y \in Y \mid \exists x \in X (x, y) \in R\} \) defines the range of relation \( R \). Given set \( X' \), \( R|_{X'} = \{(x, y) \in R \mid x \in X'\} \) defines the restriction of relation \( R \)'s domain to \( X' \). The \( n \)-ary cartesian product over sets \( X_1, \ldots, X_n \) is defined by \( X_1 \times \ldots \times X_n = \{(x_1, \ldots, x_n) \mid \forall i \in \{1, \ldots, n\} x_i \in X_i\} \). We write \( (x_i)_{i \in \{1, \ldots, n\}} \in X_1 \times \ldots \times X_n \) for an \( n \)-tuple, which is an element of the \( n \)-ary cartesian product over \( X_1, \ldots, X_n \). If \( X_1 = X, \ldots, X_n = X \), we also write \( X^n \) for \( X_1 \times \ldots \times X_n \). Function application is naturally extended to sets, i.e., \( f(X) = \{f(x) \mid x \in X\} \).

A multiset (or bag) \( m \) over \( X \) is a function \( m : X \to \mathbb{N} \), i.e., \( m(x) \in \mathbb{N} \) or \( x^{m(x)} \) for \( x \in X \) denotes the number of times \( x \) appears in \( m \). For \( x \notin X \), we define \( m(x) = 0 \). \( \mathcal{B}(X) \) denotes the set of all finite multisets over \( X \) and \( \emptyset \) also denotes the empty multiset, i.e., we overload notation. The cardinality of a multiset is defined by \( |m| = \sum_{x \in X} m(x) \). The support of multiset \( m \in \mathcal{B}(X) \) is defined by \( \text{supp}(m) = \{x \in X \mid m(x) > 0\} \), i.e., the support is the set of distinct elements that appear in \( m \) at least once. A set \( X' \subseteq X \) is also the multiset \( m(x') = 1, x' \in X' \). We also write \( m = [x_1^{m(x_1)}, \ldots, x_n^{m(x_n)}] \) for \( \text{supp}(m) = \{x_1, \ldots, x_n\} \).

Given two multisets \( m_1, m_2 \in \mathcal{B}(X) \), we lift the subset relation, addition and subtraction operation to multisets: (i) \( m_1 \subseteq m_2 \) iff for all \( x \in X \) it holds that \( m_1(x) \leq m_2(x) \); (ii) \( m_1 + m_2 \) \( (x) = m_1(x) + m_2(x) \) for each \( x \in X \); and (iii) if \( m_1 \subseteq m_2 \), then \( (m_2 - m_1) \) \( (x) = m_2(x) - m_1(x) \) for each \( x \in X \).

A trace over \( X \) of length \( n \in \mathbb{N} \) is a function \( \sigma : \{1, \ldots, n\} \to X \). \( |\sigma| \) denotes the length of trace \( \sigma \). For \( |\sigma| = 0 \), we write \( \sigma = \epsilon \) and for \( |\sigma| > 0 \), we write \( \sigma = (x_1, \ldots, x_n) \). The set of all finite traces over \( X \) is denoted by \( X^* \).

We write \( x \in \sigma \) for \( x \in X \), if \( \exists i \in \{1, \ldots, |\sigma|\} x = \sigma(i) \). The support of trace \( \sigma \) is \( \text{supp}(\sigma) = \{x \in X \mid \exists 1 \leq i \leq |\sigma| \sigma(i) = x\} \). We define the concatenation \( \nu \cdot \gamma \) of two traces \( \nu, \gamma \in X^* \) by \( \sigma : \{1, \ldots, |\nu| + |\gamma|\} \to X,\sigma(i) = \nu(i) \) if \( 1 \leq i \leq |\nu| \) and \( \sigma(i) = \gamma(i) \) if \( |\nu| + 1 \leq i \leq |\nu| + |\gamma| \). We inductively define the projection of a trace on a set \( Y \) by \( \epsilon|_Y = \epsilon, (\langle x \rangle \cdot \sigma)|_Y = \langle x \rangle \cdot \sigma|_Y \) if \( x \in Y \) and \( \langle (\langle x \rangle \cdot \sigma)|_Y = \sigma|_Y \) otherwise.

Let \( A \) be a finite set of activity labels. A labelled, directed process graph is a 3-tuple \( (S, \Lambda, E) \) where \( S \) is the set of states, and \( E \subseteq S \times A \times S \) is the set of labelled edges, where \( \Lambda_\tau = \Lambda \cup \tau \) for \( \tau \notin A \) the silent activity. For \( (s, \alpha, s') \in E \), we also write \( s \xrightarrow{\alpha} s' \). We write \( s_1 \xrightarrow{\sigma(1)} s_2 \cdots s_n \xrightarrow{\sigma(n)} s_{n+1} \) iff there exists \( \sigma \in \Lambda^*_\tau \) with \( |\sigma| = n \) and \( s_1, \ldots, s_{n+1} \in S \) such that \( s_1 \xrightarrow{\sigma(1)} s_2 \cdots s_n \xrightarrow{\sigma(n)} s_{n+1} \).

We define the weak transition relation \( \xrightarrow{\alpha} \subseteq S \times A_\tau \times S \) with \( \alpha \in A_\tau \) by (i) \( s \xrightarrow{\alpha} s_1 \xrightarrow{\alpha} s_2 \cdots \xrightarrow{\alpha} s' \), 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 notion of the weak
transition relation to traces and write $s_1 \xrightarrow{\sigma} s_{n+1}$ iff there exists $\sigma \in \Lambda^*$ with $|\sigma| = n$ and $s_1, \ldots, s_{n+1} \in S$ such that $s_1 \xrightarrow{\sigma(1)} s_2 \xrightarrow{\sigma(2)} \cdots \xrightarrow{\sigma(n)} s_{n+1}$.

A labelled transition system (LTS) over $\Lambda$ is a labelled, directed process graph with an initial state: $\Gamma_A = (S, \Lambda, s_0, \rightarrow)$, where $s_0$ is the initial state. We define the equivalence relation (rooted) weak bisimulation equivalence on the set of all LTSs over $\Lambda$. Let $\Gamma_I = (S_1, \Lambda, s_{0,1}, \rightarrow_1)$ and $\Gamma_F = (S_2, \Lambda, s_{0,2}, \rightarrow_2)$ be two LTSs. A relation $R \subseteq S_1 \times S_2$ is a (rooted) weak bisimulation equivalence, denoted by $\Gamma_I \approx_R \Gamma_F$, iff (i) $(s_{0,1}, s_{0,2}) \in R$, i.e., the initial states are related; and for every $(p, q) \in R$ and $\alpha \in \Lambda$, it holds that (ii) if $p \xrightarrow{\alpha} p'$, then $\alpha = \tau$ and $(p', q) \in R$, or there exists $q' \in S_2$ such that $q \xrightarrow{\alpha} q'$ and $(p', q') \in R$; and (iii) if $q \xrightarrow{\alpha} q'$, then $\alpha = \tau$ and $(p, q') \in R$, or there exists $p' \in S_1$ such that $p \xrightarrow{\alpha} p'$ and $(p', q') \in R$. If $\Gamma_I \approx_R \Gamma_F$, we say $\Gamma_I$ is weakly bisimilar to $\Gamma_F$.

A labelled Petri net is a 5-tuple $N = (P, T, F, l, \Lambda)$, where $P$ is the set of places, $T$ is the set of transitions with $P \cap T = \emptyset$, $F \subseteq \left((P \times T) \cup (T \times P)\right)$ is the flow relation, and $l : T \to \Lambda$ is the transition labelling function. 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 $m \in \mathcal{B}(P)$ 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. Given labelled Petri net $N, \mathcal{N}(N) = \{(N, m) \mid m \in \mathcal{B}(P)\}$ denotes the set of all marked Petri nets of $N$. The transition enabling $(N, m)[t]$ for $t \in T$ is defined by $(N, m)[t]$ iff $m(p) \geq 1$ for all $p \in \bullet t$. An enabled transition $(N, m)[t]$ can fire, which remove a token from each of it’s input places, adds a token to each of it’s output places, and executes an activity $\alpha \in \Lambda$ represented by $t(\alpha)$. We define this behavior as a relation called the firing rule $\cdot \mid \cdot \subseteq \mathcal{N}(N) \times \Lambda \times \mathcal{N}(N)$ by $(N, m)[t](N, m')$ iff $(N, m)[t]$ and $m' \mathrel{\bullet t} = m + t \bullet$.

The semantics of a marked Petri net $(N, m_0)$ with $N = (P, T, F, l, \Lambda)$ is defined by the LTS $\Gamma_{N, m_0} = (\mathcal{B}(P), \Lambda, m_0, \rightarrow)$ with $m \xrightarrow{\alpha} m'$ iff $\alpha = l(t)$ and $(N, m)[t](N, m')$. $(N, m_0)$ is weakly bisimilar to marked Petri net $(\Lambda', m_0')$ iff their semantics are. A trace $\sigma \in \Lambda^*$ is a firing trace of $(N, m_0)$ iff $m_0 \xrightarrow{\sigma} m'$. We omit the labelled Petri net $N$, if the context is clear. We define the set of reachable markings by $\mathcal{R}(N, m_0) = \{ m' \mid \exists \sigma \in \Lambda^* m_0 \xrightarrow{\sigma} m' \}$, given marking $m' \in \mathcal{B}(P)$, we denote the set of all finite firing traces that reach $m'$ in $(N, m_0)$ by $\mathcal{L}_\tau(N, m_0, m') = \{ \sigma \in \Lambda^* \mid m_0 \xrightarrow{\sigma} m' \}$. The trace language only contains observable activities $a \in \Lambda$ and is defined by $\mathcal{L}(N, m_0) = \{ \sigma \in \Lambda^* \mid m_0 \xrightarrow{\sigma} m' \}$.

A marked Petri net $(N, m_0)$ is bounded iff its set of reachable markings $\mathcal{R}(N, m_0)$ is finite. $(N, m_0)$ is live iff for every transition $t \in T$ and reachable marking $m \in \mathcal{R}(N, m_0)$, there exists a reachable marking $m' \in \mathcal{R}(N, m)$ such that $(N, m)[t]$. A transition $t \in T$ is dead iff there is no reachable marking $m \in \mathcal{R}(N, m_0)$ such that $(N, m)[t]$. $N$ 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 that its postset is empty: $\bullet o = \emptyset$; and (iii) every node $x \in P \cup T$ is on a directed path from $i$ to $o$. $N$ is sound iff (i) it is weakly terminating (also called option to complete), i.e., for all reachable markings.
Collaboration Petri Nets

In this section, we introduce collaboration Petri nets \( (cPN) \) as a standardized Petri net class to model collaboration processes. \( cPN \) are designed to reuse concepts, patterns, and analysis techniques from existing work on collaboration processes and process orchestrations such that we can apply the vast existing body of knowledge and tools without modifications. As we will show, \( cPN \) are WF-nets, i.e., we can recursively model and discover collaboration processes at various granularity levels. Consequently, our model of collaboration processes is equivalent to process orchestrations in its modeling and decision power, a result that is not obvious from the multitude of Petri net extensions proposed in existing work. With respect to modeling convenience, \( cPN \) explicitly represent all four of the known collaboration types \([17]\) similar to \( cPN \)\( [77, 76, 53] \), in a declarative, concise, standardized, modular, and correct definition.

3.1 Building Blocks of Collaboration Processes

To model collaboration processes, we first identify relevant building blocks that we formalize step-by-step in this section to finally construct \( cPN \) in the next section. To begin with, collaboration processes are studied in various research areas \([17]\). Research areas with a modeling focus are process choreographies \([28, 29, 37, 55, 38]\), service compositions \([66, 9, 7]\), multi-agent systems \([70, 71]\), interorganizational workflows \([511]\), virtual enterprises \([22, 44, 43]\), and distributed business processes \([19, 68]\), while object-centric process mining \([8, 15]\) and collaboration mining \([40, 77, 69, 75, 24, 50, 59, 53, 72]\) are research areas with a discovery focus. Despite the variety of areas and approaches, collaboration processes are conceptualized as a set of collaboration concepts whose process orchestrations collaborate. Below, we formalize collaboration concepts \( C \) and the collaboration concepts involved in a single collaboration process.

Definition 1 (Collaboration Concept, Concept Collection). Let \( C \) be a set of collaboration concepts. We define a (collaboration) concept collection \( cc \) of length \( n \in \mathbb{N} \) to be an injective trace over \( C \): \( cc : \{1, \ldots, n\} \rightarrow C \).

By means of the concept collection \( cc \), we group the subset of collaboration concepts that are involved in a collaboration process, e.g., \( cc(1) = \) “emergency”, \( cc(2) = \) “X-ray” with \( n = 4 \) in Fig. 1. Because each collaboration concept’s dynamic behavior is a process orchestration, every collaboration concept \( c \in \text{RNG}(cc) \) is assigned a WF-net that models the collaboration concept’s process orchestration. We formalize this notion as a workflow collection:
**Definition 2 (Workflow Collection).** Let $cc$ be concept collection. A workflow collection is a $|cc|$-tuple $WC = (N_c)_{c \in RNG(cc)}$ of WF-nets $N_c = (P_c, T_c, F_c, l_c, \Lambda)$ such that $\forall k, l \in \{1, \ldots, n\}$ if $k \neq l$, then $(P_{cc(k)} \cup T_{cc(k)}) \cap (P_{cc(l)} \cup T_{cc(l)}) = \emptyset$. We define:

- $T^u = \bigcup_{c \in RNG(cc)} T_c$, $P^u = \bigcup_{c \in RNG(cc)} P_c$, $F^u = \bigcup_{c \in RNG(cc)} F_c$, the sets of transitions, places, and arcs of the workflow collection respectively,
- $l^u : T^u \rightarrow \Lambda$, $l^u(t) = l_c(t)$ for $t \in T_c$.

Existing work on collaboration processes shares the conceptualization (modulo naming) up to this point, i.e., collaboration processes are modelled as a set of WF-nets to the best of our knowledge. Since the WF-nets in a collaboration process collaborate, they must be composed to represent the overall collaboration process. Each existing approach models the composition in a certain way, but they all are formalized according to the way WF-nets collaborate, i.e., the type of composition is determined by the collaboration type studied.

There are four collaboration types: Handover-of-work, message exchange, resource sharing, and activity execution [76,17]. The first three types are asynchronous, and the fourth is synchronous as depicted with their respective, typical Petri net pattern to model the collaboration type in Fig. 2. For example, the discovery techniques in [76,54,53,72] represent handing the treatment of patients over in healthcare collaboration processes using the depicted handover-of-work pattern. The pattern includes an asynchronous collaboration place $p_{ac}$ (denoted by a grey filling) that must be marked before the first transition in $N_{cc(1)}$ can fire. The handover-of-work type is a special case of the message exchange type and determined by condition $\text{HO}$ given some collaboration concept $c \in RNG(cc)$, a transition $t \in i_c \bullet$ in the postset of source place $i_c$ in WF-net $N_c$ has the asynchronous collaboration place $p_{ac}$ in its preset $p_{ac} \in \bullet$ [76]. Although the resource sharing pattern with place $p_r$ (denoted with dashed outline in Fig. 1 and Fig. 2) does not affect the behavior of the models in [77,76,54] due to the lack of a time notion, it can be seen as a precursor of mining enhanced models that introduce a time notion such as the healthcare Petri net with time delays for transition firings in [54]. We follow the same path and keep a time notion out of

![Fig. 2. Petri net patterns to model collaboration between process orchestrations.](image-url)
our formalizations bearing in mind that it can be easily added later. Lastly, the activity execution pattern includes a transition \( t_{sc} \) whose firing is synchronized across multiple WF-nets, as, for example, employed in [15,35]. Since our aim is to standardize and integrate, we cover all existing collaboration types in a collaboration pattern as follows.

**Definition 3 (Collaboration Pattern).** Let \( WC = (N_c)_{c \in \text{RNG}(cc)} \) be a workflow collection. A collaboration pattern is a tuple \( CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET) \), where:

1. \( P_{AC} \) is the set of asynchronous collaboration places that do not intersect with existing names, i.e., \( P_{AC} \cap (P^u \cup T_u) = \emptyset \).
2. \( P_{RS} \subseteq P_{AC} \) is the set of shared resource collaboration places.
3. \( ra : P_{RS} \rightarrow \mathbb{N}^+ \) is the resource allocation function, i.e., for shared resource type \( p_r \in P_{RS} \), there exist \( ra(p_r) \) shared resources,
4. \( AC = \{(p_{ac}, T_s, T_r) \in P_{AC} \times P^+(T_u) \times P^+(T_u) \mid \forall t \in T_s, t' \in T_r, l^u(t) \neq \tau \land l^u(t') \neq \tau \} \) is the asynchronous collaboration relation, i.e., \( (p_{ac}, T_s, T_r) \) with channel \( p_{ac} \notin P_{RS} \) denotes that transitions \( t \in T_s \) send a message and transitions \( t' \in T_r \) receive a message via channel \( p_{ac} \).
5. for every \( p_r \in P_{RS} \) there exists \( (p_r, T_1, T_2) \in AC \) such that \( T_1 = T_2 \), i.e., resource types are used and released in transitions \( t \in T_1 \), and
6. \( ET = \{(t_{sc}, T_{sc}) \in T_u \times P^+(T_u) \mid t_{sc} \in T_{sc} \land l^u(t_{sc}) \neq \tau \land \forall t, t' \in T_{sc}, l^u(t) = l^u(t') \} \) is the relation of synchronous collaborations induced by equally-labelled transitions.

A collaboration pattern \( CP_{WC} \) specifies how the WF-nets in the workflow collection collaborate in terms of the four collaboration types. All three asynchronous collaboration types are represented by \( P_{AC} \) (cf. 1 in Def. 3) and \( AC \) (4). The resource sharing type is differentiated by \( P_{RS} \) (2), the resource allocation \( ra \), and the self-loop requirement in (5) from the other two asynchronous types. Message exchange and handover-of-work are not explicitly distinguished in the collaboration pattern, because their only difference is captured by condition [10]. Although synchronized transitions \( t \in T_u \) with \( t \in T_{sc}, (t_{sc}, T_{sc}) \in ET \) are pre-determined by equal labels (6) following CPD techniques with synchronous collaboration [10,60,68,69,53,15], we still formalize them in the collaboration pattern for the sake of completeness.

A collaboration pattern \( CP \) is powerful and convenient in representing collaborations (see Sect. 4). For example, \( CP \) can represent many web service interaction patterns [10] as message exchanges along the dimensions of \# of services involved in an exchange (\( T_s \) or \( T_r \) may contain transitions from an arbitrary number of collaboration concepts), the \# of exchanges (transitions in \( T_s \) or \( T_r \) may be in a loop in their respective WF-net \( N_c \)), and the difference between round-trip interactions (two channels \( p_{ac} \) with alternating sender and receiving collaboration concepts) and routed interactions (a transition is allowed to be both a receiver from one and sender to another collaboration concept).

In the following section, we will compose the WF-nets in the workflow collection by means of the collaboration pattern into a collaboration Petri net.
3.2 Collaboration Petri Nets

The definition of the collaboration Petri net given a workflow collection and collaboration pattern takes the same definition approach as the interorganizational workflow in [5] with extensions and modifications to cover all four collaboration types. The modifications and extensions are defined such that the structure of the resulting Petri net remains a WF-net, which we will show in the next section.

Definition 4 (Collaboration Petri Net (cPN)). Let \( CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET) \) be a collaboration pattern with \( WC = (N_c)_{c \in RNG(c)} \) a workflow collection. A Collaboration Petri Net is a marked Petri net \( \bigcup_{c \in RNG(c)} N_c \) defined as\footnote{We choose to abbreviate collaboration Petri nets with cPN to mitigate confusion with the abbreviation cPN of colored Petri nets [10].}

1. \( P = P^u \cup P_{AC} \cup \{i, o\} \),
2. \( T = r(T^u) \cup \{t_i, t_o\} \), with \( r \) a renaming function: \( r(x) = t_{sc} \) if there exists a \((t_{sc}, T_{sc}) \in ET \) such that \( t \in T_{sc} \), otherwise \( r(x) = x \),
3. \( \{i, o, t_i, t_o\} \cap (P^u \cup T^u \cup P_{AC}) = \emptyset \),
4. \( F' = F^u \cup \{(t, p) \in T^u \times P_{AC} \mid (p, x, y) \in AC \land t \in x \} \cup \{(p, t) \in P_{AC} \times T^u \mid (p, x, y) \in AC \land t \in y \} \cup \{(i, t_i), (t_o, o)\} \cup \{(t_i, t_o) \mid c \in RNG(\vert c)\} \cup \{(o_c, t_o) \mid c \in RNG(\vert c)\} \),
5. \( F = \{(r(x), r(y)) \mid (x, y) \in F'\} \),
6. \( l(t) = l_i(t) \) if \( t \in T^u_s \), \( l(t) = \tau \) otherwise,
7. \( m_0(p) = 1 \) if \( p = i \), \( m_0(p) = ra(p) \) if \( p \in P_{RS} \) and \( m_0(p) = 0 \) otherwise.

Observe that the example in Fig. 1 is indeed a cPN by definition. Aside from these properties of cPN that mostly determine modeling convenience, our approach to separate the WF-nets in the workflow collection, from their collaborations in the collaboration pattern, and their resulting collaboration process model in the cPN resembles, yet clarifies the compositional approach commonly taken for modeling and discovery of collaboration processes.

In the last section, we have informally introduced the four collaboration net patterns in Fig. 2. Below, we formalize them:

Definition 5 (Collaboration Types). Let \( CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET) \) be a collaboration pattern with \( WC = (N_c)_{c \in RNG(c)} \) a workflow collection. We define \( T = \{v_s, v_m, v_h, v_r\} \) the set of collaboration types. Then,

- Activity execution, denoted by \( v_s \), is contained in \( CP \) iff \( ET \neq \emptyset \),
- Message exchange, denoted by \( v_m \), is contained in \( CP \) iff \( P_{AC} \setminus P_{RS} \neq \emptyset \) \& \( \exists (p, t, T) \in AC \) \( p \in P_{AC} \),
- Handover-of-work, denoted by \( v_h \), is contained in \( CP \) iff it contains \( v_m \) such that the handover condition \( [HO] \) holds, and
- Resource sharing, denoted by \( v_r \), is contained in \( CP \) iff \( P_{RS} \neq \emptyset \).
We say that collaboration Petri net $cPN = \bigcup_{c \in \text{RNG}(cc)} N_c$ contains collaboration types $\Upsilon' \subseteq \Upsilon$ iff every collaboration type $\upsilon \in \Upsilon'$ is contained in collaboration pattern $CP$.

Hence, a collaboration pattern can represent all four collaboration types and in the collaboration Petri net are exactly the Petri net patterns as depicted in Fig. 2, e.g., $t_7, p_{ac,1}, t_{16}$ is the Petri net pattern for handover-of-work in Fig. 1.

Multiplicities represented by arc weights and other higher-level net concepts such as token colors [49,64] or variable arcs [8] are missing. Avoiding higher-level net concepts strikes a balance between the four requirements for a standard (cf. Sect. 1). On the one hand, modeling power is slightly reduced such that collaboration processes in which multiple process orchestration instances of a collaboration concept are required for a single execution of the collaboration processes (cf. [76,8]) cannot be represented. Hence, token colors are not needed to distinguish the process instances running in the same process orchestration, as there is only one. The resulting representational bias [3] is chosen by the majority of modeling and discovery techniques [5,28,40,9,77,54,75,51,69,50,59,53,72], which we take as evidence that our standard’s representational bias is acceptable for most collaboration processes and domains (cf. Tab. 1). On the other hand, more relevant problems are decidable and likely fall into a more desirable complexity class. Moreover, a $cPN$ is likely to be more understandable due to its restriction on basic net concepts, i.e., modeling convenience for practitioners is better.

Interestingly, typed Jackson nets [15] employ token colors in the form of identifiers to represent the collaboration process in a continuously running state, but still require a one-to-one correspondence between types in synchronous collaborations within their discovery framework. A $cPN$ models the collaboration process in a static state, i.e., for a single execution similar to WF-nets for process orchestrations. The next section shows that a $cPN$ is a WF-net, from which we conclude decidability of soundness.

### 3.3 Automated Verification of Collaboration Petri Nets

We start by showing that a $cPN$ is a WF-net:

**Lemma 1.** Let $cPN = ((P, T, F, l, A_\tau), m_0) = \bigcup_{c \in \text{RNG}(cc)} N_c$ be a collaboration Petri net. $cPN$ is a WF-net.

**Proof.** It follows from Def. 4 (1), (3), (4), and (5) that the only source place $i$ and sink place $o$ are connected to the source places $i_c$ and $o_c$ of the workflow nets $N_c$ for $c \in \text{RNG}(cc)$ via $t_i$ and $t_o$ respectively. Hence, any node in workflow net $N_c$ lies on a directed path from $i$ to $o$. Moreover, asynchronous places $p_{ac} \in P_{AC}$ with $(p_{ac}, T_s, T_r) \in AC$ lie on a directed path from transitions $t \in T_s$ to transitions $t' \in T_r$ as follows from Def. 4 (4) and (5). Fused transitions $t \in T_{sc}$ for some $(t_{sc}, T_{sc}) \in ET$ lie on a directed path within all their fused collaboration concept’s WF-nets $N_c$ as follows from Def. 4 (2), (4), and (5).
As mentioned in Sect. 3.1, resource places $p_r$ leave behavior unaffected without a time notion in our model, but still render the classical soundness definition inapplicable. However, soundness is meaningful in our untimed setting, which is expressed in a slightly changed soundness definition for $cPN$.

**Definition 6 (Sound $cPN$).** Let $cPN = ((P, T, F, l, A_{\tau}), m_0) = \bigcup_{c \in \text{RNG}^{cc}(c)} N_c$ be a collaboration Petri net. We say $cPN$ is sound if the WF-net resulting from removing all shared resource places $p_r \in P_{RS}$ and their arcs through the “elimination of self-loop places” reduction rule [56] is sound.

We are able to verify soundness of collaboration Petri nets.

**Theorem 1 (Decidability of Soundness for $cPN$).** Soundness for the collaboration Petri net $cPN = ((P, T, F, l, A_{\tau}), m_0) = \bigcup_{c \in \text{RNG}^{cc}(c)} N_c$ is decidable.

**Proof.** Observe that the reduction rule “elimination of self-loop places” preserves liveness, safety and boundedness [56] and that reducing the $cPN$ to a collaboration Petri net without any shared resource places does not break the WF-net structure, as the reduction is similar to constructing $cPN$ with collaboration pattern $CP$ that does not contain any shared resources, i.e., $P_{RS} = \emptyset$. Since soundness is decidable for WF-nets through reducing it to deciding boundedness and liveness in the short-circuited WF-net in which the sink place is connected to the source place via a transition labelled with $\tau$ [2], it follows from Lemma 1 that soundness is decidable for collaboration Petri nets.

To sum up, $cPN$ have sufficient modeling power to represent all collaboration types and soundness is decidable. We can reuse existing results and techniques developed for modeling and analysing process orchestrations also for collaboration processes by avoiding higher-level net concepts and other structural extensions. On top, we can recursively stick a $cPN$ into another workflow collection at a higher granularity level such that we can seamlessly model collaboration at any granularity level. Next, we move from modeling to discovering $cPN$.

### 4 Discovering Collaboration Petri Nets

Figure [3] presents the overall framework to discover collaboration Petri nets $cPN$ (all related concepts are defined in the following sections): Given collaboration event log $cL$ converted to $cL'$ in the required input format of CPD technique discover, converted log $cL'$ is projected to each collaboration concept’s event log $L^{cc(1)}_{cc}, \ldots, L^{cc(n)}_{cc}$. Then, process discovery technique disc discovers WF-nets $N^{cc(1)}_{cc}, \ldots, N^{cc(n)}_{cc}$, while some custom mining functionality cdisc mines asynchronous collaborations as Petri net patterns resulting in $N_{ac}$ (synchronous collaborations are always determined by equal labels). All results are composed with some custom model composition $\text{comp}(N^{cc(1)}_{cc}, \ldots, N^{cc(n)}_{cc}, N_{ac}) = N$. If discover has property $Q_{discover}$, we can construct WF-nets $N^{cc(1)}_{cc}, \ldots, N^{cc(n)}_{cc}$ to simulate $N_{ac}$ and $N$ is weakly bisimilar to collaboration Petri net $cPN$. Because
many existing CPD techniques (cf. Sect. 5) have $Q_{\text{discover}}$, collaboration Petri nets achieve comparability between CPD techniques despite the heterogeneity of collaboration concept types, collaboration types, and discovered models.

### 4.1 Generating Collaboration Event Logs

In short, a collaboration event log $cL$ is generated by a collaboration Petri net $cPN = (N, m_0)$ that models the real-world collaboration process. If we only record the trace of (observable) activity labels $\sigma \in \Lambda^*$ in a firing trace $(N, m_0) \xrightarrow{\sigma} (N, m)$ for some $m \in \mathcal{R}(N, m_0)$ to generate a collaboration event log, we do not have any information on collaborations, i.e., synchronous activities, message exchanges and resource requirements are unknown. To additionally record information on collaborations during execution of a $cPN$, we propose the collaboration labelling $l_{cl}$ that replaces $l$ in a $cPN$.

**Definition 7 (Collaboration Labelling).** Let $CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET)$ be a collaboration pattern with $WC = (N_c)_{c \in \text{RNG}(cc)}$ a workflow collection. Collaboration labelling is a function $l_{cl}: T^u \cup \{t_i, t_o\} \rightarrow \Lambda_{cl}$ with collaboration labels $\Lambda_{cl} = \Lambda \times \mathcal{P}(\text{RNG}(cc)) \times \mathcal{P}(P_{AC})^3$ for $t_i, t_o \notin T^u$:

$$l_{cl}(x) = \begin{cases} (l^u(t), \text{sync}(T_{sc}), \text{send}(x), \text{rec}(x), \text{res}(x)) & \text{if } (x, T_{sc}) \in ET \land t \in T_{sc} \\ (l^u(x), \{c\}, \text{send}(x), \text{rec}(x), \text{res}(x)) & \text{if } (x, T_{sc}) \notin ET \land \exists c \in \text{RNG}(cc) \land x \in T_{c} \\ \tau & \text{otherwise.} \end{cases}$$

where

- $\text{sync}(T_{sc}) = \{c \in \text{RNG}(cc) \mid t \in T_{sc} \land t \in T_{c}\}$
- $\text{send}(x) = \{p_{ac} \mid \exists (p_{ac}, T_s, T_r) \in AC \land x \in T_s \land p_{ac} \notin P_{RS}\}$
- $\text{rec}(x) = \{p_{ac} \mid \exists (p_{ac}, T_s, T_r) \in AC \land x \in T_r \land p_{ac} \notin P_{RS}\}$, and
- $\text{res}(x) = \{p_r \mid \exists (p_r, T_1, T_2) \in AC \land x \in T_1 \land p_r \in P_{RS}\}$.

---

2 A $cPN$ used to generated $cL$ is a reference model with which we can compare model quality metrics computed on $cPN$ and a discovered $cPN'$, i.e., our framework builds the foundation to study the maturity of CPD techniques.
We call \((activity, concepts, send, receive, resource) \in A_d\) a collaboration label.

With the collaboration labelling \(l_d\), we can generate collaboration event logs \(cL\) given a collaboration Petri net \(cPN\) as follows:

**Definition 8 (Collaboration Event Log).** Given a set of activity labels \(\Lambda\), a set of traces \(L \subseteq \Lambda^*\) is called an event log. An event log is generated by a marked Petri net \((N, m_0)\), if \(m_0 \xrightarrow{\sigma} m\) with \(m \in B(P)\) for all \(\sigma \in L\), i.e., an event log is a subset of the trace language \(L \subseteq \mathcal{L}(N, m)\). For a collaboration Petri net \(cPN = ((P, T, F, l_d, A_d), m_0)\) with collaboration labelling, we call event log \(cL \subseteq A_d^*\) generated by \(cPN\), i.e., \(cL \subseteq \mathcal{L}(cPN)\), a collaboration event log.

Note that the information recorded in \(cL\) is on the highest logging level according to logging levels specified in [67,32,31,41] for discovery of web service compositions, i.e., collaboration processes. The highest logging level is sufficient to discover the complete, executable collaboration process by some CPD technique discover [50], if no multiplicities between collaboration concept’s WF-net instances are present (cf. Sect. 3.2).

We can convert \(cL\) into an eXtensible Event Stream (XES) [11] event log \(cL'\) (cf. Fig. 3), in which a collaboration label’s activity corresponds to the concept: name, it’s concepts correspond to the org: resource, it’s send corresponds to the message: sent, it’s receive corresponds to the message: rec, and it’s resource to the resource event attributes as specified in [53]. Also, we give an intuition on how we can convert \(cL\) into an object-centric event log (OCEL) \(cL'\) by defining object types for each collaboration concept \(c \in RNG(cc)\) and each asynchronous type \(p_{ac} \in P_{AC}\). Each event’s object map denoted by omap\((e)\) in the OCEL \(cL'\) that maps object types to object identifiers such that the respective objects are associated with the event has to be set. Each object type can be taken as a case notion [8]. As \(\sigma_j(i) \in A_d\) for \(\sigma_j \in cL\), \(i \in \{1, \ldots, |\sigma_j|\}\) corresponds to an event in \(cL'\), set omap\((e)\)\((c) = j\) if \(c \in concepts_{i,j}\), omap\((e)\)\((p_{ac}) = \langle p_{ac}, j, \#_j(p_{ac} mode)\rangle\) if \(p_{ac} \in send_{i,j} \cup receive_{i,j}\) and \(p_{AC} \in P_{AC} \setminus P_{RS}\), and omap\((e)\)\((p_{ac}) = j\) if \(p_{AC} \in resource_{i,j}\) and \(p_{AC} \in P_{RS}\) in ascending order of \(i\). Overall, it is important to observe that the conversions are bijections without information loss such that we can convert \(cL'\) back into \(cL\).

As CPD techniques discover typically employ a divide-and-conquer approach to discover collaboration process models \(M\) from \(cL\) as depicted in Fig. 3, the first step of discover is a log projection as follows.

**Definition 9 (Log Projection).** Let \(cL \subseteq \mathcal{L}(cPN)\) for collaboration Petri net \(cPN\) be a collaboration event log. Log projection is defined by \(\pi_c(cL) = \{\sigma_c \mid \sigma_c \in cL \land \sigma_c = \sigma|_{A_c \times \{X \subseteq \mathcal{P}(RNG(cc))|c \in X\} \times \mathcal{P}(P_{AC})}\}\) for \(c \in RNG(cc)\).

In the next section, we show that the asynchronous collaboration types as represented by \(P_{AC}\) and \(AC\) are syntactic sugar for the synchronous collaboration type represented by \(ET\) in a collaboration pattern \(CP\).

---

\(^3\) \#_j is a counter function counting how often a message of type \(p_{AC}\) is sent/received as passed by argument mode in \(\sigma_j\) such that a first-in first-out message queue is assumed.
4.2 Minimal Collaboration Types

Taking the modeling power point of view, collaboration types can be simulated by another, i.e., we can reduce the number of collaboration types to a minimal set while maintaining weak bisimilarity between the respective collaboration Petri nets. We show that synchronous collaboration is minimal. To prove our statement, a relation on the set of collaboration types $\mathcal{Y}$ is required. We aim for a relation that at least partially orders the powerset of collaboration types $\mathcal{P}(\mathcal{Y})$ such that $\mathcal{Y}$ is the maximal element and, ideally, a unique minimal element exists. A minimal element $\mathcal{Y}'$ means that all other collaboration types $\upsilon \in \mathcal{Y} \setminus \mathcal{Y}'$ are syntactic sugar and are included in collaboration Petri nets to improve the modeling convenience.

**Definition 10 (Simulating Collaboration Types, Minimality).** A non-empty set of collaboration types $\mathcal{Y}_1 \subseteq \mathcal{Y}$ simulates collaboration types $\mathcal{Y}_2 \subseteq \mathcal{Y}$ with $\mathcal{Y}_1 \subseteq \mathcal{Y}_2$, denoted by $\mathcal{Y}_1 \preceq \mathcal{Y}_2$ iff for any collaboration Petri net $c\mathcal{P}N$ that contains collaboration types $\upsilon \in \mathcal{Y}_2$ and is defined on workflow collection $WC = (N_c)_{c \in \text{RNG}(cc)}$, there exists a collaboration Petri net $c\mathcal{P}N'$ that only contains types $\upsilon' \in \mathcal{Y}_1$ and is defined on $WC' = (N_c)_{c \in \text{RNG}(cc')}$ with $cc' = cc \cdot cc''$ such that $c\mathcal{P}N \approx c\mathcal{P}N'$. A subset $\mathcal{Y}_m \subseteq \mathcal{Y}$ of collaboration types is minimal iff collaboration types $\mathcal{Y}_m$ simulate $\mathcal{Y}$ and there exists no other set of collaboration types $\mathcal{Y}_m'$ with $|\mathcal{Y}_m'| < |\mathcal{Y}_m|$ such that collaboration types $\mathcal{Y}_m'$ simulate $\mathcal{Y}$.

In the construction of a $c\mathcal{P}N'$ that simulates $c\mathcal{P}N$, it is important to prohibit changes to the workflow collection $WC = (N_c)_{c \in \text{RNG}(cc)}$ except than extending to $WC'$, as other changes to $WC$ would mean that we allow simulating collaboration types by a collaboration concept’s internal WF-net and not by some other collaboration type. Note that $\preceq \subseteq \mathcal{Y} \times \mathcal{Y}$ defines a partial order on collaboration types. In the following, we show that the synchronous collaboration type is a minimal element of $\preceq$, under two conditions: the empty trace is not in the trace language and no $\tau$-labelled loop for asynchronous collaboration exists. A $\tau$-labelled loop in $c\mathcal{P}N = (N,m_0)$ exists for asynchronous collaboration $(p_{ac},T_s,T_r) \in AC$ iff $m_1 \xrightarrow{\{t\}} m_2$, $m_1 \in \mathcal{R}(N,m_0)$ and $m_2 \xrightarrow{\tau} m_3$ such that $(N,m_3)[t']$ for some $t,t' \in T_s$. The following theorem proves our statement with an efficient construction in the sense that the constructed, synchronous collaboration only $c\mathcal{P}N_{ac}$ that is weakly bisimilar to a given $c\mathcal{P}N$ does not copy complete WF-nets of collaboration concepts.

**Theorem 2 (Synchronous activity execution is minimal).** $\mathcal{Y}' = \{\upsilon_s\}$ is minimal, if no asynchronous collaboration $(p_{ac},T_s,T_r) \in AC$ with $\tau$-labelled loops in $c\mathcal{P}N$ exists and $\epsilon \notin \mathcal{L}(c\mathcal{P}N)$.

**Proof.** Let $c\mathcal{P}N = ((P,T,F,l,A_r),m_0) = \bigcup_{c \in \text{RNG}(cc)} N_c$ be a collaboration Petri net that contains collaboration types $\upsilon \in \mathcal{Y}$ with $c\mathcal{P}WC = (P_{AC},P_{RS},ra,AC,ET)$ a collaboration pattern and $WC = (N_c)_{c \in \text{RNG}(cc)}$ a workflow collection. Observe that labelling of transitions with observable activities $\alpha \neq \tau$, $\alpha \in A_r$ is a bijection in any collaboration Petri net (cf. Def. 3), so we can use activity
that mark the global sink place such that \(\forall\) source type becomes a collaboration concept. Let \(M\) collaboration to a collaboration concept, i.e., every message channel and re-

\[\text{synchronously with activities "after" which sending cannot occur anymore. Also,}\]

\[\text{also, we distinguish three cases of asynchronous collaboration. Given}\ (p_{ac}, T_s, T_r) \in AC, \text{either (i) } p_{ac} \text{ is dead, i.e., } p_{ac} \text{ is dead iff all } t \in T_s \text{ are dead or for every firing trace } \sigma \in \bigcup_{m \in M_f(cPN)} \mathcal{L}_r(N, m_0, m) \text{ that marks the global sink place it holds that } \forall_{\alpha \in l(T_s)} \alpha \notin \sigma; \text{(ii) } p_{ac} \text{ is optional, i.e., } p_{ac} \text{ is optional if there exists at least two firing traces } \sigma, \sigma' \in \bigcup_{m \in M_f(cPN)} \mathcal{L}_r(N, m_0, m) \text{ that mark the global sink place such that } \forall_{\alpha \in l(T_s)} \alpha \notin \sigma \text{ (no sending activity occurs in } \sigma) \text{ and } \exists_{\alpha \in l(T_s)} \alpha \in \sigma' \text{ (a sending activity occurs in } \sigma'); \text{ and (iii) } p_{ac} \text{ is compulsory, i.e., } p_{ac} \text{ is compulsory iff for every firing trace that marks the global sink place } \sigma \in \bigcup_{m \in M_f(cPN)} \mathcal{L}_r(N, m_0, m) \text{ it holds that } \exists_{\alpha \in l(T_s)} \alpha \in \sigma \text{ (at least one sending activity always occurs)}. \text{ Note that we can distinguish these three cases by analysing the coverability graph} [56][52]. \text{ However, we do not have to determine for each asynchronous collaboration what case applies, as we only have to compute certain markings with respective sets of activities in the coverability graph that we will define in the following.} \text{ We can ignore case (i) to construct a weakly bisimilar } cPN_{os}, \text{ as the sending activities of } p_{ac} \text{ can never occur in any firing trace.} \text{ Due to (ii), we have to ensure that the WF-net } N_{p_{ac}} \text{ that simulates optional asynchronous collaboration } p_{ac} \text{ with synchronous collaboration is able to skip all transitions with sending activities, as otherwise sink place } o_{p_{ac}} \text{ cannot be marked such that } [o_{p_{ac}}] \text{ becomes unreachable. Skipping has to occur synchronously with activities “after” which sending cannot occur anymore. Also, repeating sending activities by some loop has to be allowed and synchronized with activities “after” which sending occurs again.} \text{ Determine by breadth-first search for each } (p_{ac}, T_s, T_r) \text{ a unique set of markings } M_\alpha \subseteq \mathcal{R}(N, m_0) \text{ and set of markings } M_\gamma \subseteq \mathcal{R}(N, m_0) \text{ in the coverability graph.} \text{ } M_\alpha \text{ is characterized by the following formula: for each } m_\alpha \in M_\alpha \text{ it holds that } m_\alpha \text{ is reachable by paths in the coverability graph in which no sending activities } \alpha \in l(T_s) \text{ have occurred and formula } \gamma(m_\alpha) \text{ holds. } \gamma(m) \text{ holds iff there exist two sets of activities } A_0, A_1 \subseteq A, \tau \notin A_1 \text{ with } m \overset{\alpha_0}{\Rightarrow} m_\alpha, \alpha_0 \in A_0 \text{ such that for all } m' \in \mathcal{R}(N, m_\alpha) \text{ sending activities cannot occur } \neg(N, m')[t] \text{ for all } t \in T_s \text{ and } m \overset{\alpha_1}{\Rightarrow} m_1, \alpha_1 \in A_1 \text{ such that there exists } m' \in \mathcal{R}(N, m_1) \text{ that enables sending activities } (N, m')[t] \text{ for some } t \in T_s. \text{ Thus, the activities } A_0 \text{ and } A_1 \text{ represent the choice between not executing optional asynchronous collaboration } p_{ac} \text{ and executing asynchronous collaboration } p_{ac}. \text{ It follows from } \epsilon \notin \mathcal{L}(cPN), \text{ that at least one } m_\alpha \text{ always exists such that } M_\alpha \text{ cannot be empty, if } p_{ac} \text{ is optional.} \text{ Let } A'_\alpha = \bigcup_{m_\alpha \in M_\alpha} A_0, m_\alpha \text{ be the union of activities } A_0, m_\alpha \text{ that are computed to satisfy } \gamma(m_\alpha). \text{ Define } T_{p_{ac}, \alpha} = \{ t \in T \mid l(t) \in A'_\alpha \setminus \{ \tau \} \} \cup \{ t_{p_{ac}, \tau} \mid \tau \in A'_\alpha \} \text{ for } t_{p_{ac}, \tau} \notin T \text{ the set of transitions for asynchronous collaboration } p_{ac} \text{ that, if executed, can only result in firing traces without any sending activity. Observe that } T_{p_{ac}, \alpha} = \emptyset \text{ in case of compulsory asynchronous collaboration (iii).}
$M_\triangleright$ is characterized by the following formula: for each $m_\triangleright \in M_\triangleright$ it holds that $m_\triangleright$ is reachable by paths in the coverability graph in which some sending activity $\alpha \in l(T_s)$ has occurred and (i) $\gamma(m_\triangleright)$ holds or (ii) there exists a token generator trace $\sigma \in A^\ast$ with sending activity $\alpha \in \sigma$ in the trace $\alpha \in \sigma$. A token generator trace results in unbounded states of the coverability graph ($\omega$-states) and is characterized by: There exists $m \in B(P)$ such that $|m| > 0$ and $m_\triangleright \overset{\sigma}{\rightarrow} m_\triangleright + m$. If (ii) holds, the transition $t \in T_s$ labelled with the sending activity $l(t) = \alpha$ can fire infinitely often. Define $A_1, m_\triangleright = \{\alpha_1\}$ with activity $\alpha_1 \in \sigma, \alpha \neq \alpha_1$ some activity from the token generator trace that always occurs in the trace, i.e., that is part of the cycle in the coverability graph that leads to the unbounded state. If a sending activity is repeated, the set $M_\triangleright$ cannot be empty and $A_1, m_\triangleright$ is always well-defined in case of (ii) for some sending activity, since $cPN$ does not contain any $\tau$-labelled loops for all $(p_{ac}, T_s, T_r) \in AC$. Note that the distinction between a choice of repeating sending activity again (i) or infinitely often repeating a sending activity (ii) is exhaustive, as duplicate labels, i.e., $t, t' \in T$ with $l(t) = l(t')$ cannot exist by definition. Let $A_\triangleright = \bigcup_{m_\triangleright \in M_\triangleright} A_1, m_\triangleright$ be the union of activities $A_1, m_\triangleright$ that are computed to satisfy $\gamma(m_\triangleright)$. These activities signify that sending activities are executed again. Let $T_{p_{ac}, \triangleright} = \{t \in T | l(t) \in A_\triangleright\}$ be the set of transitions for asynchronous collaboration $p_{ac}$ that, if executed, result in firing traces with repeated sending activities. Observe that $T_{p_{ac}, \triangleright} = \emptyset$ in case of no loops involving sending activities.

Since $T_{p_{ac}, \triangleright}$ may not yet include transitions that indicate that sending activities are not sent again in a loop, we add them as follows. Let $A_0, m_\triangleright$ for $m_\triangleright \in M_\triangleright$ be the sets of activities that were computed to satisfy $\gamma(m_\triangleright)$, i.e., an activity $\alpha \in A_0, m_\triangleright$ indicates that after some sending activity has occurred, it will not occur again. Next, $A_\times = A'_\times \cup \bigcup_{m_\triangleright \in M_\triangleright} A_0, m_\triangleright$. Observe that the choice of optional asynchronous collaboration and repeating asynchronous collaboration $p_{ac}$ coincides if $A_\times = A'_\times$. We similarly extend the respective transitions: $T_{p_{ac}, \times} = \{t \in T | l(t) \in A_\times \setminus \{\tau\}\} \cup \{t_{p_{ac}, \tau} | \tau \in A_\times\}$ for $t_{p_{ac}, \tau} \not\in T$.

![Fig. 4. WF-net $N_{p_{ac}}$ constructed for asynchronous collaboration $(p_{ac}, T_s, T_r) \in AC.$](image)
For each \((p_{ac}, T_s, T_r) \in AC\), construct WF-net \(N_{p_{ac}} = (P_{p_{ac}}, T_{p_{ac}}, \Sigma_{p_{ac}}, l_{p_{ac}}, \Lambda_r)\) as depicted in Fig. 4. The construction copies all transitions \(t \in T_s \cup T_r \cup T_{p_{ac}, x} \cup T_{p_{ac}, c}\) as depicted by dashed ovals in Fig. 4. A new transition \(i_{p_{ac}}\) connects a new source \(i\) labelled transitions \(\tau\) place such that the new sink place \(o_{p_{ac}}\) (green highlighted in Fig. 4) and conditional transitions \(T_{p_{ac}, c}\) to repeat sending of transitions. If \(T_{p_{ac}, c} \neq \emptyset\) (dashed, black oval in Fig. 4), we connect \(p_{ac}\) as defined by \(T_s\) and \(T_r\) by means of the new flow relation \(F_{p_{ac}}\) in \(N_{p_{ac}}\). Another new place \(p_{ac, c} \notin P\) is the postset of sending transitions and the preset of both a conditional new transition \(t_{o,p_{ac}}\) that connects to the new sink place \(o_{p_{ac}}\) (green highlighted in Fig. 4) and conditional transitions \(T_{p_{ac}, c}\) (blue highlighted “repeat” oval in Fig. 4) to repeat sending of transitions. If \(T_{p_{ac}, c} = \emptyset\), no repeating takes place such that the \(\tau\)-labelled new transition \(t_{o,p_{ac}}\) connects \(p_{ac, c}\) with \(o_{p_{ac}}\).

If \(T_{p_{ac}, c} \neq \emptyset\) and the infinite repeating behavior (ii) with unbounded states does not occur for any \(m_{c} \in M_{c}\) of \(p_{ac}\), \(\tau\)-labelled new transition \(t_{o,p_{ac}}\) is replaced by arcs to \(T_{p_{ac}, c}\) and transitions \(T_{p_{ac}, c}\). If \(T_{p_{ac}, c} \neq \emptyset\) and the infinite repeating behavior (ii) with unbounded states does occur for some \(m_{c} \in M_{c}\) of \(p_{ac}\), then the respective sending activities \(T_{\infty} \subseteq T_{s}\) and their corresponding repeating activities \(T_{\infty} \subseteq T_{p_{ac}, c}\) are additionally connected through the new place \(p_{ac, \infty} \notin P\) (purple highlighted in Fig. 4). Place \(p_{ac, \infty}\) is only added to \(P_{p_{ac}}\), if sending activities fire infinitely often (cf. ii for \(m_{c}\)). Hence, we ensure that sending activities \(T_{\infty}\) in \(N_{p_{ac}}\) can fire infinitely often by adding a “token generator” structure to the WF-net, i.e., \(T_{\infty}\) has not only the place \(p_{ac}\), but also the place \(p_{ac, \infty}\) as its postset such that through repeating with its corresponding \(T_{\infty}\) it can always fire again. By labelling copied transitions in \(N_{p_{ac}}\) similar to their original counterparts in \(ePN\) and all other new transitions with \(\tau\), we encode executing asynchronous collaboration \((p_{ac}, T_s, T_r)\) in newly constructed WF-net \(N_{p_{ac}}\).

Let \(c_{p_{ac}} : \{1, \ldots, |P_{AC}|\} \rightarrow P_{AC}\) be a concept collection of asynchronous places. Define \(c_{c_{p_{ac}}} = c_{c_{p_{ac}}}, WC_{c_{p_{ac}}} = (N_{c})_{c \in RNG(c_{p_{ac}})}\) and \(CPW_{C_{c_{p_{ac}}} = (\emptyset, \emptyset, \emptyset, \emptyset, ET_{c_{p_{ac}}})\) with \(ET_{c_{p_{ac}}} = \{(t_{c_{p_{ac}}}, T_{c_{p_{ac}}}) \subseteq T_{c_{p_{ac}}} \times P^{\tau}(T_{c_{p_{ac}}}) | t_{c_{p_{ac}}} \in T_{c_{p_{ac}}} \land t_{c_{p_{ac}}} \neq \tau \land \forall t, t' \in T_{c_{p_{ac}}} p_{c_{p_{ac}}}(t) \leq p_{c_{p_{ac}}}(t')\}\) the extended set of equally-labelled transitions in the extended workflow collection \(WC_{c_{p_{ac}}}\). Then, \(ePN_{c_{p_{ac}}} = \bigcup_{c \in RNG(c_{p_{ac}})} N_{c}\). Define relation \(Q \subseteq R(ePN) \times R(ePN)\) such that \((m, m') \in Q\) if \(m(p) \leq m'(p)\) for all places \(p \in P_{ac}\) of \(ePN_{ac}\). Then, \(ePN \equiv Q ePN_{ac}\), since with the exception of new \(\tau\)-labelled transitions in \(N_{p_{ac}}\) for asynchronous collaboration \(p_{ac}\), all transitions of the constructed WF-nets \(N_{p_{ac}}\) are fused with their original counterparts in \(WC\), exactly the same flow relation is encoded in \(p_{ac}\) and \(p_{ac}\) is defined by (4) in Def. 3 and optional and repeating behavior is exactly encoded. By definition only \(T'\) is contained in \(ePN_{ac}\). Since collaboration types \(v_{m}, v_{h}, v_{b}\) are defined by \((p_{ac}, T_s, T_r) \in AC\) and \(\emptyset\) is the only subset of \(\mathcal{T}\) smaller than \(\mathcal{T}'\), it follows from definition in Def. 10 that \(\mathcal{T}'\) is minimal.

In the following, alternative formulation of Theorem 2 we prove the statement by an inefficient construction that copies complete WF-nets of collaboration concepts. Although the alternative formulation does not impose any con-
ditions on the asynchronous collaboration, the size of the constructed $cPN_{os}$ “explodes” quickly.

**Theorem (Synchronous activity execution is minimal - Alternative)**

$\mathcal{Y}' = \{v_s\}$ is minimal.

**Proof.** Let $cPN = ((P, T, F, l, A_r), m_0) = \bigcup_{c \in \text{RNG}(cc)} N_C$ be a collaboration Petri net that contains collaboration types $v \in \mathcal{Y}$ with $CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET)$ a collaboration pattern and $WC = (N_c)_{c \in \text{RNG}(cc)}$ a workflow collection, i.e., $N_c = (P_{ac}, T_{ac}, F_{ac}, l_{ac}, A_{ac})$ is a WF-net for each $c \in \text{RNG}(cc)$. Define for each $p_{ac} \in P_{AC}$, the set of collaboration concepts $C_{p_{ac}} = \{c \in \text{RNG}(cc) | (p_{ac}, T_{ac}, T_{r}) \in AC \land (t \in T_{ac} \lor t \in T_{r}) \land t \in T_{c}\}$ that asynchronously collaborate via $p_{ac}$. We construct a new WF-net $N_{p_{ac}}$ for each $(p_{ac}, T_{ac}, T_{r}) \in AC$ by copying all WF-nets $N_c$ that collaborate via $p_{ac}$. We set $P^p_{RS} = P_{RS} \cap \{p_{ac}\}$. The equally-labelled transitions in $N_{p_{ac}}$ are $ET_{p_{ac}} = \{(t_{ac}, T_{ac}) \subseteq T_{p_{ac}} \times \mathcal{P}^+(T_{ac}) \mid t_{ac} \in T_{ac} \land l^{n}(t_{ac}) \neq \tau \land \forall \tau, t' \in T_{ac} \land l^{n}(t) = l^{n}(t')\}$ with $T_{p_{ac}} = \bigcup_{c \in C_{p_{ac}}} T_{c}$. Then, $CP_{p_{ac}} = (N_{p_{ac}}, P^p_{RS}, ra_{p_{ac}} \cap \{p_{ac}\}, \{p_{ac}, T_{ac}, T_{r}\}, ET_{p_{ac}})$. Lastly, $N_{p_{ac}} = \bigcup_{c \in C_{p_{ac}}} N_c$.

Let $cc_{p_{ac}} : \{1, \ldots, |P_{AC}|\} \rightarrow P_{AC}$ be a concept collection of asynchronous places. Define $cc_{os} = cc_{p_{ac}} \cdot WC_{os} = (N_c)_{c \in \text{RNG}(cc_{os})}$ and $CP_{WC_{os}} = (\emptyset, \emptyset, \emptyset, \emptyset, ET_{os})$ with $ET_{os} = \{(t_{ac}, T_{ac}) \subseteq T_{os} \times \mathcal{P}^+(T_{os}) \mid t_{ac} \in T_{ac} \land l^{n}(t_{ac}) \neq \tau \land \forall \tau, t' \in T_{ac} \land l^{n}(t) = l^{n}(t')\}$ the extended set of equally-labelled transitions in the extended workflow collection $WC_{os}$. Then, $cPN_{os} = \bigcup_{c \in \text{RNG}(cc_{os})} N_C$. Define relation $Q \subseteq R(cPN) \times R(cPN_{os})$ such that $(m, m') \in Q$ iff $m(p) \leq m'(p)$ for all places $p \in P_{os}$ of $cPN_{os}$. Then, $cPN \approx_Q cPN_{os}$, since with the exception of new $\tau$-labelled transitions in $N_{p_{ac}}$, all transitions of the constructed WF-nets $N_{p_{ac}}$ are fused with their original counterparts in $WC$ and exactly the same flow relation is encoded in $p_{ac} \cdot p_{ac}$ as is defined by (4) in [Def. 3]. By definition only $\mathcal{Y}'$ is contained in $cPN_{os}$. Since collaboration types $v_m, v_h, v_k$ are defined by $(p_{ac}, T_{ac}, T_{r}) \in AC$ and $\emptyset$ is the only subset of $T$ smaller than $T'$, it follows from definition in [Def. 10] that $\mathcal{Y}'$ is minimal.

Although $\{v_s\}$ is only minimal under two conditions for an efficient construction (cf. [Theorem 2]), both conditions are not realistic in a real-world collaboration process, as the $\tau$-labelled loop condition excludes arbitrary, non-observable sending of messages or requiring of resources and the empty trace would mean that a collaboration process can execute without any observable activity. From [Theorem 2] we cannot conclude that $\{v_s\}$ is the only minimal element of the partial order $\preceq$. The next theorem proves that synchronous collaboration cannot be simulated by asynchronous collaboration such that $\{v_s\}$ is the only, unique minimal element of $\preceq$.

**Theorem 3 (Synchronous activity execution cannot be simulated).** Let $\mathcal{Y}' = \{v_r, v_h, v_m\}$. Then, $\mathcal{Y}' \not\preceq T$.

**Proof.** We prove by contradiction. Assume $\mathcal{Y}' \preceq T$. Let $cPN = \bigcup_{c \in \text{RNG}(cc)} N_c$ with $CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET)$ be a collaboration Petri net that contains...
all $v \in \mathcal{T}$ and $cPN_{sc} = \bigcup_{c \in RNG(\mathcal{T}_{sc})} N_c$ with $CP_{WC_{sc}} = (P_{AC,sc}, P_{RS,sc}, r_{sc}, AC_{sc}, \emptyset)$ be a collaboration Petri net that contains only $v' \in \mathcal{T}'$ such that $cPN \approx cPN_{sc}$. Observe that all three synchronous collaboration types are represented by $AC$ and do not merge transitions or change their label in Def. 3. Hence, from Def. 10 it follows that all transitions $t \in T_{sc}, (t_{sc}, T_{sc}) \in ET$ are still present in $WC_{sc}$ (changing the workflow collection $WC$ is prohibited) such that $CP_{WC_{sc}} = (P_{AC,sc}, P_{RS,sc}, r_{sc}, AC_{sc}, ET)$ and $\emptyset \neq ET$.

Overall, our theory on collaboration types in collaboration Petri nets demonstrates that modulo weak bisimilarity, asynchronous collaboration can be simulated by synchronous collaboration, but not vice versa. The implications are twofold. First, it suffices to prove weak bisimilarity of discovered Petri nets by some CPD technique discover for $v$, in the next section, if discover constructs similar Petri net patterns for the collaboration types. Second, if a CPD technique discover can only mine asynchronous collaboration types, it is not possible to transform the input $cL'$ or tweak discover in such a way that it can mine synchronous collaborations.

4.3 Bisimulation of $cPN$ and Discovered Petri Nets

This section shows that for any discovered model $M = discover(cL')$ there exists a collaboration Petri net $cPN$ that is weakly bisimilar to $M$, if $Q_{discover}$ holds (cf. Fig. 3). $Q_{discover}$ holds iff (i) the discovered model $M$ is a labelled Petri net $N$ as defined in this paper (cf. Sect. 2), (ii) only supports collaboration types $\mathcal{T}$ with corresponding Petri net patterns that are weakly bisimilar to the typical patterns in Fig. 2 and Def. 5 and (iii) takes the divide-and-conquer approach as depicted in Fig. 3 with some custom model composition $comp$ and some custom mining functionality $cdisc$ to mine asynchronous collaborations (synchronous collaborations are always determined by equal labels). The third condition in $Q_{discover}$ means that discover projects the input log $cL'$ to logs $L_{cc}(1), \ldots, L_{cc}(n)$, applies process discovery $disc$ on each log $L_{cc}(1), \ldots, L_{cc}(n)$, mines some Petri net $N_{ac}$ to represent the asynchronous collaborations, and composes with some custom model composition $comp(N_{cc}(1), \ldots, N_{cc}(n), N_{ac}) = N$.

To illustrate property $Q_{discover}$, the CPD techniques [15] to discover typed Jackson nets and OCPD [8] to discover OCPNs (cf. Tab. 1 and Sect. 5) do not have property $Q_{discover}$, as their discovered models are defined with multiple higher-level net concepts, e.g., token colors, that violate (i). However, OCPD discovers a labelled Petri net $N$ after “step 3” that is a $cPN$ without the global source and sink place. Only in the next steps, $N$ is extended to OCPN with place

\footnote{Although the contradiction follows from a technicality, the statement still holds, if we enable synchronizing arbitrary transitions in a collaboration pattern $CP$ in Def. 3 which is rarely done. To the best of our knowledge, there exists only a single modeling technique [5] that enables synchronizing arbitrary transitions. Nevertheless, asynchronous collaboration does not merge transitions, so we can never avoid firing traces $\sigma$ that include all the labels of synchronized transitions. These labels cannot be simulated by a single, synchronizing label.}
types, token colors and variable arcs. Similarly, their implementation in PM4PY\footnote{https://pm4py.fit.fraunhofer.de/} internally represents OCPN as labelled Petri net $N$ that is extended with the higher-level net concepts for visualization and export only. Besides, for one-to-one correspondences between the collaboration concept’s cases, the higher-level net concepts in an OCPN are not required. Hence, for our collaboration event log $cL$, OCPD has property $Q_{discovery}$, which we demonstrate in the next section.

Two exemplary CPD techniques from Tab.\ 1 that have $Q_{discovery}$ are cross-department collaborative healthcare process (CCHP) discovery\footnote{\[53\]} and the Colliery technique\footnote{\[24\]}. For the sake of brevity, we only report the reasons for CCHP discovery\footnote{\[53\]} and the department collaborative healthcare process\footnote{\[53\]} discovery\footnote{\[53\]} and the department collaborative healthcare process\footnote{\[53\]} discovery\footnote{\[53\]}. For the sake of brevity, we only report the reasons for CCHP discovery\footnote{\[53\]}, (cf. Line 2 of Algorithm 1 in \[53\]). Then, Petri net $N_{ac}$ representing all collaborations between departments discovered by cdisc\textsubscript{CCHP} (cf. Algorithm 2 in \[53\]), is composed with all WF-nets $N_{cc(i)}$ to yield the composed RM\_WF\_nets $N$. Hence, (iii) is satisfied and CCHP has property $Q_{discovery}$.

Note that the construction to create a WF-net for each message channel and resource type $p_{ac} \in P_{AC}$ used to prove that the three asynchronous collaboration types $\nu_m, \nu_h$, and $\nu_r$ are simulated by synchronous activity execution $\nu_s$ (cf. Theorem\ 2) can also be applied to CPD technique discover\ that has property $Q_{discovery}$, in particular (ii). From Theorem\ 2, we can simulate any $N_{ac}$ with asynchronous collaborations by synchronous collaborations, i.e., equally-labelled transitions in the WF-nets $N_{cc(1)}, \ldots, N_{cc(n)}$ and additional WF-nets $N_{p_{ac}}$ for each asynchronous collaboration $p_{ac}$ in $N_{ac}$. Hence, the custom model composition $comp$ in discover reduces to the following model composition:

\begin{enumerate}
\item $P = P^u$ as defined in Def. \ref{def:translate}
\item $T = \bigcup_{cc \in \text{RNG}(cc)} r(T_c)$, with $r$ a renaming function: $r(x) = t_s$ if there exists $T_s \in ST$ such that $x \in T_s$ and $t_s \in T_s$ a fixed transition with $ST = \{ T_s | T_s \subseteq T^u \land \forall t, t' \in T_s, l^u(t) = l^u(t') \land l^u(t) \neq \tau \}$ the set of equally-labelled (synchronous) transition subsets, otherwise $r(x) = x$.
\end{enumerate}
Given collaboration event log $cL$ that contains collaboration concepts in $cc$ and a CPD technique $discover$ with property $Q_{discover}$, we define $discover(cL') = \bigcup_{c \in RNG(cc')}(disc(\pi_c(cL')))_{c \in RNG(ccac)}$ with $cc' = cc \cdot ccac$ for $(N_c)_{c \in RNG(ccac)}$ the workflow collection of WF-nets constructed to simulate asynchronous collaboration by synchronous collaboration. Coming back to the discovery framework in Fig. 3, we are left to prove that $discover(cL') = N \approx cPN$.

**Theorem 4 (Composed models are weakly bisimilar to collaboration Petri nets).** Let $(N, m_0) = \bigcup_{c \in RNG(cc)} N_c$ be a model composition. Then, there exists collaboration Petri net $cPN$ such that $N \approx cPN$.

**Proof.** Given $(N, m_0) = \bigcup_{c \in RNG(cc)} N_c$ with $N = (P, T, F, l, \Lambda, \tau)$ for workflow collection $WC = (N_c)_{c \in RNG(cc)}$. Define collaboration pattern $CP_{WC} = (\emptyset, \emptyset, \emptyset, \emptyset, ET)$ with $ET$ a set that satisfies property (6) in Def. 3 and $|ET| = |ST|$ as defined in Def. 11. From Def. 3 (6), it follows that $ST = \{T_{sc} \mid (t_{sc}, T_{sc}) \in ET\}$. Then, $cPN = (N', m'_0) = \bigcup_{c \in RNG(cc)} N_c$. The statement follows from Def. 4 and from relation $Q \subseteq R(N, m_0) \times R(N', m'_0)$ defined by $(m, m') \in Q$ iff $m(p) \leq m'(p)$ for all places $p \in P$ of the model composition's net $N$.

Consequently, for a CPD technique $discover$ that has property $Q_{discover}$, every discovered Petri net $N = discover(cL')$ is weakly bisimilar to a collaboration Petri net $cPN$. As our proofs are constructive, we are always able to explicitly construct, export or visualize the weakly bisimilar collaboration Petri net such that subsequent analysis techniques in a collaboration mining pipeline can be developed for and applied on collaboration Petri nets. In the next section, we demonstrate how collaboration Petri nets allow to compare CPD techniques.

## 5 Comparative Evaluation

In this section, we present results from comparing CPD techniques with our discovery framework (cf. Fig. 3). There exist at least 14 techniques $discover$ as depicted in Tab. 1. Each technique discovers a specific class of models $N = discover(cL')$. The class of discovered models can either be defined using basic net concepts as introduced in this paper (cf. Sect. 2) or using higher-level net concepts such as token colors, variable arcs, and place types (cf. Sect. 3.2).

From the fourteen CPD techniques in Tab. 1 only four are publicly available such that we can apply the implementation in our publicly available framework implementation. The four publicly available CPD techniques are CCHP [53],

---

7 Concatenation is similarly defined for tuples as for traces.

8 https://github.com/janikbenzin/cpn_discovery
Table 1. Overview of existing CPD techniques discover.

<table>
<thead>
<tr>
<th>CPD</th>
<th>Year</th>
<th>Model</th>
<th>$\mathcal{T}$</th>
<th>Domains</th>
<th>Net Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>[22]</td>
<td>2013/15</td>
<td>Artifact-centric models</td>
<td>$\upsilon_a$</td>
<td>Accounting</td>
<td>Basic</td>
</tr>
<tr>
<td>[8]</td>
<td>2020</td>
<td>Object-centric Petri nets</td>
<td>$\upsilon_a$</td>
<td>Commerce</td>
<td>Higher</td>
</tr>
<tr>
<td>[15]</td>
<td>2023</td>
<td>Typed Jackson nets</td>
<td>$\upsilon_a$</td>
<td>Commerce</td>
<td>Higher</td>
</tr>
<tr>
<td>[25]</td>
<td>2008</td>
<td>WF-nets</td>
<td>$\upsilon_m$</td>
<td>Web service</td>
<td>Basic</td>
</tr>
<tr>
<td>[69]</td>
<td>2019</td>
<td>Communication nets</td>
<td>$\upsilon_m$</td>
<td>Web service</td>
<td>Basic</td>
</tr>
<tr>
<td>[22]</td>
<td>2022</td>
<td>System net</td>
<td>$\upsilon_r$, $\upsilon_m$, $\upsilon_h$</td>
<td>Retail</td>
<td>Higher</td>
</tr>
<tr>
<td>[59]</td>
<td>2023</td>
<td>Generalized WF-nets</td>
<td>$\upsilon_r$, $\upsilon_m$</td>
<td>Multi-agent systems</td>
<td>Basic</td>
</tr>
<tr>
<td>[72]</td>
<td>2023</td>
<td>Multi-agent system net</td>
<td>$\upsilon_h$</td>
<td>Healthcare &amp; other</td>
<td>Basic</td>
</tr>
<tr>
<td>[77]</td>
<td>2013</td>
<td>Integrated RMWF nets</td>
<td>$\upsilon_m$, $\upsilon_s$, $\upsilon_v$</td>
<td>Logistics, Healthcare</td>
<td>Basic</td>
</tr>
<tr>
<td>[65]</td>
<td>2020</td>
<td>Top-level process model</td>
<td>$\upsilon_m$</td>
<td>Logistics</td>
<td>Basic</td>
</tr>
<tr>
<td>[24]</td>
<td>2022</td>
<td>BPMN collab. diagram</td>
<td>$\upsilon_m$</td>
<td>Healthcare &amp; other</td>
<td>Basic</td>
</tr>
<tr>
<td>[50]</td>
<td>2022</td>
<td>Industry net</td>
<td>$\upsilon_m$</td>
<td>Theoretical</td>
<td>Basic</td>
</tr>
<tr>
<td>[53]</td>
<td>2023</td>
<td>Composed RMWF nets</td>
<td>$\mathcal{T}$</td>
<td>Healthcare</td>
<td>Basic</td>
</tr>
</tbody>
</table>

Colliery [24], OCPD [8], and Agent Miner [72]. Although OCPD generally uses higher-level net concepts, for our collaboration event logs $cL$ (cf. Def. 7) OCPD discovers models $N$ that are weakly bisimilar to collaboration Petri nets in an intermediate step of the technique as explained in more detail in Sect. 4.3. Taking the intermediate representation for OCPD into account, all of the four CPD techniques have the property $Q_{\text{discover}}$ and use the Inductive Miner for disc.

We evaluate in a controlled setting similar to [74] in which the true $cPN$ that generates $cL$ is known. As the generation of various collaboration Petri nets $cPN$ is out of scope, an existing dataset consisting of multiple $cPN$ and $cL$ is selected. Selection criteria are a diverse set of collaboration types and collaboration patterns while maintaining a majority of the four CPD techniques to be applicable. The dataset in [58] covers $\upsilon_m$ and $\upsilon_s$ in twelve collaboration Petri nets each with a distinct collaboration pattern. The following datasets are not selected: [53], [75], and logs from the Business Process Intelligence Challenge as organized by the IEEE Task Force on Process Mining do not feature a true collaboration Petri net, [24] only covers $\upsilon_m$, and public OCEL logs only cover $\upsilon_s$. Our selected dataset means that Agent Miner is inapplicable, as Agent Miner can only discover $\upsilon_h$ that is a special case of $\upsilon_m$ (cf. Sect. 3.1).

Our selected dataset models twelve multi-agent systems consisting of two or three agents ($|\mathcal{cc}| = 2$ or $|\mathcal{cc}| = 3$ in Tab. 2) whose collaboration patterns represent various (service) interaction patterns as specified in [16] and described in [59]. Collaboration patterns vary along the number of message transmissions via channel $p_{ac}$ (trans in Tab. 2), the number of message exchanges classified into one-way, two-way or multiple exchanges (# col.), and the relation one-to-one (denoted by 1:1), one-to-many (1:n), and many-to-many (m:n) between sending and receiving activities of the agents via channel $p_{ac}$ (cf. act. rel.). For each $cPN$, the dataset contains a collaboration event log $cL$ with $|cL| = 5000$ traces generated from $cPN$. Further metrics of $cL$ are reported in Tab. 2.

---

For each $cL$, we apply convert resulting in a collaboration event log $cL'$ that is in the format required by the respective CPD technique. Note that we can “apply” the simulation of collaboration type $\upsilon_m$ by $\upsilon_s$ also during conversion to $cL'$, i.e., the OCEL $cL'$ for OCPD includes message object types such that $\upsilon_m$ is simulated by $\upsilon_s$ on the log-level (cf. Sect. 4.1), but in a version that does not properly simulate optional and repeated asynchronous collaboration. The implementation of a construction that completely mirrors Theorem 2 on the log-level is left to future work (cf. future work in Sect. 7). For CCHP and Colliery simulation of types is not required, as they discover $\upsilon_m$ (cf. Fig. 2). Due to Theorem 3 the reverse simulation is not possible, i.e., Colliery cannot discover $\upsilon_s$ such that fitness is reduced.

Table 2. Comparing CPD techniques CCHP [53], Colliery [24], and OCPD [8] with our discovery framework to the true $cPN$ that generated $cL$ [58,59].

<table>
<thead>
<tr>
<th>Col. event log $cL$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events</td>
<td>95052</td>
<td>149988</td>
<td>92668</td>
<td>102404</td>
<td>182452</td>
<td>123322</td>
<td>88068</td>
<td>157098</td>
<td>115000</td>
<td>102548</td>
<td>160000</td>
<td>88089</td>
</tr>
<tr>
<td>Col. types $\upsilon$</td>
<td>$\upsilon_m$</td>
<td>$\upsilon_m$</td>
<td>$\upsilon_m$</td>
<td>$\upsilon_m$</td>
<td>$\upsilon_m$</td>
<td>$\upsilon_m$</td>
<td>$\upsilon_m$</td>
<td>$\upsilon_m$</td>
<td>$\upsilon_m$</td>
<td>$\upsilon_m$</td>
<td>$\upsilon_m$</td>
<td>$\upsilon_m$</td>
</tr>
<tr>
<td>$\upsilon_m$: act. rel.</td>
<td>1:n</td>
<td>m:n</td>
<td>1:1</td>
<td>m:n</td>
<td>1:n</td>
<td>m:n</td>
<td>m:n</td>
<td>m:n</td>
<td>m:n</td>
<td>m:n</td>
<td>m:n</td>
<td>m:n</td>
</tr>
<tr>
<td>Min. trace</td>
<td>17</td>
<td>29</td>
<td>17</td>
<td>18</td>
<td>36</td>
<td>24</td>
<td>8</td>
<td>30</td>
<td>23</td>
<td>20</td>
<td>32</td>
<td>17</td>
</tr>
<tr>
<td>Avg. trace</td>
<td>19</td>
<td>30</td>
<td>19</td>
<td>20</td>
<td>36</td>
<td>25</td>
<td>18</td>
<td>31</td>
<td>23</td>
<td>21</td>
<td>32</td>
<td>18</td>
</tr>
<tr>
<td>Max. trace</td>
<td>21</td>
<td>31</td>
<td>20</td>
<td>23</td>
<td>37</td>
<td>25</td>
<td>29</td>
<td>32</td>
<td>23</td>
<td>21</td>
<td>32</td>
<td>18</td>
</tr>
<tr>
<td>Fitness</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>True Fitness</td>
<td>0.716</td>
<td>0.401</td>
<td>0.754</td>
<td>0.759</td>
<td>0.39</td>
<td>0.564</td>
<td>0.817</td>
<td>0.481</td>
<td>0.714</td>
<td>0.793</td>
<td>0.495</td>
<td>0.766</td>
</tr>
<tr>
<td>Sound</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CCHP Fitness</td>
<td>ex</td>
<td>ex</td>
<td>1.0</td>
<td>ex</td>
<td>ex</td>
<td>ex</td>
<td>ex</td>
<td>ex</td>
<td>ex</td>
<td>ex</td>
<td>ex</td>
<td>0.384</td>
</tr>
<tr>
<td>CCHP Precision</td>
<td>ex</td>
<td>ex</td>
<td>0.765</td>
<td>ex</td>
<td>ex</td>
<td>ex</td>
<td>ex</td>
<td>ex</td>
<td>ex</td>
<td>ex</td>
<td>0.583</td>
<td></td>
</tr>
<tr>
<td>CCHP Sound</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Colliery Fitness</td>
<td>0.863</td>
<td>0.966</td>
<td>1.0</td>
<td>0.683</td>
<td>0.965</td>
<td>0.926</td>
<td>0.587</td>
<td>0.699</td>
<td>0.64</td>
<td>0.711</td>
<td>0.93</td>
<td>0.893</td>
</tr>
<tr>
<td>Colliery Precision</td>
<td>0.722</td>
<td>0.426</td>
<td>0.765</td>
<td>0.776</td>
<td>0.26</td>
<td>0.597</td>
<td>0.687</td>
<td>0.306</td>
<td>0.5</td>
<td>0.669</td>
<td>0.468</td>
<td>0.697</td>
</tr>
<tr>
<td>Colliery Sound</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>OCPD Fitness</td>
<td>1.0</td>
<td>1.0</td>
<td>ex</td>
<td>1.0</td>
<td>1.0</td>
<td>ex</td>
<td>0.727</td>
<td>ex</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.818</td>
</tr>
<tr>
<td>OCPD Precision</td>
<td>0.876</td>
<td>0.351</td>
<td>ex</td>
<td>0.792</td>
<td>0.156</td>
<td>ex</td>
<td>0.956</td>
<td>ex</td>
<td>0.892</td>
<td>0.919</td>
<td>0.431</td>
<td>0.887</td>
</tr>
<tr>
<td>OCPD Sound</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

To evaluate $\text{discover}(cL') = N$, we construct $cPN'$ with $N \approx cPN'$ (cf. Sect. 4.3) and apply alignment-based fitness [4] and alignment-based precision [12] as implemented in PM4PY. Although [63] propose monotone alternatives to both alignment-based metrics, their computation with Entropia on $cPN'$ resulted in > 16 GB heap space and execution times > 3 hours (more than [13] allowed for their benchmark study), so we chose the more efficient alignment-based metrics. With PM4PY’s soundness verification, we also report its results. Additionally, we compute fitness, precision and soundness for the true collaboration Petri net $cPN$ from the dataset (True in Tab. 2).

Table 2. Comparing CPD techniques CCHP [53], Colliery [24], and OCPD [8] with our discovery framework to the true $cPN$ that generated $cL$ [58,59].

To evaluate $\text{discover}(cL') = N$, we construct $cPN'$ with $N \approx cPN'$ (cf. Sect. 4.3) and apply alignment-based fitness [4] and alignment-based precision [12] as implemented in PM4PY. Although [63] propose monotone alternatives to both alignment-based metrics, their computation with Entropia on $cPN'$ resulted in > 16 GB heap space and execution times > 3 hours (more than [13] allowed for their benchmark study), so we chose the more efficient alignment-based metrics. With PM4PY’s soundness verification, we also report its results. Additionally, we compute fitness, precision and soundness for the true collaboration Petri net $cPN$ from the dataset (True in Tab. 2).

https://github.com/jbpt/codebase/tree/master/jbpt-pm/entropia
As CCHP assumes a unique message channel \( p_{ac} \) per sending/receiving activity pair (1:1) and adds each channel with their respective arcs to the discovered model \( cPN' \), their construction often results in a \( cPN' \) with \([o] \notin R(cPN')\), if \( \text{act. rel. of } v_m \) is 1:n or m:n. Without \([o] \notin R(cPN')\), alignments fail, which we report as \( \text{ex} \) in Tab. 2. Although Colliery assumes 1:1 for \( \text{act. rel.} \), it picks one of the unique pairs and adds a message channel for this pair only, i.e., the construction always results in \([o] \in R(cPN')\) for the twelve \( cL \), but with a reduced precision. Additionally, Colliery does not discover a sound model given any of the twelve logs \( cL \). However, four of the twelve true models are not sound.

OCPD is the only CPD technique that discovers some sound models. Three times, OCPD discovers \( cPN' \) with \([o] \notin R(cPN')\), because there is a choice between sending message via two channels \( p_{ac} \) and \( p'_{ac} \). Our currently implemented construction to simulate asynchronous collaboration on the log-level does not allow exclusive channels, i.e., two optional channels \( (p_{ac}, T_s, T_r), (p'_{ac}, T'_s, T'_r) \in AC \) and no trace \( o \in cL \) with \( m_0 \rightarrow [o] \) in the true \( cPN \) exists such that \( \exists i, j \in \{1, \ldots, |o|\} \) \( \sigma(i) \in l^o(T_s) \land \sigma(j) \in l^o(T'_s) \). As the implementation is missing the skipping of optional asynchronous collaboration, the collaboration Petri net of OCPD never reaches marking \([o] \) in case of exclusive channels (cf. future work in Sect. 7). Hence, no alignment is possible. If there are no exclusive channels, OCPD often discovers the most fitting and the most precise models. Surprisingly, for one of the unsound true \( cPN \), OCPD discovers a sound model that is not only as fitting as the true model, but even more precise.

To sum up, CCHP is sensitive to the relationship between sending and receiving activities per message channel. Colliery is the most robust technique, as it always discovers a relatively fitting and precise model. OCPD is less robust than Colliery due to exclusive channels, but discovers the most fitting models.

### 6 Related Work

First, [17] gives a recent overview and analysis of existing Petri net classes to model collaboration processes that are alternatives to collaboration Petri nets. [17] concludes that existing work in modeling of collaboration processes spans many research areas (cf. Sect. 3.1) and is focused on a specific domain or type of collaboration. Furthermore, multiple Petri net extensions are used such that existing techniques are not applicable anymore, while no increased modeling power is achieved (cf. [77,76,53]). An important distinction that holds for any Petri net class is given in Sect. 5 and depicted for the models of CPD techniques in Tab. 1. Basic vs. higher-level net concepts. Many classes with higher-level net concepts exist [35,64], but for many collaboration processes these concepts are not required (cf. Sect. 3.2). Collaboration Petri nets resemble characteristics of the class in [76] as it is the first class with all four collaboration types and of the class in [51] (cf. Sect. 3.2). Both classes are integrated into a single class and extended with respect to their respective advantageous properties such as collaboration types and conciseness. Also, a consistent labelling with silent ac-
tions is introduced to formalize activities similar to [8] and to state a Petri net’s semantics in an activity-centric manner.

Second, CPD techniques as depicted in Tab. 1 are to the best of our knowledge not compared yet, because a standard with a theoretical foundation that enables comparability was missing. Typically, CPD techniques compare with process discovery techniques applied on the collaborating case [77,72,53] or do not compare with any other discovery technique, but only via varying the technique-specific parameters [69,24]. Third, process discovery techniques are extensively compared in [27,23,20,13]. [13] is the most recent benchmark and does not need to develop a theory to standardize Petri net classes to model process orchestrations, as WF-nets were studied before and have emerged as the de-facto standard already. In that regard, we study collaboration Petri net with the aim of bringing CPD research to the same standardization as process discovery research already experiences. Fourth, collaboration event logs can be generated from other generating models such as web service compositions [32,31], inter-organization business processes [33], multi-agent systems [47,57], communicating resource systems [69], and process choreographies [18]. Despite the heterogeneity, we generate collaboration event logs given a WF-net, i.e., the collaboration Petri net, and the collaboration labelling. Hence, in the context of our representational bias (cf. Sect. 3.2), it is possible to apply existing event log generators for WF-nets [26,21] with minor adjustments on the labelling.

7 Conclusion and Outlook

In this paper, we presented collaboration Petri nets as a standard Petri net class for modeling and discovery of collaboration processes along with formal and empirical evidence to show that collaboration Petri nets meet the standard’s requirements in a balanced way. Modeling power is sufficient to represent most collaboration processes that feature collaboration types Υ and occur in the domains healthcare, web services, logistics, and multi-agent systems. While collaboration processes with 1:n or m:n relationships between its collaboration concepts cases can only be represented with extensions, collaboration Petri nets with their WF-net structure benefit from decidability of (conventional) soundness, recursive applicability to deal with various granularity levels, and the vast body of knowledge, techniques, and tools available for WF-nets. Due to the syntactic sugar for the asynchronous collaboration types, the sole reliance on well-known, basic net concepts, and constructive proofs that enable comparability between CPD techniques in our discovery framework, modeling convenience is adequate. If a CPD technique discover has property $Q_{\text{discover}}$, it discovers labelled Petri nets $N$ that are weakly bisimilar to collaboration Petri net. Since at least the three existing CPD techniques CCHP, Colliery, and OCPD have property $Q_{\text{discover}}$, collaboration Petri net possesses desirable relations to existing model classes.

Outlook. Simulating asynchronous collaboration results in collaboration Petri nets and logs that may present technical difficulties for empirical evaluation, so
we will propose a refined construction in our implementation that is not only weakly bisimilar, but preserves reaching the global sink place as is theoretically shown in Theorem 2. A more thorough, comparative maturity analysis of more existing CPD techniques that possess $Q_{\text{discover}}$ with our discovery framework brings collaboration mining closer to the state of process mining. Furthermore, the framework in [15] achieves rediscoverability of typed Jackson nets by projecting onto the subset-closed set of type combinations. Their construction indicates that our framework in Fig. 3 has to be extended to project onto the subset-closed set of collaboration concept combinations to guarantee rediscoverability of collaboration Petri nets. Under what circumstances can we transfer the theory in [15] to collaboration Petri nets remains to be investigated.

References


