## **Process Mining representation using Communication Structured Acyclic Nets (CSA-nets)**

Dr. Nadiyah Almutairi<sup>1,2</sup>, Tuwailaa Alshammari<sup>1</sup> and Mohammed Alahmadi<sup>1,3</sup>

<sup>1</sup>School of Computing, Newcastle University, Science Square, Newcastle upon Tyne, NE4 5TG, United Kingdom
 <sup>2</sup>Department of Science and Technology, University of Hafr Albatin, Hafr Al-Batin 39524, Kingdom of Saudi Arabia
 <sup>3</sup>Department of Software Engineering, University of Jeddah, Jeddah 21493, Kingdom of Saudi Arabia

#### Abstract

Communication Structured Acyclic Nets (CSA-nets) are derived from Structured Occurrence Nets SON which are a Petri net-based formalism used to represent the behaviour of Complex Evolving Systems (CESs). It consists of sets of Acyclic Nets ANs that communicate with each other through a set of buffer places. It supports the analysis and visualisation of activities of CESs. In this work, we use SONs formalism to represent processes that have been decomposed using approaches in [1], we used SONs to compose the subnets and use its robust representation to analyse and visualise the same model, however, without any existing cycles. Our approach provides a mechanism for composing a SON model by using the decomposed subnets. We argue that the resulting model exhibits similar behaviour to the original model.

#### **Keywords**

structured occurrence net, structured acyclic nets, communication structured acyclic net, labeled Petri nets, model decomposition

### 1. Introduction

Process mining is a technique that combines data mining and process analysis to extract valuable insights from event logs generated by information systems. Its aim is to discover, monitor, and improve real processes by extracting knowledge related to these processes from event logs recorded during the execution of business operations. Process mining utilises various techniques, including process discovery, which involves deriving process models from event logs; conformance checking, which compares the actual processes to predefined models; and process enhancement, which improves processes based on insights gained from event logs [2].

Petri nets are one of the popular models for representing processes in process mining. Petri nets can effectively capture concurrency, conflict, and synchronisation aspects of process flows between processes or activities. Specifically, different techniques and algorithms can be used to generate Petri nets from event logs such as the  $\alpha$ -algorithm, state-based regions, and language-based regions.

SONs is generalised in [3] as Communication Structure Acyclic Nets (CSA-nets) which are a Petri net-based formalism used to represent the behaviour of Complex Evolving Systems (CESS). CSA-nets consist of sets of acyclic nets that communicate with each other through a set of buffer places. These buffer places allow both asynchronous and synchronous communication

PNSE'24, International Workshop on Petri Nets and Software Engineering, 2024

<sup>© 0 2024</sup> Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

between acyclic nets, and so can represent the concurrent behaviour of the system as well as synchronisation behaviour. They are suitable tools for modelling and visualising the behaviour of event-based systems.

In this paper, we focus on the outcome of the Petri net decomposition approach presented in [1]. Essentially, we propose constructing a CSA-net model from the decomposed fragments of a process model. That is accomplished by identifying each decomposite *subnet* as an individual *acyclic net* and subsequently applying the proposed rules to establish communication via *buffer places*.

The remainder of this paper is organised as follows: Section 2 summarise some of the existing knowledge and related work. Basic properties of acyclic nets are introduced in the preliminaries Section 3. Section 4 introduces CSA-nets and some of the definitions, while Section 5 discusses the steps taken to compose SON models from decomposed process model. Section 6 concludes our study summarising our findings.

### 2. Related work

Various algorithms utilise Petri nets as representations for process mining. Region-based process discovery techniques and  $\alpha$ -algorithm are examples of such techniques. Basically,  $\alpha$ -algorithm can be used to construct a Petri net process model to describe behaviour from an *event log*, which consists of a set of traces (sequence of events). This is known as process discovery problem. Also, measuring the differences between the observed behaviour (captured by the traces in the event log) and the modelled behaviour (firing sequence of a Peti net) for a given event log and Petri net is known as *conformance checking problem*. In the context of process mining, these two problems are formulated using Petri nets. For instance, [4] developed a regionbased algorithm for discovering Petri nets from event logs, addressing the challenge of deriving accurate process models in process mining. Their approach focuses on constructing a bounded Petri net that encompasses the behaviour observed in event logs, prioritising the generation of models with minimal behaviour while still capturing the essence of the logged events. This method is notable for its ability to produce nets that accurately reflect the observed processes, enhancing the precision of process discovery. The effectiveness of their algorithm is demonstrated through implementation in a tool and validation on various examples, highlighting its potential in improving the reliability and accuracy of process mining outcomes. Where in [5], methods for identifying workflow models are developed, starting with a "workflow log" that records the actual execution of the workflow process. Additionally, a new algorithm is introduced to extract a process model from the log and represent it through a Petri net. The authors demonstrate the capability of the  $\alpha$ -algorithm to effectively mine workflows described by a so-called SWF-net.

An important study in [1], presented a generic approach of decomposing Petri nets for process mining has been introduced to address the challenges of complexity and scalability in analysing large process models or event logs. In another words, a decomposition approach for process mining problems was proposed, in particular, *process discovery* and *conformance checking*. Basically, a process model is decomposed into smaller subnets and the conformance is preformed to each subnet locally to reduce the computation time.

Generally, a large body of work exists in the literature about Petri nets decomposition and

composition. For instance, the authors in [6] present a method for increasing productivity in system development through the use of Petri nets class input-output-place-transition (IOPT) for modelling. They introduce a technique for partitioning Petri net models into concurrent sub-models for distributed implementation, extending the model's capabilities with directed synchronous communication channels. This approach aims to propose a net-splitting operation that allows centralised models to be transformed into distributed models suitable for heterogeneous platforms. The operation includes specific rules for handling structural complexities and ensures behavioural consistency through node duplication. The approach has been validated through case studies demonstrating its applicability in co-design techniques. Moreover, [7] presents a compositional approach for a new extension of ordinary Petri nets called open Petri nets where the places are distinguished as *open (input* or *output)* to represent the interaction with the environment. In their approach, two independent open nets which they agree on a *common* subnet can be glued together to compose a new net. More precisely, a composition operation *pushout* is introduced over open nets such that the common part of the component nets is fused in the composed net. Their approach enables the systematic construction and analysis of large and complex systems by composing smaller nets and understanding their interactions. However, only the semantics of deterministic processes are considered. An extended version of the compositional approach in [7] was introduced in [8]. Hence, non-deterministic processes semantic is investigated in the compositional approach. Ranked open nets were defined (an extension of open nets introduced in [7]) where the maximum number of allowed input and output connections are specified. The composition operation is extended as well to non-deterministic processes. Modal I/O-Petri Nets (MIOPNs) is introduced in [9] for asynchronous composition via communication channels. More precisely, the interfaces are defined by labelled Petri nets which distinguished input and output labels. In their composition approach, individuals components are connected by channels for asynchronous communication. That is components with shared input and output actions are connected together with a new place called communication channel. Generally, the paper investigates how MIOPNs can be utilised to model asynchronous interactions between components via compositional methodology.

Despite the fact that there exists an extensive literature on the approaches of net composition and decomposition, yet difficulties dealing with large event log still an issue. Hence, [1] proposes innovative methods for process discovery decomposition and conformance checking decomposition. In process discovery problem, the approach involves dividing the set of activities into partly overlapping activity sets, discovering model fragments for each activity set, and then combining these to form a complete process model. We utilised this approach to represent acyclic nets from the decomposed sets (nets) identified.

## 3. Preliminaries

#### 3.1. Acyclic Petri Nets

This section provides basic concepts related to acyclic Petri nets and event logs. Acyclic nets can represent alternative ways of interpreting what has happened, and so may exhibit (backward and forward) non-determinism.

An *acyclic net* is a triple *acnet* = (P, T, F), where *P* and *T* are disjoint finite sets of *places* and *transitions* respectively, and  $F \subseteq (P \times T) \cup (T \times P)$  is a *flow relation* such that: (i) *P* is nonempty and *F* is acyclic; and (ii) for every  $t \in T$ , there is  $p \in P$  such that pFt. Graphically, places are represented by circles, transitions by boxes, and arcs between the nodes represent the flow relation If needed, we denote *P* by *P*<sub>*acnet*</sub>, etc. For all  $x \in P \cup T$ ,  $\bullet x = \text{pre}_{acnet}(x) \triangleq \{z \mid zFx\}$  and  $x^{\bullet} = \text{post}_{acnet}(x) \triangleq \{z \mid xFz\}$  (and similarly for sets of nodes  $\bullet X = \bigcup_{x \in X} \bullet x$  and  $X^{\bullet} = \bigcup_{x \in X} x^{\bullet}$ ). Moreover,  $P_{acnet}^{init} = \{p \in P \mid \bullet p = \emptyset\}$ ,  $P_{acnet}^{fin} = \{p \in P \mid p^{\bullet} = \emptyset\}$  are the *initial places* and *final places* respectively.

Given an acyclic net, its execution proceeds by the occurrence (or firing) of sets of transitions. Let acnet = (P, T, F) be an acyclic net.

- markings(*acnet*)  $\triangleq \{M \mid \subseteq P\}$  are the *markings* (shown by placing tokens within the circles), and  $M_{acnet}^{init} \triangleq P_{acnet}^{init}$  is the default *initial* marking, and  $M_{acnet}^{fin} \triangleq P_{acnet}^{fin}$  is the final marking.
- steps $(acnet) \triangleq \{U \subseteq T \mid U \neq \emptyset \land \forall t \neq u \in U : \bullet t \cap \bullet u = \emptyset\}$  are the *steps*.
- enabled<sub>*acnet*</sub> $(M) \triangleq \{U \in \text{steps}(acnet) \mid {}^{\bullet}U \subseteq M\}$  are the steps *enabled* at a marking M, and a step U enabled at M can be *executed* yielding a new marking  $M' \triangleq (M \cup U^{\bullet}) \setminus {}^{\bullet}U$ . This is denoted by  $M[U\rangle_{acnet}M'$ .

Let  $M_0, \ldots, M_k$   $(k \ge 0)$  be markings and  $U_1, \ldots, U_k$  be steps of an acyclic net *acnet* such that  $M_{i-1}[U_i]_{acnet}M_i$ , for every  $1 \le i \le k$ .

- $\sigma = U_1 \dots U_k$  is a step sequence from  $M_0$  to  $M_k$ denote by  $M_0[\sigma]_{acnet}M_k$ , and  $M_0[\rangle_{acnet}M_k$  denotes that  $M_k$  is reachable from  $M_0$ .
- If  $M_0 = M_{acnet}^{init}$ , then  $\sigma \in sseq(acnet)$  is a step sequence, and  $M_k$  is a reachable marking of acnet. Also, if  $\sigma$  cannot be extended further, it is a maximal step sequence in maxsseq(acnet).

A labeled acyclic Perti net ([1]), acnet = (P,T,F,l) is an acyclic Petri net (P,T,F) with labeling function  $T \rightarrow \mathcal{U}_{\mathcal{A}}$  where  $\mathcal{U}_{\mathcal{A}}$  is some universe of activity labels. Let  $\sigma_v = \{a_1, a_2, \dots, a_n\} \in \mathcal{U}_{\mathcal{A}}^*$  be a sequence of activities.  $M_0[\sigma_v \triangleright M_k$  iff there is a step sequence  $\sigma \in \text{sseq}(acnet)$  such that  $M_0[\sigma_{acnet}M_k \text{ and } l(\sigma) = \sigma_v$ . Also,  $dom(l) \subseteq T$  is the domain of l and the range of l denoted by rng(l) is the set of the corresponding observable activities.

For each transition  $t \in T$ , t is called *invisible* if it is without a label,  $t \notin dom(l)$ , however, it is *visible* when  $l(t) \in dom(l)$ . Note that occurrence of visible transition t corresponds to observable activity l(t) [1].

**Example 1.** Figure 1 shows a labelled acyclic Petri net (generated using *ProM*, a process mining framework after modifying the event log used in [1]). The set of places are  $P = \{start, p_1, \ldots, p_9, end\}$ , and the transitions are  $T = \{t_1, t_2, \ldots, t_{11}\}$ .  $t_1$  is enabled at the initial marking  $M_{init} = \{start\}$ , so  $\{start\}[t_1\rangle$ . Hence,  $t_1$  can be fired and produce new marking  $M' = \{p_1, p_2\}$  denoted by  $\{start\}[t_1\rangle\{p_1, p_2\}$ . One of the maximal step sequences is  $\sigma = \{t_1, t_3, t_4, t_5, t_{10}\}$ . The labelling function is  $dom(l) = \{t_1, t_3, t_4, t_5, t_8, t_9, t_{10}\}$  (visible transitions). with  $l(t_1) = a$ ,  $l(t_3) = b$ ,  $l(t_4) = c$ ,  $l(t_5) = d$ ,  $l(t_8) = f$ ,  $l(t_9) = g$ ,  $l(t_{10}) = h$  such that:

a = "register request"b = "examine file"c = "check ticket"d = "decide"f = "send acceptance letter"g = " pay compensation"h = "send reject letter".



Figure 1: A labelled acyclic net

Whereas, the invisible transitions are  $t_2, t_7, t_{11}$  which are unlabelled transitions (silent action). Finally, for the sequence of activities  $\sigma_v = \{a, b, c, d, f, g\}$ , then  $\{start\}[\sigma_v \triangleright \{end\}$  because there is a step sequence  $\sigma = \{t_1, t_3, t_4, t_5, t_7, t_8, t_9, t_{11}\}$  where  $l(\sigma) = \sigma_v$ 

Note that a labelled acyclic Petri net *acnet* with maximal step sequences (starting from the initial marking and ending in final marking) is called *system net* and denoted by  $SN = (acnet, M_{init}, M_{fin})$ .

Event log are the key concept in the context of process mining. Basically, any event log, denoted by L, consists of multiset of sequence of events called *traces*. A trace captures the evolution (from start to end) of a *case*, which is a *process instance*, according to the executed *activities*. Note that it is possible for different cases to have the same traces. Hence, one can say that an event log is *multiset* of traces [1].

As indicated earlier, conformance checking and process discovery are related process mining problems. Conformance checking approaches examine to what extend an event log L and a system acyclic net  $SN = (acnet, M_{init}, M_{fin})$  fit together. Basically, it quantifies the consistency between observed behaviour stored in L and the modelled behaviour represented by SN. There are different techniques of conformance checking and exploring them here is out of the scope of this paper.

Performing conformance checking may take long time as a lot of different traces need to be conformed with a process model which might include an exponential number of traces. For example, when an event log contains millions of events, the entire event log must be checked in order to do conformance checking, which is in fact time consuming. To reduce time required for conformance checking, Petri net and event log are decomposed into smaller fragments. Basically, after generating a Petri net process model based on the activities in the event log, decomposing the process model into smaller subnets and performing conformance checking [1]. There are some work in the literature regarding decomposing process mining problems [1, 10, 11, 12, 13].

More precisely, if there is a system net  $SN = (acnet, M_{init}, M_{fin})$ , then it is decomposed into set of subnets  $SN = \{SN^1, SN^2, \dots, SN^n\}$  such that  $SN^i = (acnet^i, M_{init}^i, M_{fin}^i)$  where  $acnet^i = (P^i, T^i, F^i, l^i)[1]$ .

The decomposition approach in [1] is *valid* if all the resulted subnets satisfy the original labelling function. It means that each place belongs to only one subnet and each invisible transition also belongs to only one subnet. Hence, only visible transitions can appear in different subnets. However, if there are different visible transitions with the same label, then they must belong to one subnet. Finally, each edge appears only in one subnet.

**Example 2.** Figure 3.1 shows the decomposition of the net in Figure 1 based on the decomposition approach in [1]. Basically, the net in Figure 1 is decomposed into six subnets,  $SN = \{SN^1, SN^2, \dots, SN^6\}$ . Their approach of the decomposition produces a maximal partitioning of the overall process model. That means each subnet is as small as possible. Also, note that only the visible transitions with unique labels appear in multiple subnets. Hence, their decomposition approach is not only about the partitioning the edges, but it considered the transitions labels.



**Figure 2:** The decomposition for the net in Figure 1 based on the decomposition approach defined in [1].

## 4. Communication Structured Acyclic Nets (csA-nets)

Communication Structured Acyclic Nets (CSA-nets) are derived from Structured Occurrence Nets (SON) which are a Petri net-based formalism and introduced in [14, 15] and elaborated in [16]. CSA-nets comprise of multiple acyclic nets that are linked through unique elements referred to

as *buffer places*. Buffer places are capable of modelling both asynchronous and synchronous communication.

SON can be used as a framework for visualising and analysing behaviour of Complex Evolving Systems (CESs). For example, in [17], SON were used to support the analysis of evidence related to the activities of complex systems, including cybercrime and major accidents. The goal was to indicate those responsible for cybercrime or identify the causes of accident. Also, SON were used in [18] as a framework for modelling cybercrime investigation. Other work on SON are related to events extraction [19, 20].

A recently designed and implemented tool SONCraft [21] (based on the WORKCraft platform [22, 23] which is a flexible common underpinning for graph based models) provides an extensive and powerful support for dealing with models of complex evolving systems based on SON, including visualisation and verification. Generalising SON as CSA-nets, [24] verifies its behavioural properties using SAT solver. Moreover, improving its visualisation aspects was discussed in [25, 26, 27]. Equipping CSA-nets with probabilities was introduced in [28].

Figure 3 depicts a simple CSA-net, which comprises two acyclic nets:  $acnet_1$  and  $acnet_2$ . Asynchronous communication between the two acyclic nets is represented by arrows, as seen between the transitions c and a in different acyclic nets linked through a single buffer place  $q_1$ . Thus, transition a will never be executed before c in any execution sequence. A synchronous communication is represented by arrows pointing in both directions, using buffer places  $q_2$  and  $q_3$ , as observed between transitions b and d. Thus, these two transitions have to be executed simultaneously.

**Definition 1 (CSA-net).** A communication structured acyclic net (or CSA-net) is a tuple

 $csan = (acnet_1, \dots, acnet_n, Q, W) \quad (n > 1)$ 

such that:

 acnet<sub>1</sub>,...,acnet<sub>n</sub> are acyclic nets with disjoint sets of nodes (i.e., places and transitions). We also denote:

- 2. *Q* is a finite set of buffer places disjoint from  $P_{csan} \cup T_{csan}$ .
- 3.  $W \subseteq (Q \times T_{csan}) \cup (T_{csan} \times Q)$  is a set of arcs.
- 4. For every buffer place q:
  (i) there is at least one transition t such that tWq; and
  (ii) if tWq and qWu then transitions t and u belong to different component acyclic nets. ◊

That is, in addition to requiring the disjointness of the component acyclic nets and the buffer places, it is also required that buffer places pass tokens between different acyclic nets. To indicate relationships between different nodes, for all  $x \in P_{csan} \cup T_{csan} \cup Q_{csan}$  and  $X \subseteq P_{csan} \cup T_{csan} \cup Q_{csan}$ , we denote the *directly preceding* and *directly following* nodes as follows:



Figure 3: Communication structured acyclic net csan.

#### 4.1. Step sequence semantics

**Definition 2 (step and marking [3]).** Let csan be a CSA-net.

- 1. markings $(csan) \triangleq 2^{P_{csan} \cup Q_{csan}}$  are the markings.
- 2.  $M_{csan}^{init} \triangleq P_{csan}^{init}$  is the default initial marking.
- 3. steps(*csan*)  $\triangleq \{U \in 2^{T_{csan}} \setminus \{\emptyset\} \mid \forall t \neq u \in U : \operatorname{pre}_{csan}(t) \cap \operatorname{pre}_{csan}(u) = \emptyset\}$  are the steps.

The initial marking of a CSA-net is the union of the initial markings of all the component acyclic nets. More precisely,  $M_{csan}^{init} = M_{acnet_1}^{init} \cup M_{acnet_2}^{init} \cdots \cup M_{acnet_n}^{init}$  with an assumption that the buffer places are always excluded from the initial marking  $M_{csan}^{init}$ .

#### **Definition 3 (enabled and executed step [3]).** Let M be a marking of csan.

- 1. enabled<sub>*csan*</sub> $(M) \triangleq \{U \in \text{steps}(csan) \mid \text{pre}_{csan}(U) \subseteq M \cup (\text{post}_{csan}(U) \cap Q)\}$  are the steps enabled at M.
- 2. A step U enabled at M can be executed yielding a new marking  $M' \triangleq (M \cup \text{post}_{csan}(U)) \setminus \text{pre}_{csan}(U)$ . This is denoted by  $M[U\rangle_{csan}M'$ .

**Definition 4 ( step sequence [3]).** Let  $M_0, \ldots, M_k$   $(k \ge 0)$  be markings and  $U_1, \ldots, U_k$  be steps of a CSA-net csan such that  $M_{i-1}[U_i\rangle_{csan}M_i$ , for every  $1 \le i \le k$ .

- 1.  $\sigma = U_1 \dots U_k$  is a step sequence from  $M_0$  to  $M_k$ , it is denoted by  $M_0[\sigma]_{csan}M_k$ . Moreover,  $M_0[\rangle_{csan}M_k$  denotes that  $M_k$  is reachable from  $M_0$ .
- 2.  $\operatorname{sseq}(\operatorname{csan}) \triangleq \{ \sigma \mid M_{\operatorname{acnet}}^{\operatorname{init}}[\sigma\rangle_{\operatorname{csan}}M \} \text{ are the step sequences }$
- 3. maxsseq(*csan*)  $\triangleq$  { $\sigma \in$  sseq(*csan*) |  $\neg \exists U : \sigma U \in$  sseq(*csan*)} *are the* maximal step sequences.
- 4. reachable $(csan) \triangleq \{M \mid M_{csan}^{init} \mid \rangle_{csan} M\}$  are the reachable markings.
- 5. finreachable(*csan*)  $\triangleq \{M \mid \exists \sigma \in \maxseq(csan) : M_{csan}^{init}[\sigma\rangle_{csan}M\}$  are the final reachable markings.

 $\diamond$ 

If k = 0 then the corresponding step sequence  $\sigma$  is the *empty* sequence denoted by  $\lambda$ .

# 5. Composing a csA-net model from a decomposed process model

Composing a CSA-net model from the decomposed subnets is based on the technique of Petri net model decomposition with distributed execution. This technique can tackle the difficulties involved in the analysis and modelling of large-scale systems. That is because complex systems can be decomposed into interconnected subsystems that can be analysed and executed independently. [29] suggested establishing communication between decomposed sub-models. Basically, they proposed an approach of composing a system model using the decomposed components. *Net Splitting* operation was defined as a new Petri net operation. Using this operation, a Petri net is split into several subnets. Then, directed synchronous communication channels are added to represent the communication between the resulted components. Also, a Petri net model is partitioned into disjoint sub-models using *Split* operation. These sub-models are associated with component with independent execution. Then, the whole system model is composed by modelling of communication channels among these components [30]. In the same vein, we would like to reuse the decomposed subnets to compose a CSA-net model.

Next we introduce our approach of constructing a CSA-net model which is based on the result of decomposing a system net defined in [1]. Basically, composing a CSA-net model requires identify the overlapped *visible* transitions between the subnets and link them using asynchronous communication and buffer places. Recall that the decomposition approach in [1] allows only visible transitions to be shared between the subnets. These overlapped visible transitions are always the *last* and *first* nodes of any two subnets. Basically, for two subnets  $SN^i$  and  $SN^j$  where i < j and their sets of transitions are  $T^i$  and  $T^j$  respectively, if  $t \in T^i \cap T^j$  then it is the first node of  $SN^j$  and the last node of  $SN^i$ . Hence, for any two subnets  $SN^i$ ,  $SN^j$ , let *IntersectTran* be the set of these overlapped visible transitions. Also, let  $T^i = \{t_1^i, \ldots, t_m^i\}$  be the set of transitions of the subnet  $SN^i$ . Similarly, let  $T^j = \{t_1^j, \ldots, t_r^j\}$  be the set of transitions of the subnet  $SN^j$ . Then,  $t_m^i = t_1^j \in IntersectTran$ . Then, to compose a CSA-net model from the decomposed subnets, an asynchronous communication between the subnets using their overlapped visible transitions is added. In this case, a buffer place q is added such that  $(t_m^i, q)$ ,  $(q, (t^{\bullet j})^{\bullet}) \in W$ . Adding asynchronous communication and the buffer place q represents the global causality between the subnets  $SN^i$  and  $SN^j$ .

The steps of composing a CSA-net model from the decomposed subnets are as follows: Let SN be a system net and the set  $\{SN^1, \ldots, SN^n\}$  be its maximal decomposed subnets. Then its composed CSA-net model can be constructed as follows:

- 1. for each subnet  $SN^i$ , an initial marking is added where  $1 < i \le n$ . Let the set of all new initial markings be  $SN(M_{init})$
- 2. for each subnet  $SN^i$ , a final marking is added where  $1 \le i < n$ . Let the set of all new added final markings be  $SN(M_{fin})$

3. for each two subnets  $SN^i$ ,  $SN^j$  where  $1 \le i < j \le n$  and  $T^i = \{t_1^i, \ldots, t_m^i\}, T^j = \{t_1^j, \ldots, t_r^j\}$ are their set of transitions respectively. Let  $IntersecTran = T^i \cap T^j$  be the set of their overlapped visible transitions. Then, a buffer place q is added such that  $(t_m^i, q), (q, (t_1^{\bullet j})^{\bullet}) \in$ W where  $t_m^i = t_1^j \in IntersecTran$ . Finally,  $t_1^j \in T^j$  is removed from  $T^j$ .

In order to composing a CSA-net model from the decomposed subnets, we assume that each subnet  $SN^i$  is an acyclic net *acnet*.



**Figure 4:** Composing two subnets  $SN_1$  and  $SN_3$  in Figure 2.

**Example 3.** Figure 4 shows an example of our approach. Subnets  $SN^1$  and  $SN^3$  in Figure 2 both have visible transition  $t_1, l(t_1) = a$ . The presence of  $t_1$  in  $SN^3$  captured the local causality between  $t_1$  and other transitions of  $SN^3$ . This local causality can be represented globally at the level of subnets. In this case, the buffer place  $q_1$  and asynchronous communication are added between  $SN^1$  and  $SN^3$  such that  $(t_1, q_1)$  and  $(q_1, t_4)$ . This asynchronous communication replaces the local causality between  $t_1$  and  $t_4$  in  $SN^3$ . Hence,  $t_1$  is removed from  $SN^3$ , however its impact is implicitly captured by the occurrence of  $t_1$  in  $SN^1$  and captured by buffer place  $q_1$  and its arcs. As in CSA-net each acyclic net is independent and it has its own initial and final places, therefore initial and final markings are added for each subnet. In particular, for subnet  $SN^1$  only final marking is added, however, subnet  $SN^3$  initial marking and final marking are added, which are the gray shaded nodes as depicted in Figure 4

Therefore, we would like to redefine Definition 1 to be based on the decomposition approach defined in [1].

**Definition 5** (CCSA-net). Let SN be a system net, and its maximal decomposition is the set of the subnets  $\{SN_1, \ldots, SN_n\}$  as defined in [1]. A composed communication structured acyclic net (or CCSA-net) is a tuple

$$ccsan = (SN^1, \dots, SN^n, Q, W) \qquad (n > 1)$$

such that:

1.  $SN_1, \ldots, SN_n$  are subnets. We also denote:

$$P_{csan} = P^{1} \cup \dots \cup P^{n} \cup SN(M_{init}) \cup SN(M_{fin})$$
  

$$T_{csan} = T^{1} \cup \dots \cup T^{n}$$
  

$$F_{csan} = F^{1} \cup \dots \cup F^{n}.$$

- 2. *Q* is a finite set of buffer places defined as in Definition 1.
- 3. for each two subnets  $SN_i$ ,  $SN_j$  where  $1 \le i < j \le n$  and  $T^i = \{t_1^i, \ldots, t_m^i\}, T^j = \{t_1^j, \ldots, t_r^j\}$ are their set of transitions respectively. Let  $IntersecTran = T^i \cap T^j$  be the set of their overlapped transitions which carry the same label. Then, a buffer place q is created such that  $(t_m^i, q), (q, t_1^j)$  where  $t_m^i, t_1^j \in IntersecTran$  and they carry the same label.
- 4.  $W \subseteq (Q \times T_{csan}) \cup (T_{csan} \times Q)$  is a set of arcs.

 $\diamond$ 

**Example 4.** Figure 5 shows the result of the CSA-net model composition. Basically, all the decomposed subnets in Figure 2 are reused with additional buffer places between the subnets connecting them using their overlapped visible transitions. For instance, in Figure 2 the subnets  $SN_1$  and  $SN_2$  both include transition  $t_1$  with *a* label. Note that  $t_1$  is only transition in  $SN^1$ , so it is the last node. However, it is first node in  $SN^2$ . So, buffer place  $q_2$  is added such that  $(t_1,q_2) \cup (q_2,t_2) \cup (q_2,t_3) \in W$ . The rest of the subnets are treated similarly. Worth noticing that in the composed CSA-net model in Figure 5 the initial markings with auxiliary transitions are added for the subnets  $SN_i$  where  $1 < i \le 6$  (which are the gray shaded nodes). Similarly, final markings are added for the subnets  $SN_i$  where  $1 \le i < 6$ .

Worth to notice that the decomposed subnets in Figure 2 have no initial nor final markings. However, initial and final markings are added when composing them together in the composed CSA-net in Figure 5. Hence, they become system nets. More precisely, the new added initial markings are for the subnets  $SN^i$  where  $1 < i \le 6$ . Also, the new added final markings are for the subnets  $SN^i$  where  $1 \le i < 6$ . The initial and final markings for  $SN^1$  and  $SN^6$  are those of the original net in Figure 1 respectively. Hence, even though the subnets  $SN^2, ..., SN^6$  become system nets, non of their visible transitions are fireable, their visible transitions are fireable when the transitions of  $SN^1$  are fired. In fact, by removing all the added initial and final markings, we obtain the decomposed subnets in Figure 2. More important, adding the initial and final markings does not change the original behaviour.

Regarding the behaviour preservation of the original net, we would like to point that the composed *csan* in Definition 5 and the original acyclic net have closely related behaviour. This is formulated in the next result.

**Theorem 5.1.** Let acnet = (P, T, F, l) be a labelled acyclic net and ccsan be as in Definition 5. Then, for every maximal step sequence  $\sigma$  in ccsan,  $\sigma \in \max(ccsan)$  there is a maximal step sequence  $\sigma'$  in acnet,  $\sigma' \in \max(acnet)$  such that  $\sigma \cap T = \sigma'$ .

Our compositional approach and the compositional proposed in [7, 8] are different in several aspects. In [7, 8], identifying a *common subnet* between two independent nets is required to compose them into a new net. As the open places were classified into *open* and *output*, they were



Figure 5: Composing CSA-net model from the decomposed subnets in Figure 2.

used as interfaces between the net and the environment whereas in our compositional approach the transitions are the interfaces. Also, in our case, there is no need to define the common subnets between the components to be able to construct the compositional model. Instead, we connect the subnets which shared the same visible transitions via buffer places. Hence, there is no need to define a composition operation as we utilise the buffer places to connect the subnets and compose the CSA-net which is similar to the composition in [9]. Finally, the proposed compositional technique is introduced in the context of process mining.

Composing a CSA-net using the decomposed subnets allows us to construct a CSA-net from a database(i.e., an event log in this case), which has not been addressed so far. CSA-net in all the previous work [31, 18, 19, 24, 26, 25, 32] was created manually. Our approach provides a mechanism of constructing a CSA-net from an event log after decomposing the associated process model. We believe that CSA-net is a suitable modelling representation for the context of process mining. Due to the structure of CSA-net where each component is independent, employing it for process mining would reduce the time computation for process mining related problems, specially conformance checking and process discovery. In fact, performing conformance checking for a set of small subnets is more efficient compared for a large model [1]. The motivation for process mining originates from the presence of event data. Although event records can accumulate to terabytes in size, performance becomes an increasingly important factor to consider. In order to effectively ensure satisfactory response times , distributed the analysis across a network of computers is the most feasible strategy [12, 10]. Moreover, utilisation CSA-net for the purpose of process mining would not only be useful in terms of performance, but also provide localised diagnostics and improve the visualisation.

## 6. Conclusion

In this work, we proposed a method to construct CSA-net from decomposed subnets. We argue that CSA-net is a suitable representation for process mining analysis. That is because CSA-net allows individually and locally diagnosis for each acyclic net. Hence, conformance checking can be performed efficiently using CSA-net representation. In future, we plan to implement our approach as a SONCraft plug-in. Furthermore, an experiment will be conducted for conformance checking problem after constructing a CSA-net process model and compare the results with some results exist in the literature. Related formal proofs will be discussed as well.

## References

- W. M. P. van der Aalst, Decomposing petri nets for process mining: A generic approach, Distributed Parallel Databases 31 (2013) 471–507. URL: https://doi.org/10.1007/s10619-013-7127-5. doi:10.1007/S10619-013-7127-5.
- [2] W. M. Van Der Aalst, B. F. Van Dongen, Discovering petri nets from event logs, in: Transactions on Petri nets and other models of concurrency vii, Springer, 2013, pp. 372– 422.
- [3] M. Alahmadi, S. Alharbi, T. Alharbi, N. Almutairi, T. Alshammari, A. Bhattacharyya, M. Koutny, B. Li, B. Randell, Structured Acyclic Nets, Technical Report, School of Computing, University of Newcastle upon Tyne, 2023.

- [4] J. Carmona, J. Cortadella, M. Kishinevsky, A region-based algorithm for discovering petri nets from event logs, in: Business Process Management: 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008. Proceedings 6, Springer, 2008, pp. 358–373.
- [5] W. van der Aalst, T. Weijters, L. Maruster, Workflow mining: discovering process models from event logs, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 1128– 1142. doi:10.1109/TKDE.2004.47.
- [6] A. K. H. da Costa, Petri net model decomposition a model based approach supporting distributed execution, Ph.D. thesis, Universidade NOVA de Lisboa (Portugal), 2010.
- [7] P. Baldan, A. Corradini, H. Ehrig, R. Heckel, Compositional semantics for open petri nets based on deterministic processes, Mathematical Structures in Computer Science 15 (2005) 1–35.
- [8] P. Baldan, A. Corradini, H. Ehrig, B. König, Open petri nets: Non-deterministic processes and compositionality, in: H. Ehrig, R. Heckel, G. Rozenberg, G. Taentzer (Eds.), Graph Transformations, 4th International Conference, ICGT 2008, Leicester, United Kingdom, September 7-13, 2008. Proceedings, volume 5214 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 257–273. URL: https://doi.org/10.1007/978-3-540-87405-8\_18. doi:10.1007/978-3-540-87405-8\\_18.
- [9] S. Haddad, R. Hennicker, M. H. Møller, Specification of asynchronous component systems with modal i/o-petri nets, in: M. Abadi, A. Lluch-Lafuente (Eds.), Trustworthy Global Computing - 8th International Symposium, TGC 2013, Buenos Aires, Argentina, August 30-31, 2013, Revised Selected Papers, volume 8358 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 219–234. URL: https://doi.org/10.1007/978-3-319-05119-2\_13. doi:10.1007/978-3-319-05119-2\\_13.
- [10] W. M. P. van der Aalst, Decomposing process mining problems using passages, in: S. Haddad, L. Pomello (Eds.), Application and Theory of Petri Nets - 33rd International Conference, PETRI NETS 2012, Hamburg, Germany, June 25-29, 2012. Proceedings, volume 7347 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 72–91. URL: https://doi.org/10.1007/978-3-642-31131-4\_5. doi:10.1007/978-3-642-31131-4\\_5.
- [11] W. M. P. van der Aalst, Distributed process discovery and conformance checking, in: J. de Lara, A. Zisman (Eds.), Fundamental Approaches to Software Engineering - 15th International Conference, FASE 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings, volume 7212 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 1–25. URL: https://doi.org/10.1007/978-3-642-28872-2\_1. doi:10.1007/978-3-642-28872-2\ 1.
- [12] C. Bratosin, N. Sidorova, W. M. P. van der Aalst, Distributed genetic process mining, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, 18-23 July 2010, IEEE, 2010, pp. 1–8. URL: https://doi.org/10.1109/CEC.2010.5586250. doi:10.1109/CEC.2010.5586250.
- [13] J. Carmona, J. Cortadella, M. Kishinevsky, Divide-and-conquer strategies for process mining, in: U. Dayal, J. Eder, J. Koehler, H. A. Reijers (Eds.), Business Process Management, 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009. Proceedings, volume 5701 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 327–343.

URL: https://doi.org/10.1007/978-3-642-03848-8\_22. doi:10.1007/978-3-642-03848-8\\_22.

- [14] B. Randell, Occurrence nets then and now: The path to structured occurrence nets, in: L. M. Kristensen, L. Petrucci (Eds.), Applications and Theory of Petri Nets 32nd International Conference, PETRI NETS 2011, Newcastle, UK, June 20-24, 2011. Proceedings, volume 6709 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 1–16.
- [15] B. Randell, M. Koutny, Failures: Their definition, modelling and analysis, in: C. B. Jones, Z. Liu, J. Woodcock (Eds.), Theoretical Aspects of Computing ICTAC 2007, 4th International Colloquium, Macau, China, September 26-28, 2007, Proceedings, volume 4711 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 260–274. URL: https://doi.org/10.1007/978-3-540-75292-9\_18. doi:10.1007/978-3-540-75292-9\\_18.
- [16] M. Koutny, B. Randell, Structured occurrence nets: A formalism for aiding system failure prevention and analysis techniques, Fundam. Informaticae 97 (2009) 41–91.
- [17] B. Randell, M. Koutny, Structured Occurrence Nets: Incomplete, contradictory and uncertain failure evidence, Technical Report 1170, School of Computing Science, University of Newcastle upon Tyne, 2009.
- [18] T. Alharbi, M. Koutny, Domain name system (DNS) tunneling detection using structured occurrence nets (sons), in: D. Moldt, E. Kindler, M. Wimmer (Eds.), Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE 2019), volume 2424 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 93–108.
- [19] T. Alshammari, Towards Automatic Extraction of Events for SON Modelling, CEUR Workshop Proceedings 3170 (2022) 188–201.
- [20] T. Alshammari, Integrating nlp and structured occurrence nets for crime modelling: A pattern-based approach, in: M. Köhler-Bussmeier, D. Moldt, H. Rölke (Eds.), Petri Nets and Software Engineering 2023 co-located with the 44rd International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2023), Lisbon, Portugal, June 27th, 2023, volume 3430 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3430/poster2.pdf.
- [21] B. Li, B. Randell, A. Bhattacharyya, T. Alharbi, M. Koutny, Soncraft: A tool for construction, simulation, and analysis of structured occurrence nets, in: 18th International Conference on Application of Concurrency to System Design, ACSD 2018, Bratislava, Slovakia, June 25-29, 2018, IEEE Computer Society, 2018, pp. 70–74.
- [22] Workcraft, https://workcraft.org/, 2018.
- [23] I. Poliakov, V. Khomenko, A. Yakovlev, Workcraft A framework for interpreted graph models, in: G. Franceschinis, K. Wolf (Eds.), Applications and Theory of Petri Nets, 30th International Conference, PETRI NETS 2009, Paris, France, June 22-26, 2009. Proceedings, volume 5606 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 333–342.
- [24] N. Almutairi, M. Koutny, Verification of communication structured acyclic nets using SAT, in: M. Köhler-Bussmeier, E. Kindler, H. Rölke (Eds.), Proceedings of the International Workshop on Petri Nets and Software Engineering 2021 co-located with the 42nd International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2021), Paris, France, June 25th, 2021 (due to COVID-19: virtual conference), volume 2907 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 175–194.

- [25] M. Alahmadi, Master channel places for communication structured acyclic nets, in: M. Köhler-Bussmeier, E. Kindler, H. Rölke (Eds.), Proceedings of the International Workshop on Petri Nets and Software Engineering 2021 co-located with the 42nd International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2021), Paris, France, June 25th, 2021 (due to COVID-19: virtual conference), volume 2907 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 233–240. URL: https://ceur-ws.org/Vol-2907/paper13.pdf.
- [26] M. Alahmadi, Parametrisation of csa-nets, in: M. Köhler-Bussmeier, D. Moldt, H. Rölke (Eds.), Petri Nets and Software Engineering 2022 co-located with the 43rd International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2022), Bergen, Norway, June 20th, 2022, volume 3170 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 215–216. URL: https://ceur-ws.org/Vol-3170/poster3.pdf.
- [27] M. Alahmadi, Parameterised csa-nets, in: M. Köhler-Bussmeier, D. Moldt, H. Rölke (Eds.), Petri Nets and Software Engineering 2023 co-located with the 44rd International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2023), Lisbon, Portugal, June 27th, 2023, volume 3430 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 167–182. URL: https://ceur-ws.org/Vol-3430/paper10.pdf.
- [28] N. Almutairi, Probabilistic communication structured acyclic nets, in: M. Köhler-Bussmeier, D. Moldt, H. Rölke (Eds.), Petri Nets and Software Engineering 2022 co-located with the 43rd International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2022), Bergen, Norway, June 20th, 2022, volume 3170 of CEUR Workshop Proceedings, CEUR-WS.org, 2022, pp. 168–187.
- [29] A. K. H. da Costa, Petri net model decomposition-a model based approach supporting distributed execution, Ph.D. thesis, Universidade NOVA de Lisboa (Portugal), 2010.
- [30] A. Costa, L. Gomes, Partitioning of petri net models amenable for distributed execution, in: 2006 IEEE Conference on Emerging Technologies and Factory Automation, 2006, pp. 1129–1132. doi:10.1109/ETFA.2006.355253.
- [31] B. Li, Visualisation and Analysis of Complex Behaviours using Structured Occurrence Nets, Ph.D. thesis, School of Computing, Newcastle University, 2017.
- [32] S. Alharbi, Hierarchical simulation of timed behaviours of structured occurrence nets, in: M. Köhler-Bussmeier, D. Moldt, H. Rölke (Eds.), Petri Nets and Software Engineering 2023 co-located with the 44rd International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS 2023), Lisbon, Portugal, June 27th, 2023, volume 3430 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 143–165. URL: https://ceur-ws.org/Vol-3430/paper9.pdf.