CEUR-WS.org/Vol-3170/paper10.pdf

Probabilistic Communication Structured Acyclic Nets

Nadiyah Almutairi[1]

[1]*School of Computing, Newcastle University*
Science Square, Newcastle Helix, Newcastle upon Tyne NE4 5TG, U.K.

Abstract
A fundamental concept used in the area of concurrent systems modelling is *conflict*. In this paper, we outline two approaches leading to a conflict-concurrent model where distributed choices are resolved in a way which allows one to carry out probabilistic estimation. In particular, the concept of *cluster-acyclic net* is introduced to transform a net with *confusion* (a specific combination of conflict and concurrency which makes the probabilistic estimation problematic), into another net whose structure is free-choice which facilitates probabilistic estimation. Then the approaches of removing confusion is extended into *Communication Structured Acyclic nets* (CSA-nets) which is so far lacked of probabilistic analysis. CSA-nets are sets of acyclic nets which can communicate by means of synchronous and asynchronous interactions.

Keywords
Petri net, acyclic net, communication structured acyclic net, conflict, confusion, cluster-acyclic net.

1. Introduction

Communication Structured Acyclic Nets (CSA-nets) are derived from Structured Occurrence Nets (SONs), which was first introduced in [1] and elaborated in [2]. CSA-nets consist of multiple acyclic nets that are joined together with the objective of recording information about the behaviour of Complex Evolving Systems (CESs). [3] lists the features of a CES as being composed of wide array of (sub)systems that interact with each other and with its surrounding environment, resulting in a highly complex structure. The structure of CSA-nets is suitable to represent the activities of such systems where the cognitive complexity can be reduced. For instance, [2] introduce the basic formalization of SON to represent complex fault-error-failure chains. Also, [3] shows that SON can be used to visualise and analyse behaviour of CESs and [4] demonstrates its capabilities for modelling cybercrime investigation. Previous work on SONs is related to a framework for provenance [5], timed behaviours [6], and for CSA-net a SAT-based model checking proposed in [7].

One of the potential applications of CSA-nets is complex crime investigation systems. In crime investigation, full information about a specific activity is, in general, not available [8]. Hence (often numerous) alternate scenarios are pursued by investigators in order to clarify the status of a crime or accident. That is because such a system is characterized as being inherent nondeterminism which originates from the lack of the ability to observe the system. To cope with this feature, investigators need to consider all the possible scenarios. However, considering

PNSE'22, International Workshop on Petri Nets and Software Engineering, Bergen, Norway, 2022
n.m.r.almutairi2@newcastle.ac.uk (N. Almutairi)
© 2022 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)

all possible scenarios might result in meaningless conclusion. Thus, reasoning about what is probable along with what is possible is an essential to obtain meaningful conclusion [9]. Therefore, providing effective probabilistic estimation of different scenarios would greatly help crime investigators to find answers to crucial questions concerning the incident.

There are several works combining probabilistic reasoning and Petri nets. For example, in [10] nets were covered by so-called agents and the non-deterministic and probabilistic decisions were resolved on the basis of local information. The choice of the set of agents that were responsible for the resolution of the choices was done independently. In [11], probabilistic models have been explored for a specific variant of Petri nets. In [12], generalized Markov chains were developed as an extension of Petri nets.The paper [13] extended Mazurkiewicz equivalence to probabilistic words, which were defined as probability distribution, to describe the probabilistic net systems.

Providing a probabilistic framework for concurrent models relies on the presence of conflict transitions which can be associated with positive numerical weights, so that the conflict is resolved by taking their weights into consideration. This seems trivial when the structure of nets is restricted to *free-choice*. However, if it is not the case, then the probabilistic analysis for concurrent models requires to take a special case in consideration. A *confusion* arises when conflict and concurrency are overlapped which causes problematic probabilistic estimation. A considerable amount of literature has been published on handling confusion. For example, in [14], a confusion-free net was constructed by a recursive static decomposition of given net. Probabilities were then added to the new non-deterministic net in order to account for the conflict between transitions. Also, confusion was controlled in [15] by attaching an external event with each transition involved in confusion. Then a control sequence is chosen so that the execution of these transitions is controlled. [16] provides algorithms to detect confusion and an approach of avoiding it by enforcing a constraint of supervisory control to ensure that conflict transitions are enabled together so that confusion cannot occur in marking evolution. Time is introduced in [17] to distinguish the proprieties of weighted transitions to resolve the confusion.

A general comment which applies to the above past research is that they are all concerned with standard Petri nets. The key contribution of this paper is to find solutions for nets supporting different kinds of interprocess communication. In this paper we extend the probabilistic analysis into CSA-nets which is so far lacked of such an extension. The confusion phenomena is also extended and an approach of removing it based on the technique proposed by [14] is introduced.

The paper is organised as follow. Section 2 presents acyclic nets and their properties. Cluster-acyclic nets are introduced in Section 3 and it is shown how to remove confusion in such a case. Section 4 presents CSA-nets and their properties. The definition of confusion is extended to CSA-nets and the approach of removing it is discussed in Section 5. We conclude in Section 6.

2. Acyclic Petri nets

Acyclic Petri nets are a well-established and basic model for representing system behaviour. They can be used to model synchronisation and conflict, two fundamental concepts used in the area of concurrent systems. In this section we first introduce acyclic nets and their properties. Probabilistic acyclic nets are introduced after that. In addition, the notion of *confusion* is

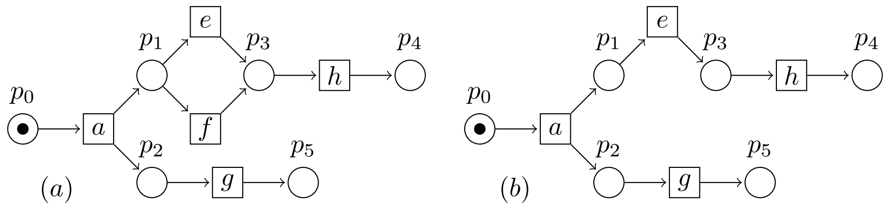


Figure 1: Acyclic net with initial marking (a), and one of its maximal scenarios (b).

discussed. We propose algorithms used for removing it in Section 3.

Basic definitions. 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 (see, e.g., Figure 1). If needed, we denote P by P_{acnet}, etc. For all $x \in P \cup T$, ${}^\bullet x = \text{pre}_{acnet}(x) = \{z \mid zFx\}$ and $x^\bullet = \text{post}_{acnet}(x) = \{z \mid xFz\}$ (and for a set of nodes X, ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$). Moreover, $P^{init}_{acnet} = \{p \in P \mid {}^\bullet p = \varnothing\}$ are the *initial places*, and *acnet* is *backward deterministic* if $|{}^\bullet p| \leq 1$, for every place p.

An *occurrence net* [2] is an acyclic net such that $|{}^\bullet p| \leq 1$ and $|p^\bullet| \leq 1$, for every place p. An occurrence net describes a single execution of some concurrent system in such a way that only the details of causality and concurrency between transitions are captured.

A *scenario* of *acnet* is an occurrence net *ocnet* such that:

- $T_{ocnet} \subseteq T_{acnet}$ and $P_{ocnet} = P^{init}_{acnet} \cup \text{post}_{acnet}(T_{ocnet})$, and
- $\text{pre}_{ocnet}(t) = \text{pre}_{acnet}(t)$ and $\text{post}_{ocnet}(t) = \text{post}_{acnet}(t)$, for every $t \in T_{ocnet}$.

It is *maximal* if there is no scenario *ocnet′* satisfying $T_{ocnet} \subset T_{ocnet'}$. This is denoted by *ocnet* $\in$ scenarios(*acnet*) and *ocnet* $\in$ maxscenarios(*acnet*), respectively. Each scenario is identified by its transitions $V \subseteq T$ and is given by $\text{scenario}_{acnet}(V) = (P, V, F_{acnet}|_{(P \times V) \cup (V \times P)})$, where $P = P^{init}_{acnet} \cup V^\bullet$. Figure 1(b) shows a maximal scenario.

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

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

- $\mu = M_0U_1M_1 \dots M_{k-1}U_kM_k$ is a *mixed step sequence from* M_0 *to* M_k and $\sigma = U_1 \dots U_k$ is a *step sequence from* M_0 *to* M_k.
  We denote $M_0[\mu\rangle\rangle_{acnet} M_k$ and $M_0[\sigma\rangle_{acnet} M_k$, respectively, and $M_0[\rangle_{acnet} M_k$ denotes that M_k is *reachable from* M_0.
- If $M_0 = M^{init}_{acnet}$, then $\mu \in \text{mixsseq}(acnet)$ is a *mixed step sequence*, $\sigma \in \text{sseq}(acnet)$ is a *step sequence*, and M_k is a *reachable* marking of *acnet*. Also, if σ cannot be extended further, it is a *maximal step sequence* in $\text{maxsseq}(acnet)$.

We can treat individual transitions as singleton steps; e.g., a step sequence $\{t\}\{u\}\{w, v\}\{z\}$ can be denoted by $tu\{w, v\}z$.

Well-formedness. An acyclic net *acnet* is *well-formed* if each transition occurs in at least one step sequence and the following hold, for every step sequence $U_1 \dots U_k \in \text{sseq}(acnet)$: (i) $t^\bullet \cap u^\bullet = \varnothing$, for every $1 \leq i \leq k$ and all $t \neq u \in U_i$; and (ii) $U_i^\bullet \cap U_j^\bullet = \varnothing$, for all $1 \leq i < j \leq k$. Intuitively, a well-formed acyclic net does not have 'useless' transitions and no place is filled more than once in any given step sequence. Hence, all its behaviours can be interpreted in terms of causal histories as it is guaranteed that each transition is caused by a unique set of transitions. Each occurrence net is well-formed, and an acyclic net is well-formed iff each transition occurs in at least one scenario, and each step sequence is a step sequence of at least one scenario. Each step sequence σ of a well-formed acyclic net *acnet* induces a scenario $\text{scenario}_{acnet}(\sigma) = \text{scenario}_{acnet}(V)$, where V are the transitions occurring in σ. Thus, different step sequences may generate the same scenario, and conversely, each scenario is generated by at least one step sequence. Moreover, two maximal step sequences generate the same scenario iff their executed transitions are identical.

Conflict, causality and concurrency. Let *acnet* $= (P, T, F)$ be an acyclic net.

- $t \neq u \in T$ are in *direct (forward) conflict*, denoted $t\#_0u$, if ${}^\bullet t \cap {}^\bullet u \neq \varnothing$.
- $\text{conflset}_{acnet}(M, t) = \{t\} \cup \{u \in \text{enabled}_{acnet}(M) \mid t\#_0u\}$ is the *conflict set* of $t \in T$ enabled at a marking M.
- $\text{caused}_{acnet}(x) = \{y \in P \cup T \mid xF^+y\}$ are the nodes *caused* by $x \in P \cup T$.
- $\text{subnet}_{acnet}(V) = ({}^\bullet V \cup V^\bullet, V, F|_{({}^\bullet V \times V) \cup (V \times V^\bullet)})$, for every $V \subseteq T$.

Calculating probabilities in acyclic nets. In order to find probabilities of alternate scenarios, conflicting transitions are assigned *positive numerical weights*, representing the likelihood of transitions. From now on, each acyclic net has an additional (last) component ω defined as a mapping from the set of transitions to positive integers. For each transition t, $\omega(t)$ is its *weight* which in diagrams annotates the corresponding node. Also, we assume that the weights are assigned to the transitions rather than the arcs and are represented inside the boxes. In such cases the names of transitions are represented outside.

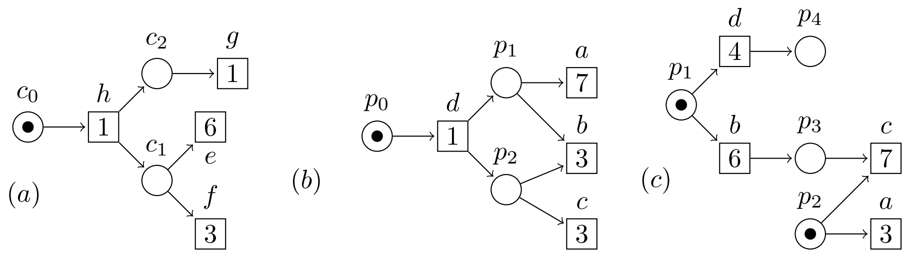


Figure 2: Acyclic nets with weights: backward deterministic (a), with symmetric confusion (b), and with asymmetric confusion (c).

Probabilities of concurrent transitions are given by the products of their weights over the sum of the weights of transitions in their conflict sets. More precisely, we define the probability of executing a transition t and step U enabled at a reachable marking M as:

$$\mathbf{P}_{acnet}(M,t) = \frac{\omega(t)}{\sum_{u\in \mathrm{conflset}_{acnet}(M,t)} \omega(u)} \quad \textit{and} \quad \mathbf{P}_{acnet}(M,U) = \prod_{t\in U} \mathbf{P}_{acnet}(M,t)\ .$$

Then, assuming $M_0[U_1\rangle_{acnet} M_1 \dots M_{k-1}[U_k\rangle_{acnet} M_k$, we define the probability of execution $\sigma = U_1 \dots U_k$ as $\mathbf{P}_{acnet}(\sigma) = \mathbf{P}_{acnet}(M_0,U_1)\cdot \ldots \cdot \mathbf{P}_{acnet}(M_{k-1},U_k)$.

Consider the acyclic net *acnet* in Figure 2(a), where e and f are in direct conflict and have weights 6 and 3, respectively. Such a conflict can be resolved probabilistically at reachable markings $M = \{c_1, c_2\}$ and $M' = \{c_1\}$ by the following calculation: $\mathbf{P}_{acnet}(M,e) = \mathbf{P}_{acnet}(M',e) = \omega(e)/(\omega(e)+\omega(f)) = 6/9$. Note that h and g are *certain transitions* since no transition competes with them for a token, and so their weights are irrelevant.

There are two possible scenarios for this acyclic net: *ocnet*$_1$ = $\mathrm{scenario}(\{h,g,e\})$ and *ocnet*$_2$ = $\mathrm{scenario}(\{h,g,f\})$. Each can be executed by following different maximal step sequences:

- *ocnet*$_1$ has three executions: $\sigma_1 = heg$, $\sigma_2 = hge$, and $\sigma_3 = h\{e,g\}$. Their probabilities are the same as we have: $\mathbf{P}_{acnet}(\sigma_1) = 1\cdot(6/9)\cdot 1 = 6/9$, $\mathbf{P}_{acnet}(\sigma_2) = 1\cdot 1\cdot(6/9) = 6/9$ and $\mathbf{P}_{acnet}(\sigma_3) = 1\cdot((6/9)\cdot 1) = 6/9$.
- *ocnet*$_2$ has also three executions with probabilities equal to $3/9$.

As a result, we can assign probabilities to the two scenarios: $\mathbf{P}_{acnet}(\mathit{ocnet}_1) = 6/9$ and $\mathbf{P}_{acnet}(\mathit{ocnet}_2) = 3/9$. Note that $\mathbf{P}_{acnet}(\mathit{ocnet}_1) + \mathbf{P}_{acnet}(\mathit{ocnet}_2) = 1$. A key issue is to *ensure in each case that no matter which of the executions of a scenario is followed in the calculation of probabilities, the same value is obtained (in other words, all maximal step sequences generated from a scenario should have the same probability).* One can then define the probability of a scenario as $\mathbf{P}_{acnet}(\mathit{ocnet}) = \mathbf{P}_{acnet}(\sigma)$, where σ is any execution of *ocnet*. Also, in a sound probability model, the sum of the probabilities of all possible scenarios should be 1. However, the above is not always the case as acyclic nets can exhibit *confusion* described next.

Confusion. An interplay between conflict and concurrency can lead to a situation called *confusion* which interferes with the calculation of probabilities. In a *symmetric confusion*, executing transition f concurrent with e removes at least one transition from the conflict set of e. In an *asymmetric confusion*, executing transition f concurrent with e adds a new transition to the conflict set of e [18]. Hence, the order in which one chooses to execute e and f may lead to different probabilities.

A well-formed acyclic net *acnet* has a *confusion* [14] at a reachable marking M if there are distinct transitions e, f, h such that $\{e, f\} \in \text{enabled}_{acnet}(M)$ and one of the following holds:

- $e\#_0 h\#_0 f$ and $h \in \text{enabled}_{acnet}(M)$.
- $e\#_0 h$ and $h \in \text{enabled}_{acnet}(M') \setminus \text{enabled}_{acnet}(M)$, where $M[f\rangle_{acnet} M'$.

We then denote $\text{symconfused}_{acnet}(M, e, f, h)$ in the first (*symmetric*) case, and in the second (*asymmetric*) case we denote $\text{asymconfused}_{acnet}(M, e, f, h)$.

Proposition 1. *Let acnet be a well-formed acyclic net and M be its reachable marking. If it is the case that* $\text{symconfused}_{acnet}(M, e, f, h)$ *or* $\text{asymconfused}_{acnet}(M, e, f, h)$ *holds, then we have* $\text{conflset}_{acnet}(M, e) \neq \text{conflset}_{acnet}(M', e)$ *and* $e \in \text{enabled}_{acnet}(M')$*, where* $M[f\rangle_{acnet} M'$.

Figure 2(b) depicts a well-formed acyclic net *acnet* with symmetric confusion such that we have $\text{symconfused}_{acnet}(M, a, c, b)$, where $M = \{p_1, p_2\}$. Consider the scenario $ocnet_1 = \text{scenario}_{acnet}(\{d, a, c\})$ with three executions ($\sigma_1 = dac$, $\sigma_2 = dca$, and $\sigma_3 = d\{a, c\}$). The probability of σ_1 is $\mathbf{P}_{acnet}(\sigma_1) = 1 \cdot 7/10 \cdot 1 = 7/10$. However, if σ_2 is executed, then the resulting probability is $\mathbf{P}_{acnet}(\sigma_2) = 3/6 \neq \mathbf{P}_{acnet}(\sigma_1)$. Hence one cannot assign a probability to $ocnet_1$. A similar conclusion can be reached for the acyclic net in Figure 2(c) which exhibits asymmetric confusion $\text{asymconfused}_{acnet}(M^{init}_{acnet}, a, b, c)$.

A crucial property is that in a confusion-free acyclic net, conflict sets of transitions are constant for all the executions of each scenario.

Proposition 2. *Let acnet be a confusion-free well-formed acyclic net.*

1. *If M and M' are two reachable markings of acnet such that $M[\rangle_{acnet} M'$, then we have* $\text{conflset}_{acnet}(M, t) = \text{conflset}_{acnet}(M', t)$*, for every* $t \in \text{enabled}_{acnet}(M) \cap \text{enabled}_{acnet}(M')$.
2. *If M and M' are two reachable markings of ocnet* $\in \text{scenarios}(acnet)$*, then it is the case that* $\text{conflset}_{acnet}(M, t) = \text{conflset}_{acnet}(M', t)$*, for every* $t \in \text{enabled}_{ocnet}(M) \cap \text{enabled}_{ocnet}(M')$.

Hence, for confusion-free acyclic nets one can calculate probabilities of individual scenarios. Note that there are classes of nets which exclude confusion by imposing structural restrictions, e.g., *free-choice nets* [19].

3. Cluster-acyclic nets

The approach of removing confusion is based on identifying the choice points by applying the concept of *clusters*.

A *cluster* of a well-formed acyclic net *acnet* $= (P, T, F)$ is a non-empty set $\kappa \subseteq T$ such that $\kappa \times \kappa \subseteq (\#_0)^*$ and if ${}^\bullet t \subseteq {}^\bullet\kappa$, then $t \in \kappa$. It is *maximal* if there is no cluster κ' such that $\kappa' \subset \kappa$. Moreover, $\sqsubset$ is a relation on the maximal clusters such that $\kappa \sqsubset \kappa'$ if $\mathrm{caused}(\kappa) \cap {}^\bullet\kappa' \neq \varnothing$. The set of all clusters is denoted by $\mathrm{clusters}(\mathit{acnet})$, and the set of all maximal clusters is denoted by $\mathrm{maxclusters}(\mathit{acnet})$.

A maximal cluster is an equivalence class of the relation $(\#_0)^*$, and so the set of maximal clusters partitions the set of all transitions. Below we will consider a subclass of acyclic nets where the ordering $\sqsubset$ is a strict partial order on the set maximal of clusters.

A well-formed backward deterministic acyclic net is *cluster-acyclic* if the relation $\sqsubset$ on its maximal clusters is a strict partial order. Note that cluster-acyclic nets include all free-choice well-formed backward deterministic acyclic nets as well as all extended free-choice acyclic nets.

A *transaction* of a maximal cluster κ of a well-formed backward deterministic cluster-acyclic net with a marking $M \subseteq {}^\bullet\kappa$ is a step θ included in κ such that ${}^\bullet\theta \subseteq M$. Moreover, it is *maximal* if there is no transaction θ' such that $\theta \subset \theta'$, and we denote $\theta \in \mathrm{maxtrans}(\kappa, M)$. Intuitively, a transaction is one of possible ways of executing the subnet induced by the cluster. *From now on we assume that every cluster-acyclic net is well-formed and backward deterministic.* Also, a cluster $V = \{t_1, \ldots, t_k\}$ can be denoted as $\theta_{t_1 \ldots t_k}$.

Proposition 3. *Let κ be a cluster of a cluster-acyclic net acnet.*

1. $(\kappa \times \kappa) \cap F^+ = \varnothing$.
2. $\mathrm{caused}(\kappa) = \bigcup \mathrm{caused}(\mathrm{trans}(\kappa, {}^\bullet\kappa))$.
3. $\mathrm{maxtrans}(\kappa, {}^\bullet\kappa) \subseteq \mathrm{maxsseq}(\mathrm{subnet}_{\mathit{acnet}}(\kappa))$.
4. $\mathrm{scenario}_{\mathrm{subnet}_{\mathit{acnet}}(\kappa)}(\mathrm{maxtrans}(\kappa, {}^\bullet\kappa)) = \mathrm{maxscenarios}(\mathrm{subnet}_{\mathit{acnet}}(\kappa))$.

Since the transaction of a cluster are maximal step sequences generating all maximal scenarios of the subnet induced by the cluster, one can say that the *behaviour* of the cluster is described by its transactions.

Consider again the confused acyclic net in Figure 2(c). Figure 3 shows its two maximal clusters, $\kappa_1 = \{b, d\}$ and $\kappa_2 = \{a, c\}$, together with their presets. We have $\kappa_1 \sqsubset \kappa_2$ and $\kappa_2 \not\sqsubset \kappa_1$, and so the net is cluster-acyclic. Moreover, we have: $\mathrm{maxtrans}(\kappa_1, \{p_1\}) = \{\{b\}, \{d\}\}$, $\mathrm{maxtrans}(\kappa_2, \{p_2, p_3\}) = \{\{a\}, \{c\}\}$, and $\mathrm{maxtrans}(\kappa_2, \{p_2\}) = \{\{a\}\}$.

3.1. Removing confusion from cluster-acyclic net (Approach A)

In this section, *acnet* $= (P, T, F)$ is a $\underline{\text{fixed}}$ cluster-acyclic net.

The encoding of a cluster-acyclic net *acnet* into an confusion-free acyclic net is based on *negative* places. For a place $p \in P$, its negative image is denoted by $\overline{p}$. Moreover, the original transitions are replaced by transitions representing transactions associated with clusters. We will proceed as follows:

- All places of *acnet* together with their markings are retained. In addition, for each place $p \in P$, an empty place $\overline{p}$ is created.

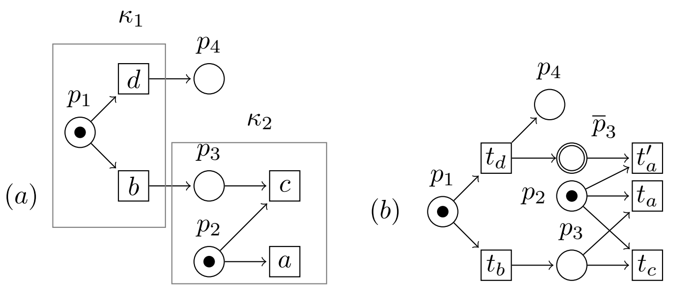


Figure 3: (a) maximal clusters with their presets of the acyclic net in Figure 2(c), (b) encoding the cluster-acyclic net of into a confusion-free acyclic net with negative place, where: $t_b = t_{\kappa_1,\{b\},\{p_1\}}$, $t_d = t_{\kappa_1,\{d\},\{p_1\}}$, $t_a = t_{\kappa_2,\{a\},\{p_2,p_3\}}$, $t_c = t_{\kappa_2,\{c\},\{p_2,p_3\}}$, and $t'_a = t_{\kappa_2,\{a\},\{p_2\}}$.

- All transitions of *acnet* are removed. Instead, for each cluster κ with marking $M \subseteq {}^\bullet\kappa$ and all its transactions θ in $\mathrm{maxtrans}(\kappa, M)$ a new transition is created. Its preset are the places in the preset of κ and the negative versions of all places in (${}^\bullet\kappa \setminus M$). Its postset are all the places in the postset of θ and the negative versions of places caused by κ which are not caused by θ (when such a place $\overline{p}$ is marked, one can be sure that the original place p will never be marked).
- Negative places which have no influence on the behaviour are removed.

The following definition provides full details of the encoding.

Definition 1 (encoding cluster-acyclic net). *The* confusion-free encoding *of a cluster-acyclic net acnet* $= (P, T, F)$ *is an acyclic net* $\mathrm{confree}(acnet) = (P', T', F')$ *constructed in three steps. First, we generate:*

- $P' = P \cup \overline{P}$.
- $T' = \{t_{\kappa,\theta,M} \mid \kappa \in \mathrm{maxclusters}(acnet) \wedge M \subseteq {}^\bullet\kappa \wedge \theta \in \mathrm{maxtrans}(\kappa, M)\}$
- ${}^\bullet t = M \cup (\overline{{}^\bullet\kappa \setminus M})$ *and* $t^\bullet = \theta^\bullet \cup \overline{\mathrm{caused}(\kappa \setminus \theta) \cap P}$, *for every* $t = t_{\kappa,\theta,M} \in T'$.

After that we delete all places $\overline{p} \in \overline{P}$ *such that* ${}^\bullet\overline{p} = \varnothing$ *or* $\overline{p}^\bullet = \varnothing$. ◇

Crucially, for every reachable marking M, if $p \in M$, then $\overline{p} \notin M$. Note that $\mathrm{confree}(acnet)$ contains no negative images of initial nor final places of *acnet*. Moreover, if *acnet* is free-choice, then $\mathrm{confree}(acnet)$ is isomorphic to *acnet*.

Figure 3(b) shows the result of our encoding for the confused cluster-acyclic net in Figure 2(c). Despite the fact that there are two transitions based on cluster κ_2 and transaction $\theta = \{a\}$, their presets are different, as one of them is associated with marking $\{p_2\}$ and the other with $\{p_2, p_3\}$. Note that the two copies of the original transition a, t_a and t'_a, never occur in the same step sequence, as p_3 and $\overline{p}_3$ cannot receive tokens in the same execution.

The behaviour of the constructed confusion-free acyclic net is closely linked to the original one. At first, both t_b and t_d are enabled. If t_b is executed, t_a and t_c become enabled. Executing t_d, on the other hand, produces tokens in places p_6 and $\overline{p}_3$, which enables t'_a. The only behaviour in the original net that is not preserved is the possibility of executing a before d as this execution leads to ambiguous analysis of system behaviour as a is executed before c is enabled. Thus, even though a is initially enabled, its firing is delayed until the decision between b and d is taken. Therefore, the construction steps include delay and additional causality for some transitions similarly as in [14].

Figure 4 shows another example of our translation for a confused cluster-acyclic net. In this case, there are three maximal clusters ($\kappa_1 = \{a, b\}$, $\kappa_2 = \{c, d\}$, and $\kappa_3 = \{x, y, z\}$).

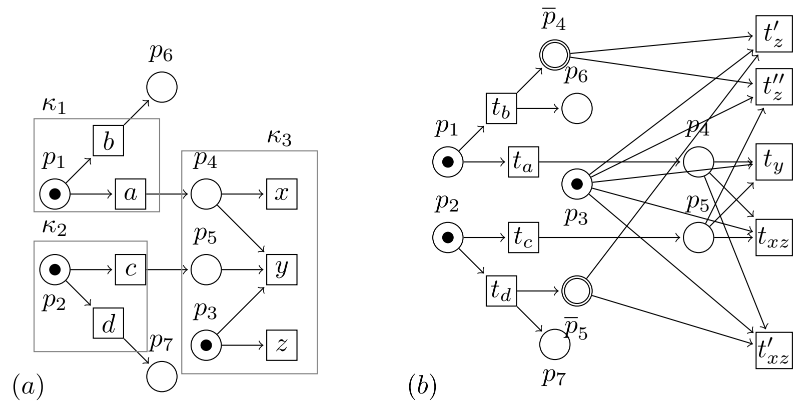


Figure 4: (a) acyclic net with two types of confusion and its encoding to confusion-free acyclic net (b), where: $t_a = t_{\kappa_1,\{a\},\{p_1\}}$, $t_b = t_{\kappa_1,\{b\},\{p_1\}}$, $t_c = t_{\kappa_2,\{c\},\{p_2\}}$, $t_d = t_{\kappa_2,\{d\},\{p_2\}}$, $t_{xz} = t_{\kappa_3,\{x,z\},\{p_3,p_4,p_5\}}$, $t_y = t_{\kappa_3,\{y\},\{p_3,p_4,p_5\}}$, $t'_{xz} = t_{\kappa_3,\{x,z\},\{p_3,p_4,\overline{p_5}\}}$, $t_z = t_{\kappa_3,\{z\},\{p_3,\overline{p_4},\overline{p_5}\}}$.

The acyclic nets constructed above are confusion-free.

Proposition 4. $\mathrm{confree}(\mathit{acnet})$ *is a confusion-free acyclic net, for every cluster-acyclic net acnet.*

The proposition below shows that the behaviour of the constructed confusion-free net is closely linked to the original cluster-acyclic net. To show the this, for every set of transitions U of $\mathrm{confree}(\mathit{acnet})$, we denote $T_U = \bigcup\{\theta \mid t_{\kappa,\theta,\beta} \in U\}$.

Proposition 5. *Let acnet be a cluster-acyclic net.*

1. *For every ocnet* $\in \mathrm{maxscenarios}(\mathit{acnet})$*, there is a maximal step sequence* $U_1 \dots U_k$ *of the net* $\mathrm{confree}(\mathit{acnet})$ *such that* $T_{U_1} \dots T_{U_k}$ *is a maximal step sequence of ocnet.*
2. *If* $U_1 \dots U_k$ *is a step sequence of* $\mathrm{confree}(\mathit{acnet})$*, then* $T_{U_1} \dots T_{U_k}$ *is a step sequence of acnet.*

Compared to the encoding in [14], we feel that the encoding in this paper is simpler because of the following:

- The contact-free net resulting from the encoding uses the same net model as all acyclic nets (i.e., persistent places and extended markings are not needed).

- The number of negative places is smaller than in [14].
- There is no need to use *dynamic nets* as an intermediate step of the construction.
- We expect that our encoding will be much easier to comprehend and use by practitioners with relatively limited formal methods skills.

Additionally, one of the limitations of [14] is that only the case of backward deterministic nets has been addressed. In our work not reported here, we extended the encoding to the unfolding semantics, where more general cases can be considered.

We do not expect the fact that our encoding works for a subclass of cluster-acyclic nets to be limiting in practical applications. This is based on the examples modelling investigations we evaluated, and also on the fact that in case of non-compliance it is always possible to require an investigator to provide additional information to 'repair' the net, or using other source of information, e.g., timing information associated with places and transitions.

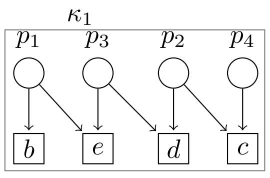
3.2. Removing confusion from cluster-acyclic net (Approach B)

In this section the acyclic net *acnet* is restricted to the class of *binary synchronisation acyclic nets* such that, for every transition $t \in T$, $|{}^\bullet t| \leqslant 2$.

Let *acnet* $= (P, T, F)$ be a well-formed backward deterministic cluster-acyclic net and $\kappa \in \mathrm{clusters}(\mathit{acnet})$. A *border* of κ is a minimal set of non-initial places $\beta \subseteq {}^\bullet\kappa$ such that $({}^\bullet\kappa)^\bullet \setminus \kappa \subseteq \beta^\bullet$. A *transaction* κ is a maximal step θ included in κ. The sets of all borders and transactions of κ are denoted by $\mathrm{borders}(\kappa)$ and $\mathrm{trans}(\kappa)$, respectively.

Definition 2 (encoding binary synchronisation cluster-acyclic net). *The* confusion-free encoding *of a binary synchronisation cluster-acyclic net acnet* $= (P, T, F)$ *is an acyclic net* $\mathrm{confree}'(\mathit{acnet}) = (P', T', F')$ *constructed in three steps. First, we generate:*

- $P' = P \cup \overline{P}$.
- $T' = \{t_{\kappa,\theta,\beta} \mid \kappa \in \mathrm{clusters}(\mathit{acnet}) \wedge \theta \in \mathrm{trans}(\kappa) \wedge \beta \in \mathrm{borders}(\kappa)\}$.



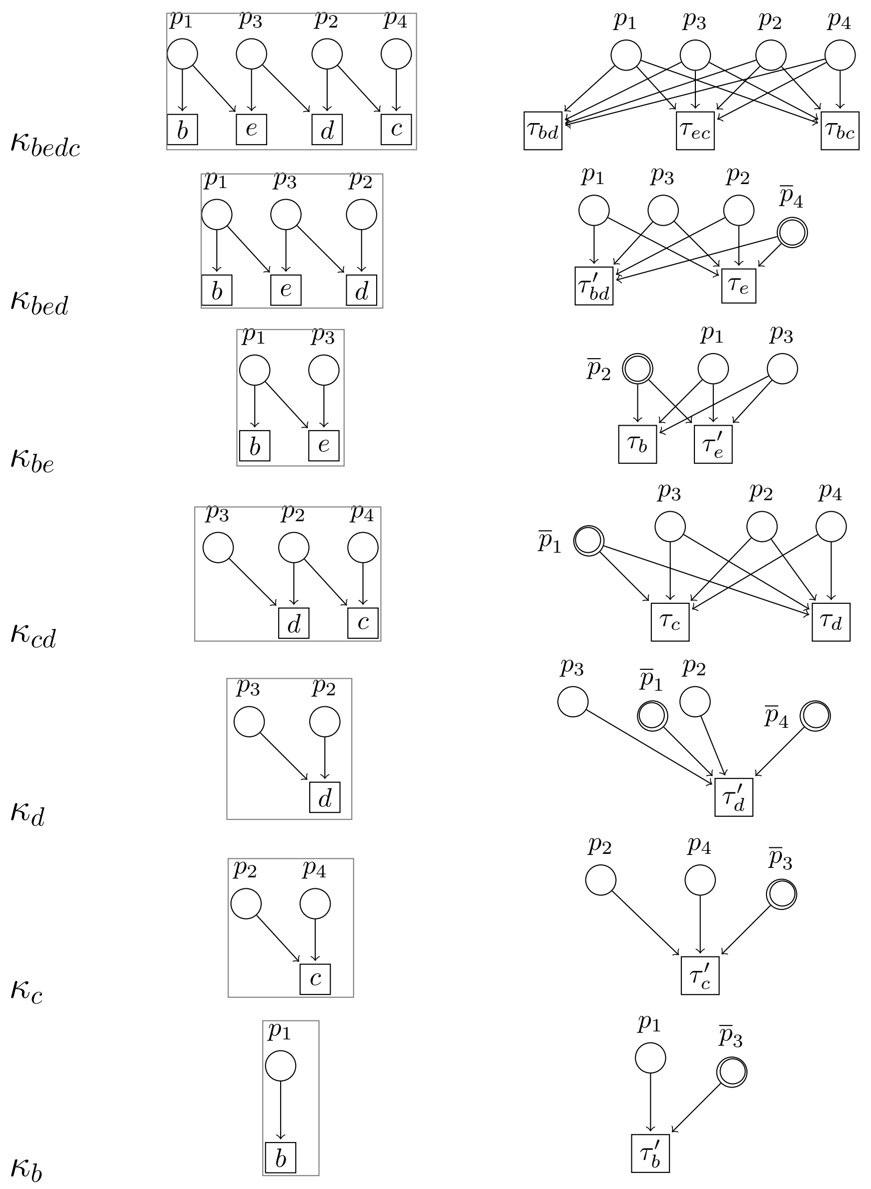
(sub-)cluster κ	$\mathrm{borders}(\kappa)$	$\mathrm{trans}(\kappa)$
$\kappa_{bedc} = \{b, d, e, c\}$	$\varnothing$	$\{\{\theta_{bd}\}, \{\theta_{bc}\}, \{\theta_{ec}\}\}$
$\kappa_b = \{b\}$	$\{\{p_3\}\}$	$\{\{\theta_b\}\}$
$\kappa_{be} = \{b, e\}$	$\{\{p_2\}\}$	$\{\{\theta_b\}, \{\theta_e\}\}$
$\kappa_{bed} = \{b, e, d\}$	$\{\{p_4\}\}$	$\{\{\theta_{bd}\}, \{\theta_e\}\}$
$\kappa_{cd} = \{c, d\}$	$\{\{p_1\}\}$	$\{\{\theta_c\}, \{\theta_d\}\}$
$\kappa_c = \{c\}$	$\{\{p_3\}\}$	$\{\{\theta_c\}\}$
$\kappa_d = \{d\}$	$\{\{p_1, p_4\}\}$	$\{\{\theta_d\}\}$

Figure 5: A maximal cluster κ_{bedc} of a binary synchronisation acyclic net, and the borders and maximal transactions of κ_{bedc} and its sub-clusters (all places are assumed to be non-initial).

- *$^\bullet t = {}^\bullet\kappa \cup \overline{\beta}$ and $t^\bullet = \theta^\bullet \cup \overline{\mathrm{caused}(\kappa \setminus \theta) \cap P}$, for every $t = t_{\kappa,\theta,\beta} \in T'$.*

After that we delete all places $\overline{p} \in \overline{P}$ such that $^\bullet\overline{p} = \varnothing$ or $\overline{p}^\bullet = \varnothing$. Finally, we replace each remaining place $\overline{p} \in \overline{P}$ with $\overline{p}^\bullet = \{t_1, \ldots, t_k\}$ by new places $\overline{p}^{t_1}, \ldots, \overline{p}^{t_k}$ satisfying $^\bullet(\overline{p}^{t_i}) = {}^\bullet\overline{p}$ and $(\overline{p}^{t_i})^\bullet = \{t_i\}$, for every $1 \leq i \leq k$. $\diamond$

Figure 5 shows a maximal cluster κ_{bedc} of some binary synchronisation acyclic net *acnet*



κ_{bedc}
κ_{bed}
κ_{be}
κ_{cd}
κ_d
κ_c
κ_b
p_1 p_3 p_2 p_4
b e d c
τ_{bd} τ_{ec} τ_{bc}
$\overline{p}_4$ τ'_{bd} τ_e
$\overline{p}_2$ τ_b τ'_e
$\overline{p}_1$ τ_c τ_d
τ'_d
$\overline{p}_3$ τ'_c
τ'_b

Figure 6: All the clusters and their associated maximal transactions for the binary synchronised acyclic net in Figure 5, where $\tau_{bd} = t_{\kappa_{bedc},\{b,d\},\varnothing}$, etc.

with symmetric confusion. It also shows its (sub-)clusters and the corresponding borders and maximal transactions.

Figure 6 shows the clusters and their correspond encoded transitions derived from the maximal transactions associated with each cluster. The maximal cluster is $\mathrm{maxclusters}(\kappa_{bedc})$ with its maximal transactions $\mathrm{maxtrans}(\kappa_{bedc}) = \{\{\theta_{bd}\}, \{\theta_{bc}\}, \{\theta_{ec}\}\}$. Since all the pre-places of κ_1 are included together with their output transitions, $\mathrm{borders}(\kappa_1) = \varnothing$. All the sub-clusters with their maximal transactions are shown in Figure 6. Note that $\mathrm{borders}(\kappa_b) = \{p_3\} = \mathrm{borders}(\kappa_c)$ and τ'_b, τ'_c are not in conflict hence, in the encoding $\overline{p}_3$ will be split into $\overline{p}_3$ and $\overline{p}'_3$.

4. Communication structured acyclic nets

Basic definitions. Communication structured acyclic nets add communication to represent the interaction among several separated subsystems [10]. Each communication structured acyclic net is a set of acyclic nets with synchronous or asynchronous communication between their transitions implemented using extra nodes called buffer places (which provided a motivation for a/syn connections discussed, e.g., in [20]). When two transitions are subject to synchronous communication, they are always executed together, but under asynchronous communication they may be executed simultaneously or one of them after the other.

A *communication structured acyclic net (or* CSA*-net)* is a tuple *csan* $= (acnet_1, \dots, acnet_n, Q, W)$ $(n \geq 1)$ such that:

- $acnet_1, \dots, acnet_n$ are well-formed acyclic nets with disjoint sets of nodes. We denote: $X_{csan} = X_{acnet_1} \cup \dots \cup X_{acnet_n}$, for $X \in \{P, T, F, P^{init}\}$.
- Q is a finite set of *buffer places* disjoint from $P_{csan} \cup T_{csan}$ and $W \subseteq (Q \times T_{csan}) \cup (T_{csan} \times Q)$ is a set of arcs. We also denote $Q_{csan} = Q$, $W_{csan} = W$.
- For every $q \in Q$, there is $z \in T$ such that zWq, and if tWq and qWu then t and u belong to different $acnet_i$'s.

For all $x \in P_{csan} \cup T_{csan} \cup Q_{csan}$, we denote $\mathrm{pre}_{csan}(x) = \{z \mid (z, x) \in F_{csan} \cup W_{csan}\}$ and $\mathrm{post}_{csan}(x) = \{z \mid (x, z) \in F_{csan} \cup W_{csan}\}$ (and similarly for sets of nodes).

The *csan* is *backward deterministic* (or BDCSA-net) if the component acyclic nets are backward deterministic and $|\mathrm{pre}_{csan}(q)| = 1$, for every $q \in Q_{csan}$. Moreover, *csan* is a *communication structured occurrence net (or* CSO*-net)* if: (i) the component acyclic nets are occurrence nets; (ii) $|\mathrm{pre}_{csan}(q)| = 1$ and $|\mathrm{post}_{csan}(q)| \leq 1$, for every $q \in Q_{csan}$; and (iii) no place in P_{csan} belongs to a cycle in the graph of *csan*. A CSO-net exhibits backward determinism and forward determinism providing full and unambiguous information about the causal histories of all transitions it involves. Figure 7(b) shows a CSO-net.

A *scenario* of a CSA-net *csan* is a CSO-net *cson* $= (ocnet_1, \dots, ocnet_n, Q', W')$ such that: (i) $ocnet_i \in \mathrm{scenarios}(acnet_i)$, for every $1 \leq i \leq n$; (ii) $Q' \subseteq Q$ and $W' \subseteq W$; and (iii) $\mathrm{pre}_{cson}(t) = \mathrm{pre}_{csan}(t)$ and $\mathrm{post}_{cson}(t) = \mathrm{post}_{csan}(t)$, for every $t \in T_{cson}$. Moreover, *cson* is *maximal* if there is no scenario *cson′* satisfying $T_{cson} \subset T_{cson'}$. We denote this by $ocnet \in \mathrm{scenarios}(csan)$ and $ocnet \in \mathrm{maxscenarios}(csan)$, respectively. Each scenario of a CSA-net *csan* is identified by the set of its transitions V, and denoted by $\mathrm{scenario}_{csan}(V)$. Figure 7(b) shows a maximal scenario for the CSA-net in Figure 7(a).

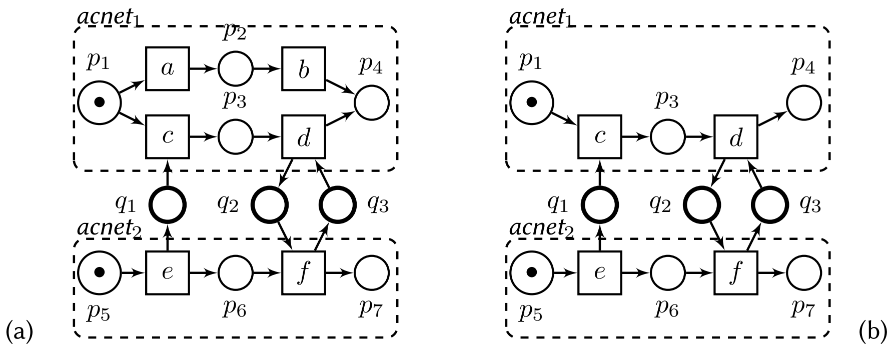


Figure 7: CSA-net with initial marking (a), and CSO-net with initial marking (b).

Step sequence semantics. Let *csan* be a CSA-net.

- $\text{markings}(csan) = \{M \mid M \subseteq P_{csan} \cup Q_{csan}\}$ are the *markings* and $M^{init}_{csan} = P^{init}_{csan}$ is the default *initial* marking.
- $\text{steps}(csan) = \{U \subseteq T_{csan} \mid U \neq \varnothing \wedge \forall t \neq u \in U : \text{pre}_{csan}(t) \cap \text{pre}_{csan}(u) = \varnothing\}$ are the *steps*.
- $\text{enabled}_{csan}(M) = \{U \in \text{steps}(csan) \mid \text{pre}_{csan}(U) \subseteq M \cup (\text{post}_{csan}(U) \cap Q)\}$ are the steps *enabled* at M, and a step U enabled at M can be *executed* yielding a new marking $M' = (M \cup \text{post}_{csan}(U)) \setminus \text{pre}_{csan}(U)$. This is denoted by $M[U\rangle_{csan} M'$.

Let $M_0, \ldots, M_k$ ($k \geq 0$) 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 $1 \leq i \leq k$.

- $\mu = M_0 U_1 M_1 \ldots M_{k-1} U_k M_k$ is a *mixed step sequence from M_0 to M_k* and $\sigma = U_1 \ldots U_k$ is a *step sequence from M_0 to M_k*.
  We denote $M_0[\mu\rangle\rangle_{csan} M_k$ and $M_0[\sigma\rangle_{csan} M_k$, respectively. Moreover, $M_0[\rangle_{csan} M_k$ denotes that M_k is *reachable from M_0*.
- If $M_0 = M^{init}_{csan}$, then $\mu \in \text{mixsseq}(csan)$ is a *mixed step sequence*, $\sigma \in \text{sseq}(csan)$ is a *step sequence*, and M_k is a *reachable marking* of *csan*. Also, if σ cannot be extended further, it is a *maximal step sequence* in $\text{maxsseq}(csan)$.

In contrast to the step sequence of an acyclic net, where a step consists only of enabled transitions, in a CSA-net an enabled step U can involve a/synchronous communications. That is, it can use not only the tokens of the places within acyclic nets, but also tokens in the buffer places generated in the same step [3].

In Figure 7(a), transitions e and c are communicating *asynchronously*, so they can be executed together, or e then c (but not c before e). On the other hand, d and f must be executed simultaneously as they are involved in *synchronous* communication. Therefore, some possible maximal step sequences are $\{a, e\}b$, $a\{b, e\}$, $ec\{d, f\}$, and $\{e, c\}\{d, f\}$.

Well-formed csa-nets. A csa-net *csan* is *well-formed* if each transition occurs in at least one step sequence and the following hold, for every step sequence $U_1 \dots U_k \in \mathrm{sseq}(csan)$: (i) $\mathrm{post}_{csan}(t) \cap \mathrm{post}_{csan}(u) = \varnothing$, for every $1 \leq i \leq k$ and all transitions $t \neq u \in U_i$; and (ii) $\mathrm{post}_{csan}(U_i) \cap \mathrm{post}_{csan}(U_j) = \varnothing$, for all $1 \leq i < j \leq k$.

Intuitively, a well-formed csa-net does not have 'useless' transitions and no place is filled more than once in any given step sequence. Each cso-net is well-formed, and a csa-net is well-formed iff each transition occurs in at least one scenario, and each step sequence is a step sequence of at least one scenario.

Each step sequence σ of a well-formed csa-net *csan* induces a scenario $\mathrm{scenario}_{csan}(\sigma) = \mathrm{scenario}_{csan}(V)$, where V are the transitions occurring in σ. Thus, different step sequences may generate the same scenario, and conversely, each scenario is generated by at least one step sequence. Moreover, two maximal step sequences generate the same scenario iff their executed transitions are identical.

Syn-cycles. In the case of cso-nets each executed step can be unambiguously represented as a disjoint union of sub-steps which cannot be further decomposed.

A *syn-cycle* of a cso-net *cson* is a maximal non-empty set of transitions $S \subseteq T_{csan}$ such that, for all $t \neq u \in S$, $(t, u) \in W^{+}_{cson}$. The set of all syn-cycles is denoted by $\mathrm{syncycles}(cson)$. The idea behind the notion of syn-cycles is to capture fully synchronous communications [10]. In Figure 7(a), there is one non-singleton syn-cycle $S = \{d, f\}$.

The set of all syn-cycles $\mathrm{syncycles}(cson)$ is a partition of the transition set T_{cson}. As stated below, each step occurring in step sequences of a cso-net can be partitioned into syn-cycles (in a unique way).

Proposition 6. *Let M be a reachable marking of a cso-net cson and $M[U\rangle_{cson} M'$. Then there are $U_1, \dots, U_k \in \mathrm{syncycles}(cson)$ such that $U = U_1 \uplus \dots \uplus U_k$ and $M[U_1 \dots U_k\rangle_{cson} M'$.*

This means, for example, that all reachable markings of a cso-net can be generated by executing syn-cycles rather than all the potential steps. Moreover, the same holds for every well-formed csa-net *csan* and the syn-cycles of its scenarios given by $\mathrm{syncycles}(csan) = \bigcup\{\mathrm{syncycles}(cson) \mid cson \in \mathrm{scenarios}(csan)\}$.

Proposition 7. *Let M be a reachable marking of a well-formed cso-net csan and $M[U\rangle_{csan} M'$. Then there are $U_1, \dots, U_k \in \mathrm{syncycles}(csan)$ with $U = U_1 \uplus \dots \uplus U_k$ and $M[U_1 \dots U_k\rangle_{cson} M'$.*

Conflict in csa-net. Let *csan* be a well-formed csa-net.

- Two syn-cycles $S \neq S' \in \mathrm{syncycles}(csan)$ are in *direct forward conflict*, denoted $S \#_0 S'$, if $\mathrm{pre}_{csan}(S) \cap \mathrm{pre}_{csan}(S') \neq \varnothing$.
- The *conflict set* of a syn-cycle $S \in \mathrm{syncycles}(csan)$ enabled at a marking M of *csan* is given as $\mathrm{conflset}_{csan}(M, S) = \{S\} \cup \{S' \in \mathrm{enabled}_{csan}(M) \mid S \#_0 S'\}$.

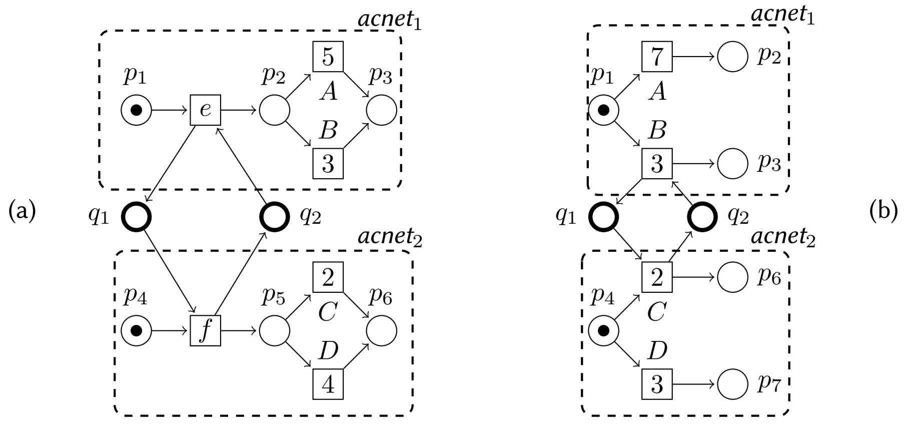


Figure 8: Probabilistic communication structured acyclic nets.

Calculating probabilities in csa-nets. We now extend the notion of calculating probabilities in acyclic nets to well-formed csa-nets. In order to determine probabilities of alternate scenarios, each syn-cycle S is assigned a *positive numerical weight* $\omega(S)$. Below we assume that $\omega(S) = \omega(t_1) + \cdots + \omega(t_k)$, where each $\omega(t_i)$ is the weight of transition t_i as before. We then define the probability of a syn-cycle S enabled at a reachable marking M as the weight of S divided the weight of all the enabled $S' \in \text{syncycles}(\mathit{csan})$ that are in conflict with S enabled at M, and then calculate the probability of a step U composed of syn-cycles $S_1, \ldots, S_k$:

$$\mathbf{P}_{\mathit{csan}}(M,S) \;=\; \frac{\omega(S)}{\sum_{S' \in \text{conflset}_{\mathit{csan}}(M,S)} \omega(S')} \quad \mathbf{P}_{\mathit{csan}}(M, \biguplus_{i=1}^{k} S_i) = \prod_{i=1}^{k} \mathbf{P}_{\mathit{csan}}(M,S_i)\ .$$

Then the probability of $\sigma = U_1 \ldots U_k$ is $\mathbf{P}_{\mathit{csan}}(\sigma) = \mathbf{P}_{\mathit{csan}}(M_0,U_1) \cdot \ldots \cdot \mathbf{P}_{\mathit{csan}}(M_{k-1},U_k)$, where $M_0, \ldots, M_{k-1}$ are such that $M_{i-1}[U_i\rangle_{\mathit{csan}}\, M_i$, for every $1 \leq i \leq k$.

Figure 8 illustrates the notion of csa-nets with weights (the weights are shown inside transitions in conflict). In Figure 8(a), there are two maximal scenarios of *acnet*$_1$ involving either A or B, and also two scenarios for *acnet*$_2$ involving either C or D. Then *csan* has four scenarios: *cson*$_1$ = $\text{scenario}_{\mathit{csan}}(\{e,f,A,C\})$, *cson*$_2$ = $\text{scenario}_{\mathit{csan}}(\{e,f,A,D\})$, *cson*$_3$ = $\text{scenario}_{\mathit{csan}}(\{e,f,B,C\})$, and *cson*$_4$ = $\text{scenario}_{\mathit{csan}}(\{e,f,B,D\})$. There are also five syn-cycles: $S_1 = \{e,f\}$, $S_2 = \{A\}$, $S_3 = \{B\}$, $S_4 = \{C\}$, and $S_5 = \{D\}$.

The probabilities of scenarios are calculated through their maximal step sequences. For example, *cson*$_1$ has three: $\sigma_1 = \{e,f\}AC$, $\sigma_2 = \{e,f\}CA$, and $\sigma_3 = \{e,f\}\{A,C\}$. All with the same probability 10/48, e.g., using σ_1 we get $\mathbf{P}_{\mathit{csan}}(\mathit{cson}_1) = 1 \cdot (5/8) \cdot (2/6) = 10/48$. Also, *cson*$_2$, *cson*$_3$, *cson*$_4$ each has three different executions and their probabilities are equal as well. As a result, one can assign probabilities to the four scenarios. $\mathbf{P}_{\mathit{csan}}(\mathit{cson}_2) = 1 \cdot (3/8) \cdot (2/6) = 6/48$, $\mathbf{P}_{\mathit{csan}}(\mathit{cson}_3) = 1 \cdot (5/8) \cdot (4/6) = 20/48$, and $\mathbf{P}_{\mathit{csan}}(\mathit{cson}_4) = 1 \cdot (3/8) \cdot (4/6) = 12/48$. Note that $\mathbf{P}_{\mathit{csan}}(\mathit{cson}_1) + \mathbf{P}_{\mathit{csan}}(\mathit{cson}_2) + \mathbf{P}_{\mathit{csan}}(\mathit{cson}_3) + \mathbf{P}_{\mathit{csan}}(\mathit{cson}_4) = 1$.

Confusion in probabilistic csa-nets. Consider the csa-net *csan* in Figure 8(b). The two possible scenarios are *cson*$_1$ = $\text{scenario}_{\mathit{csan}}(\{A,D\})$ and *cson*$_2$ = $\text{scenario}_{\mathit{csan}}(\{B,C\})$. Note that the latter scenario is executed in a single execution step due to a synchronous communication that forms the syn-cycle $S_1 = \{B, C\}$. In $\text{scenario}_{csan}(\{A, D\})$ transitions belong to separate syn-cycles as $S_2 = \{A\}$ and $S_3 = \{D\}$, and it has three different maximal step sequences: $\sigma_1 = S_2S_3$, $\sigma_2 = S_3S_2$, and $\sigma_3 = (S_2 \cup S_3)$. If σ_1 is executed, then the probability of S_3 is 1 (D is in this case is a certain transition), and so we get $\mathbf{P}_{csan}(\sigma_1) = (7/10) \cdot 1 = 7/10$. However, if σ_2 is executed, then the resulting probability is $(3/5) \cdot 1 = 3/5$. Hence, one cannot assign probability to $cson_1$. The behaviour of this net is similar to the acyclic net with *symmetric confusion* in Figure 2(b). Here, A and D are independent transitions, but firing any of them has influence on the conflict set of the other. For example, firing A decreases the conflict set of D, which is why confusion arises.

A well-formed CSA-net *csan* has a *confusion* at a reachable marking M if there are distinct syn-cycles $S_1, S_2, S_3 \in \text{syncycles}(csan)$ such that $S_1, S_2, S_1 \uplus S_2 \in \text{enabled}_{csan}(M)$, and one of the following holds:

- $S_1 \#_0 S_3 \#_0 S_2$ and $S_3 \in \text{enabled}_{csan}(M)$.
- $S_1 \#_0 S_3$ and $S_3 \in \text{enabled}_{csan}(M') \setminus \text{enabled}_{csan}(M)$, where $M[S_2\rangle_{csan} M'$.

We then denote $\text{symconfused}_{csan}(M, S_1, S_2, S_3)$ in the first (*symmetric*) case, and in the second (*asymmetric*) case we denote $\text{asymconfused}_{csan}(M, S_1, S_2, S_3)$. If *csan* has no confusion at all the reachable markings, then it is *confusion-free*.

For the CSA-net in Figure 8(b), we have $\text{symconfused}_{csan}(M, S_1, S_2, S_3)$.

Proposition 8. *Let csan be a well-formed* CSA*-net and M be its reachable marking. If it is the cae that* $\text{symconfused}_{csan}(M, S_1, S_2, S_3)$ *or* $\text{asymconfused}_{csan}(M, S_1, S_2, S_3)$*, then we have* $\text{conflset}_{csan}(M, S_1) \neq \text{conflset}_{csan}(M', S_1)$ *and* $S_1 \in \text{enabled}_{csan}(M')$*, where* $M[S_2\rangle_{csan} M'$.

5. Removing confusion from CSA-nets

In this section, we assume that a well-formed CSA-net *csan* has a confusion. To remove the confusion from *csan*, it is encoded into a single acyclic net. If the result is an acyclic well-formed net whose clusters are *acyclic*, the approaches proposed in Section 3.1 is applied. The encoding of *csan* is based on expanding the underlying transitions involved in asynchronous communications. Hence, buffer places are considered as regular places. For the transitions communicate *synchronously*, they are always combined together as one synchronised transition. The buffer places are removed for combined synchronised transitions. The transitions of *csan* are removed. Instead, for each syn-cycle S of *csan*, a transition τ_S is created (graphically, we put the transitions of S inside the box representing τ_S). Its preset and postset are those of syn-cycle S except those involved in synchronous communication. The encoding is done as follows:

- All the places of *csan* together with their markings are retained.
- Each buffer place becomes a regular place.
- For each syn-cycle $S \in \text{syncycles}(csan)$, transition τ_S is created. Its presets are the pre-places of S except the buffer places in $\text{pre}_{csan}(S) \cap \text{post}_{csan}(S)$. Its post-sets are the post-places of S except the buffer places in $\text{pre}_{csan}(S) \cap \text{post}_{csan}(S)$.

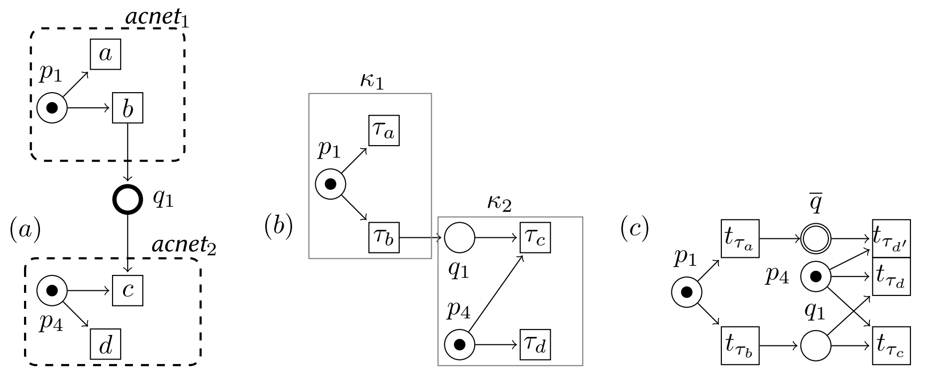


Figure 9: Confused CSA-net (a), its equivalent acyclic net (b), and its confusion-free acyclic net (c).

- The original buffer places with empty pre-sets are removed.

The following definition formally captures the details of the encoding. Let *csan* be a well-formed CSA-net. Then $\mathrm{acyclicnet}(\mathit{csan}) = (P, T, F)$ is constructed so that:

- $P = P_{\mathit{csan}} \cup Q_{\mathit{csan}}$ and $T = \{\tau_S \mid S \in \mathrm{syncycles}(\mathit{csan})\}$,
- $\mathrm{pre}_{\mathit{acnet}}(\tau_S) = \mathrm{pre}_{\mathit{csan}}(S) \setminus Q_{\mathit{csan}} \cap \mathrm{pre}_{\mathit{csan}}(S) \cap \mathrm{post}_{\mathit{csan}}(S)$, for every $\tau_S \in T$, and
- $\mathrm{post}_{\mathit{acnet}}(\tau_S) = \mathrm{post}_{\mathit{csan}}(S) \setminus Q_{\mathit{csan}} \cap \mathrm{pre}_{\mathit{csan}}(S) \cap \mathrm{post}_{\mathit{csan}}(S)$, for every $\tau_S \in T$.

After that, the original buffer places with empty pre-sets are removed and, if $\mathrm{acyclicnet}(\mathit{csan})$ is a well-formed acyclic net whose clusters are acyclic, we apply the construction from Section 2.

Figure 9(a) illustrates the notion of asymmetric confusion in a well-formed CSA-net *csan*. The *csan* is composed of two acyclic nets, *acnet*$_1$ and *acnet*$_2$, with asynchronous communication via buffer place q_1. There are four syn-cycles: $S_1 = \{a\}$, $S_2 = \{b\}$, $S_3 = \{c\}$, and $S_4 = \{d\}$. These syn-cycles exhibit asymmetric confusion, as we have $\mathrm{asymconfused}_{\mathit{csan}}(S_1 \cup S_4, S_2, S_3)$. The acyclic net in Figure 9(b) is $\mathrm{acyclicnet}(\mathit{csan})$. Its transitions are the transitions derived from the syn-cycles in Figure 9(a). Each syn-cycle $S = \{x\}$ is removed and instead transition τ_x is created. Note that despite the fact that there is an asynchronous communication between b and c, and so they might be executed simultaneously, they are encoded as singleton transitions and the buffer place q_1 as a regular place. Note also that the behaviour of the constructed acyclic net is closely linked to the original CSA-net. First, $\mathrm{scenario}_{\mathit{acnet}}(\tau_a, \tau_d)$ and $\mathrm{scenario}_{\mathit{acnet}}(\tau_b, \tau_d)$ can be executed in any order. The only behaviour that is not preserved is the possibility of executing $\{b, c\}$ as the corresponding syn-cycles, $S_2 = \{b\}$ and $S_3 = \{c\}$, are not *explicitly* synchronised. Note that the transformed net in Figure 9(b) is cluster-acyclic.

The following result shows that the behaviour of the constructed acyclic net is equivalent to the original CSA-net. Below, for a set of transitions U of the former, $T_U = \bigcup\{S \mid \tau_S \in U\}$.

Proposition 9. *Let csan be a well-formed* CSA*-net, and acnet* $= \mathrm{acyclicnet}(\mathit{csan})$.

1. *For every cson* $\in \mathrm{maxscenarios}(\mathit{csan})$*, there is a maximal step sequence* $U_1 \dots U_k$ *of acnet such that* $T_{U_1} \dots T_{U_k}$ *is a maximal step sequence of cson.*

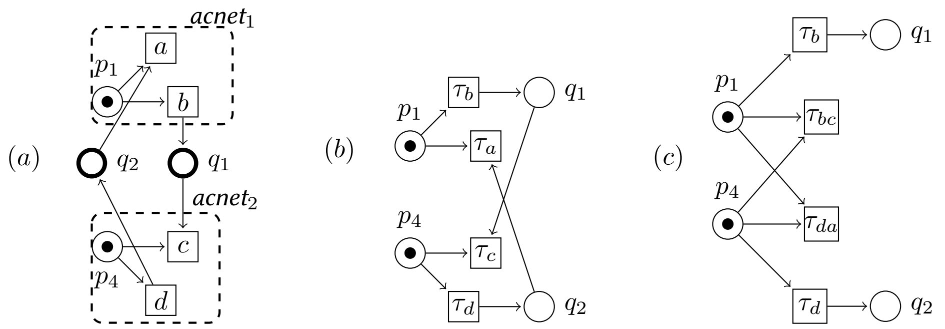


Figure 10: CSA-net (a), its encoding into acyclic net which is not cluster-acyclic (b), and another encoding (c).

2. If $U_1 \dots U_k$ is a step sequence of acnet, then $T_{U_1} \dots T_{U_k}$ is a step sequence of csan.

The first stage of the approach of removing confusion in *csan* of Figure 9(a) was already presented. We now present the second and third stages. Since the transformed acyclic net is cluster-acyclic, the techniques of the confusion-free encoding in Section 3.1 can be applied. There are two maximal clusters in this case: $\kappa_1 = \{\tau_a, \tau_b\}$ and $\kappa_2 = \{\tau_c, \tau_d\}$ with $\kappa_1 \sqsubset \kappa_2$. The associated maximal transactions are: $\mathrm{maxtrans}(\kappa_1, \{p_1\}) = \{\{\tau_a\}, \{\tau_b\}\}$, $\mathrm{maxtrans}(\kappa_2, \{p_4, q_1\}) = \{\{\tau_c\}, \{\tau_d\}\}$, and $\mathrm{maxtrans}(\kappa_2, \{p_4\}) = \{\{\tau_d\}\}$. The net in Figure 9(c) represents the third stage, which is carried out similarly to the first approach for the net in Figure 3(b).

Asynchronous communication and CSA-net acyclicity. Consider again the nets in Figure 9, for the cluster-acyclicity constraint hold. Therefore, it is possible to reuse the proposed approach discussed in Section 3.1 after transforming *csan* into $\mathrm{acyclicnet}(csan)$. However, there is an intuition that cluster-acyclicity constraint can be checked at the level of CSA-nets. This happens when *asynchronous* communications between the transitions of the component acyclic nets are *unidirectional*, in the following sense.

The net in Figure 9(a) adheres this condition. However, if exists an a$\backslash$synchronous communication between $(d, a) \in W_{csan}$ as it is portrayed in Figure 10(a), then the *csan* is not cluster-acyclic. As a result, even if this net is transformed into an acyclic net, the approach proposed previously is not applicable as it depicted in Figure 10(b). In order to solve this issue, one can construct the equivalent acyclic net differently by combining all the transitions involve in asynchronous communications into one synchronised transition. For example, the transitions $(b, c) \in W_{csan}$ and $(d, a) \in W_{csan}$ whose communications are asynchronous in Figure 10(a) can be translated into one synchronised transition τ_{bc} and τ_{da} respectively as the semantic of asynchronous communication allows those transitions to be executed together. Figure 10(c) shows the alternative encoding of *csan* in (a).

Proposition 10. *Let csan* $= (acnet_1, \dots, acnet_n, Q, W)$ *($n \geq 1$) be a well-formed CSA-net such that each acnet$_i$ is cluster-acyclic and, for each $1 \leq i < jleqn$, there are no $t \in T_{acnet_j}$ and*

$u \in T_{acnet_i}$ *such that* $(t, u) \in W^2_{csan}$. *Then* $\mathrm{acyclicnet}(csan)$ *is a cluster-acyclic.*

6. Conclusion

This paper investigated two approaches of removing confusion in acyclic nets, one of which is based on the work presented in [14]. Then the proposed solution was lifted to the level of CSA-nets. There are some issues regarding the approach of handling the confusion in CSA-nets as the acyclicity constraint may not be satisfied due to communication, and one may address these issues by combining transitions involved in synchronous communication. In the future work, we plan to extend the current work to *behavioural structured occurrence nets* [2], where the dynamic behaviour of a concurrent system is represented at different levels of abstraction.

References

[1] 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.

[2] M. Koutny, B. Randell, Structured occurrence nets: A formalism for aiding system failure prevention and analysis techniques, Fundam. Inform. 97 (2009) 41–91.

[3] B. Li, Visualisation and analysis of complex behaviours using structured occurrence nets, 2017.

[4] T. Alharbi, Analysing and visualizing big data sets of crime investigations using structured occurrence nets (PhD thesis), 2016.

[5] P. Missier, B. Randell, M. Koutny, Modelling provenance using structured occurrence networks, in: P. Groth, J. Frew (Eds.), Provenance and Annotation of Data and Processes - 4th International Provenance and Annotation Workshop, IPAW 2012, Santa Barbara, CA, USA, June 19-21, 2012, Revised Selected Papers, volume 7525 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 183–197.

[6] A. Bhattacharyya, B. Li, B. Randell, Time in structured occurrence nets, in: L. Cabac, L. M. Kristensen, H. Rölke (Eds.), Proceedings of the International Workshop on Petri Nets and Software Engineering 2016, Toruń, Poland, June 20-21, 2016, volume 1591 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016, pp. 35–55.

[7] 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, volume 2907 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 175–194.

[8] M. Koutny, B. Randell, Structured occurrence nets: incomplete, contradictory and uncertain failure evidence, Technical Report, School of Computing Science, Newcastle University, 2009.

[9] D. Koller, N. Friedman, Probabilistic graphical models: principles and techniques, MIT press, 2009.

[10] J.-P. Katoen, D. Peled, Taming Confusion for Modeling and Implementing Probabilistic

Concurrent Systems, volume 7792 of *Lecture Notes in Computer Science*, Springer, Berlin, 2013, pp. 411–430.
[11] D. Varacca, H. Völzer, G. Winskel, Probabilistic event structures and domains, Theor. Comput. Sci. 358 (2006) 173–199.
[12] S. Abbes, A. Benveniste, True-concurrency probabilistic models: Markov nets and a law of large numbers, Theor. Comput. Sci. 390 (2008) 129–170.
[13] D. Varacca, M. Nielsen, Probabilistic Petri nets and Mazurkiewicz equivalence, 2003.
[14] R. Bruni, H. C. Melgratti, U. Montanari, Concurrency and probability: Removing confusion, compositionally, CoRR abs/1710.04570 (2017).
[15] X. Chen, Z. Li, N. Wu, A. Al-Ahmari, E.-T. A.M., E. Abouel Nasr, Confusion diagnosis and avoidance of discrete event systems using supervisory control, IEEJ Transactions on Electrical and Electronic Engineering 11 (2015) n/a–n/a.
[16] X. Chen, Z. Li, N. Wu, A. M. Al-Ahmari, A. M. El-Tamimi, E. S. Abouel Nasr, Confusion avoidance for discrete event systems by P/E constraints and supervisory control, IMA Journal of Mathematical Control and Information 33 (2014) 309–332.
[17] R. D. Bamberg, Non-Deterministic Generalised Stochastic Petri Nets Modelling and Analysis, 2012.
[18] X. Chen, Z. Li, N. Wu, A. M. Al-Ahmari, A. M. El-Tamimi, E. S. Abouel Nasr, Confusion avoidance for discrete event systems by P/E constraints and supervisory control, IMA Journal of Mathematical Control and Information 33 (2016) 309–332.
[19] J. Desel, J. Esparza, Free Choice Petri Nets, volume 40 of *Cambridge Tracts in Theoretical Science*, Springer, 1995.
[20] J. Kleijn, M. Koutny, M. Pietkiewicz-Koutny, Regions of Petri nets with a/sync connections, Theor. Comput. Sci. 454 (2012) 189–198.