Verification of Communication Structured Acyclic Nets Using SAT

Nadiyah Almutairi and Maciej Koutny

School of Computing, Newcastle University
1 Science Square, Newcastle Helix, Newcastle upon Tyne NE4 5TG, U.K.
{n.m.r.almutairi2,maciej.koutny}@newcastle.ac.uk

Abstract. Model checking is an established strategy for automatic verification of software and hardware systems. It supports extensive analyses of the properties of states and behaviours of computing systems. Petri net verification using SAT-solvers has already received considerable attention and effective tools based on it have been developed. In this paper, we are concerned with the verification of communication structured acyclic nets (CSA-nets) which so far lacked robust verification methodology. CSA-nets are sets of acyclic nets which can communicate by means of synchronous and asynchronous interactions. In this paper, we introduce several propositional formulas which can be used to verify properties of CSA-nets using SAT-solvers.

Keywords: Petri net, acyclic net, communication structured acyclic net, model checking, reachability, deadlock, SAT-solver

1 Introduction

Model checking is a verification approach that, for example, performs a search of the system state space to examine whether certain property is satisfiable. This technique is often used for automatic verification of both hardware and software systems. It is a powerful method for detecting bugs in, for example, concurrent systems whose correct design poses a significant challenge [4]. For instance, [7] proposed an approach using SAT-solver to detect state encoding conflicts in Signal Transition Graphs which are converted into equivalent Petri nets and the verification is performed on the finite complete prefixes of their unfoldings ([6] extends this approach to model-checking for merged processes). [4] introduced a general study of model-checking based on Petri net unfoldings. A similar work has been done for the verification of contextual nets in [14].

A complex evolving system (CES) is composed of a number of concurrently-acting subsystems interacting with each other and with the environment. Such systems suffer from a very high complexity in terms of design and behaviour. The approach [9,10,1] based on communication structured acyclic nets (CSA-nets) — and their direct precursors communication structured occurrence nets (CSO-nets) — which employs different types of formally-defined types of abstraction can play an important role regarding the

representation of behaviours of CESs using structure to reduce the complexity of representations. A recently designed and implemented tool SONCRAFT [11] (based on the WORKCRAFT platform [13,15] providing a flexible common underpinning for graph-based models) extensive powerful support for dealing with models of CESs based on CSO-nets, including visualization and verification. Previous work on CSO-nets provided a framework for visualising and analysing behaviour of CESs [10], modelling cyber-crime investigation [1], provenance [12], and timed behaviours [2].

A satisfactory property verification for CSA-nets is still missing. To address this, our main contribution in this paper is to fill this gap by providing SAT-encoding of the most significant behavioural properties of CSA-nets, such as reachability. This is the first paper to propose a model checking solution of this kind. The proposed verification algorithms will be added as a plug-in to SONCRAFT.

The paper is organised as follows. In Section 2, we provide basic definitions concerning acyclic nets. In Section 3, we introduce formulas characterising various basic properties of acyclic nets, such as well-formedness and scenarios (valid executions) which can then be checked for satisfiability using SAT-solvers. In Section 4, basic definitions concerning communication structured acyclic nets are introduced. In Section 5, we extend the property formulas to CSA-nets. We conclude in Section 6. Due to the page limit proofs of formal results are not included.

2 Acyclic nets

In this section we introduced basic notions concerning acyclic nets.

Definition 1 (acyclic net). *An* acyclic net *is a triple* $acnet = (P,T,F)$*, where* P *and* T *are finite disjoint sets of* places *and* transitions*, respectively, and* $F \subseteq (P \times T) \cup (T \times P)$ *is a* flow relation *such that* F *is acyclic and, for every* $t \in T$*, there are* $p,q \in P$ *such that* pFt *and* tFq*.* ♢

For every $x \in P \cup T$, ${}^\bullet x = \mathrm{pre}_{acnet}(x) \triangleq \{y \mid yFx\}$ and $x^\bullet = \mathrm{post}_{acnet}(x) \triangleq \{y \mid xFy\}$ are the elements *directly preceding* and *directly following* x. We denote P, T, and F by P_{acnet}, T_{acnet}, and F_{acnet}, respectively, to indicate explicitly the net *acnet*. An acyclic net can exhibit backward non-determinism (if $|{}^\bullet p| > 1$, for some $p \in P$) and forward non-determinism (if $|p^\bullet| > 1$, for some $p \in P$).

Definition 2 (step and marking). *Let* $acnet = (P,T,F)$ *be an acyclic net.*

1. $\mathrm{steps}(acnet) \triangleq \{U \in \mathbb{P}(T) \setminus \{\varnothing\} \mid \forall t,u \in U : t \neq u \implies {}^\bullet t \cap {}^\bullet u = \varnothing\}$ *are the* steps*.*
2. $\mathrm{markings}(acnet) \triangleq \mathbb{P}(P)$ *are the* markings*.*
3. $M^{init}_{acnet} \triangleq \{p \in P \mid {}^\bullet p = \varnothing\}$ *is the default* initial *marking, and* $M^{fin}_{acnet} \triangleq \{p \in P \mid p^\bullet = \varnothing\}$ *is the default* final *marking.* ♢

Graphically, places are represented by circles, transitions by boxes, and arcs represent the flow relation F. Markings are shown by placing tokens within the circles. For acyclic nets we introduce step sequence semantics as such a semantics is used in the model of communication structured occurrence nets.

Definition 3 (enabled and executed step). *Let M be a marking of an acyclic net acnet.*

1. $\text{enabled}_{acnet}(M) \triangleq \{U \in \text{steps}(acnet) \mid {}^\bullet U \subseteq M\}$ *are the steps* enabled *at* M.
2. *A step* $U \in \text{enabled}_{acnet}(M)$ *can be* executed *yielding a new marking given by* $M' \triangleq (M \cup U^\bullet) \setminus {}^\bullet U$. *This is denoted by* $M[U\rangle_{acnet} M'$. ♢

Note that markings of acyclic nets are 'safe' by definition. The emptiness of the post-places of an executed U will hold for the well-formed acyclic nets introduced later.

Definition 4 ((mixed) step sequence). *Let acnet be an acyclic net and μ be a sequence $M_0U_1M_1 \ldots M_{k-1}U_kM_k$ ($k \geq 0$) such that $M_0, \ldots, M_k$ are markings and $U_1, \ldots, U_k$ are steps.*

1. *μ is a* mixed step sequence from M_0 to M_k *if* $M_{i-1}[U_i\rangle_{acnet} M_i$, *for every* $1 \leq i \leq k$.
2. *If μ is a mixed step sequence from M_0 to M_k, then $\sigma = U_1 \ldots U_k$ is a* step sequence from M_0 to M_k. *This is denoted by* $M_0[\sigma\rangle_{acnet} M_k$.

Also, $M_0[\rangle_{acnet} M_k$ *denotes that* M_k *is* reachable from M_0. ♢

If $k = 0$ then $\mu = M_0$ and the corresponding step sequence σ is the *empty* sequence denoted by λ. A mixed step sequence $M_0U_1M_1 \ldots M_{k-1}U_kM_k$ can also be denoted as $M_0 \xrightarrow{U_1} M_1 \ldots M_{k-1} \xrightarrow{U_k} M_k$.

Definition 5 (behavioural notions). *Let acnet be an acyclic net.*

1. $\text{sseq}(acnet) \triangleq \{\sigma \mid M^{init}_{acnet}[\sigma\rangle_{acnet} M\}$ *are the* step sequences *of acnet.*
2. $\text{maxsseq}(acnet) \triangleq \{\sigma \in \text{sseq}(acnet) \mid \neg\exists U : \sigma U \in \text{sseq}(acnet)\}$ *are the* maximal step sequences *of acnet.* ♢

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

Acyclic nets allow one to define conflicts between transitions in a structural way.

Definition 6 (structural notions). *Let $acnet = (P,T,F)$ be an acyclic net.*

1. *Two transitions $t \neq u \in T$ are in* direct (forward) conflict, *denoted $t\#_0u$, if they have a common place in their presets, i.e., ${}^\bullet t \cap {}^\bullet u \neq \varnothing$.*
2. *Two transitions $t \neq u \in T$ are in* direct backward conflict *if they have a common place in their postsets, i.e., $t^\bullet \cap u^\bullet \neq \varnothing$.*
3. *Two nodes $x, y \in P \cup T$ are in* conflict, *denoted $x\#y$, if there are transitions t and u such that $t\#_0u$ and $(t,x),(u,y) \in F^*$.*
4. *A transition $t \in T$ is in* self-conflict *if $t\#t$.* ♢

Note that conflicts between transitions emerge from having a common pre-place (in the case of forward non-determinism) or a common post-place (in the case of backward non-determinism), while concurrency results from multiple post-places emerging from a transition [10].

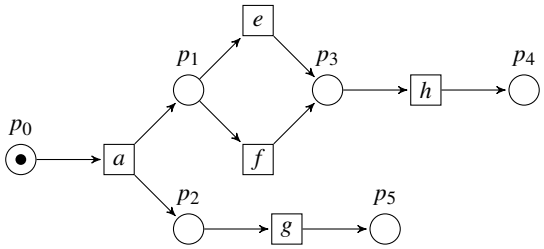

Fig. 1. An acyclic net.

Example 1. Figure 1 shows an acyclic net *acnet*, where $P = \{p_0, p_1, p_2, p_3, p_4, p_5\}$, $T = \{a, e, f, g, h\}$, and $F = \{(p_0, a), (a, p_1), (a, p_2), \ldots, (h, p_4)\}$. The initial and final markings are $\{p_0\}$ and $\{p_4, p_5\}$, respectively. Two possible steps sequences are $\sigma_1 = a\{e, g\}h$ and $\sigma_2 = a\{f, g\}h$. In this acyclic net, transitions e and f are involved both in direct forward conflict and direct backward conflict. $\diamondsuit$

If backward conflicts are not allowed, one can characterise structurally transitions which can be executed.

Definition 7 (backward deterministic acyclic net). *A* backward deterministic acyclic net *is an acyclic net acnet* $= (P, T, F)$ *such that* $|{}^\bullet p| \leq 1$, *for all* $p \in P$. $\diamondsuit$

Proposition 1. *Let acnet be a backward deterministic acyclic net.*

1. *No step sequence contains multiple occurrences of transitions.*
2. *Each transition which is not in self-conflict occurs in at least one of the step sequences.*

Figure 2 shows an example of backward deterministic acyclic net, where no transition is in self-conflict and it can be executed.

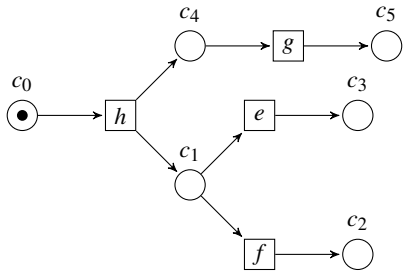

Fig. 2. A backward deterministic acyclic net.

2.1 Occurrence nets and scenarios

An occurrence net [9] is a Petri net describing a single execution of some concurrent system in such a way that only the details of causality and concurrency between transitions are captured. It is deterministic and can represent an execution history of a system with a clear interpretation of causal dependencies between transitions.

Definition 8 (occurrence net). *An* occurrence net *is an acyclic net onet* $= (P,T,F)$ *such that* $|{}^\bullet p| \leq 1$ *and* $|p^\bullet| \leq 1$*, for every* $p \in P$. ♢

We use occurrence nets embedded in an acyclic net to provide a characterisation of its potential executions.

Definition 9 (scenario and maximal scenario). *Let acnet be an acyclic net.*

1. *A* scenario *of acnet is an occurrence net ocnet such that:*
   (a) $T_{ocnet} \subseteq T_{acnet}$ *and* $P_{ocnet} = M^{init}_{acnet} \cup \text{post}_{acnet}(T_{ocnet})$.
   (b) $\text{pre}_{ocnet}(t) = \text{pre}_{acnet}(t)$ *and* $\text{post}_{ocnet}(t) = \text{post}_{acnet}(t)$*, for every* $t \in T_{ocnet}$.
2. *A* maximal scenario *of acnet is a scenario ocnet such that there is no other scenario* $ocnet'$ *satisfying* $T_{ocnet} \subset T_{ocnet'}$.

The sets of all scenarios and maximal scenarios of acnet are scenarios(*acnet*) *and* maxscenarios(*acnet*)*, respectively.* ♢

Figure 3 shows the only two maximal scenarios of the acyclic net in Figure 1.

Different step sequences may generate the same scenario, and each scenario is generated by at least one step sequence. Two step sequences generate the same scenario iff their executed transitions are identical. Each scenario is identified by the set of its transitions, and we denote by $\text{scenario}_{acnet}(V)$ the unique scenario with the transition set V.

A maximal scenario can be seen as a *deterministic process*, which captures an equivalence class of maximal step sequences that are distinguishable only in the order of concurrent transitions [7].

Example 2. Figure 3 illustrates the last point by identifying two maximal scenarios, $ocnet_1 = \text{scenario}_{acnet}(\{a,e,g,h\})$ and $ocnet_2 = \text{scenario}_{acnet}(\{a,f,g,h\})$, of the acyclic net *acnet* in Figure 1. For example, $ocnet_1$ captures the following five executions:
$\{p_0\} \xrightarrow{a} \{p_1,p_2\} \xrightarrow{\{e,g\}} \{p_3,p_5\} \xrightarrow{h} \{p_4,p_5\}$,
$\{p_0\} \xrightarrow{a} \{p_1,p_2\} \xrightarrow{e} \{p_2,p_3\} \xrightarrow{g} \{p_3,p_5\} \xrightarrow{h} \{p_4,p_5\}$,
$\{p_0\} \xrightarrow{a} \{p_1,p_2\} \xrightarrow{e} \{p_2,p_3\} \xrightarrow{\{g,h\}} \{p_4,p_5\}$,
$\{p_0\} \xrightarrow{a} \{p_1,p_2\} \xrightarrow{e} \{p_2,p_3\} \xrightarrow{h} \{p_2,p_4\} \xrightarrow{g} \{p_4,p_5\}$, and
$\{p_0\} \xrightarrow{a} \{p_1,p_2\} \xrightarrow{g} \{p_1,p_5\} \xrightarrow{e} \{p_3,p_5\} \xrightarrow{h} \{p_4,p_5\}$. ♢

Not every acyclic net can be seen as a valid representation of causal dependencies in some execution of a concurrent system. The next definition introduces the notion of well-formedness which basically means that all the executions an acyclic net are the executions of its scenarios. In addition, an acyclic net has no redundant transitions whenever each of its transitions can be executed.

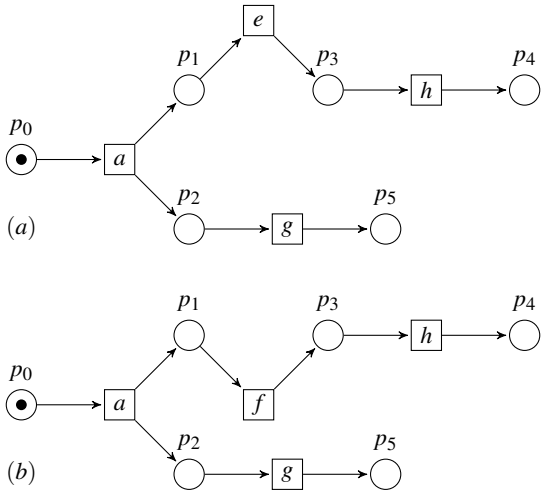

Fig. 3. Maximal scenarios of the acyclic net in Figure 1: $ocnet_1 = \text{scenario}_{acnet}(\{a,e,g,h\})$ in (a), and $ocnet_2 = \text{scenario}_{acnet}(\{a,f,g,h\})$ in (b).

Definition 10 (well-formedness and non-redundant transitions). *An acyclic net acnet is* well-formed *if* $\text{sseq}(acnet) = \bigcup \text{sseq}(\text{scenarios}(acnet))$. *Moreover, acnet has* non-redundant transitions *if* $T_{acnet} = \bigcup\{\bigcup \sigma \mid \sigma \in \text{sseq}(acnet)\}$. ♢

Each backward deterministic acyclic net, and so also each occurrence net, is well-formed. Moreover, each occurrence net has non-redundant transitions. As the result shows, in a given execution a well-formed acyclic net no token can be created 'more than once'.

Proposition 2. *acnet is well-formed iff the following hold for every step sequence* $U_1 \ldots U_k \in \text{sseq}(acnet)$*:*

1. $t^\bullet \cap u^\bullet = \varnothing$*, for every* $1 \leq i \leq k$ *and all distinct* $t,u \in U_i$.
2. $U_i^\bullet \cap U_j^\bullet = \varnothing$*, for all* $1 \leq i < j \leq k$.

3 Verifying properties of acyclic nets

The task of a SAT-solver is to find a satisfying assignment for the variables used in a propositional boolean formula ϕ. That is, the SAT-solver looks for an assignment $f : Var \to \{0,1\}$ for the variables Var which occur in ϕ that makes ϕ true, i.e., $\phi[f] = 1$. And, if such an assignment exists, ϕ is *satisfiable*. The formula ϕ is often expressed in *conjunctive normal form (*CNF*)*, i.e., a conjunction of clauses which are disjunctions of literals (where a literal is either a variable or the negation of a variable).

3.1 Identifying scenarios

For a given acyclic net $acnet = (P,T,F)$, a subset of transitions $T' \subseteq T$ induces a scenario whenever the following three properties hold:

- *Causality*: all non-initial pre-places of T' received tokens from T'.
- *No Forward Conflict*: the transitions in T' are free from forward conflicts.
- *No Backward Conflict*: the transitions in T' are free from backward conflicts.

The next definition and proposition make this formal, by identifying the scenarios of an acyclic net with occurrence net restrictions induced by subsets of transitions.

Definition 11 (restricting acyclic net). *Let $acnet = (P,T,F)$ be an acyclic net and $T' \subseteq T$. Then $acnet|_{T'} = (P',T',F|_{(P'\times T')\cup(T'\times P')})$, where $P' = M^{init}_{acnet} \cup \mathrm{post}_{acnet}(T')$, is the* restriction *of acnet to T'.* ♢

Note that if $acnet|_{T'}$ is a scenario, then $acnet|_{T'} = \mathrm{scenario}_{acnet}(T')$.

Proposition 3. *The following statements are equivalent for an acyclic net $acnet = (P,T,F)$ and $T' \subseteq T$:*

1. $acnet|_{T'} \in \mathrm{scenarios}(acnet)$.
2. $acnet|_{T'}$ *is an occurrence net and* $\mathrm{pre}_{acnet}(T') \setminus M^{init}_{acnet} \subseteq \mathrm{post}_{acnet}(T')$.

We want to obtain a propositional formula which can be used to identify all the scenarios of an acyclic net *acnet*, i.e., to construct a formula $Scenario_{acnet}$ which evaluates to 1 iff all the transitions assigned 1 induce a scenario of *acnet* (note that *acnet* is not assumed to be well-formed). The following boolean variables will be used in the construction of $Scenario_{acnet}$ and the translation into SAT :

- For every $t \in T$, we have a variable in_t tracing that t belongs to a scenario.

The constraints on the above variables are defined, following Proposition 3, as follows:

- To ensure that all non-initial pre-places of T' received tokens from T':

$$Causality_{acnet} \triangleq \bigwedge_{t\in T} \Big(in_t \rightarrow \bigwedge_{p\in {}^\bullet t \setminus M^{init}_{acnet}} \bigvee_{u\in {}^\bullet p} in_u\Big)$$

  where $\bigwedge_{p\in {}^\bullet t\setminus M^{init}_{acnet}} \bigvee_{u\in {}^\bullet p} in_u$ is set to 1 if ${}^\bullet t \subseteq M^{init}_{acnet}$.
- To ensure that scenario has no forward conflicts:

$$NoForwardConflict_{acnet} \triangleq \bigwedge_{t\in T} \bigwedge_{u\in ({}^\bullet t)^\bullet \setminus \{t\}} \neg(in_t \wedge in_u)$$

  where $\bigwedge_{u\in ({}^\bullet t)^\bullet\setminus\{t\}} \neg(in_t \wedge in_u)$ is omitted if $({}^\bullet t)^\bullet = \{t\}$.
- To ensure that scenario has no backward conflicts:

$$NoBackwardConflict_{acnet} \triangleq \bigwedge_{t\in T} \bigwedge_{u\in {}^\bullet(t^\bullet)\setminus\{t\}} \neg(in_t \wedge in_u)$$

  where $\bigwedge_{u\in {}^\bullet(t^\bullet)\setminus\{t\}} \neg(in_t \wedge in_u)$ is omitted if ${}^\bullet(t^\bullet) = \{t\}$.

Then the formula which characterises all the scenarios of *acnet* is:

$$Scenario_{acnet} \triangleq Causality_{acnet} \wedge NoForwardConflict_{acnet} \wedge NoBackwardConflict_{acnet}$$

The size of the above formula (in terms of the occurrences of literals) is bounded by $|T| + 3 \cdot |T| \cdot \min\{|F|^2, |P| \cdot |T|\}$.

Proposition 4. *Let acnet be an acyclic net.*

1. *If f is a satisfying assignment for $Scenario_{acnet}$ then $acnet|_{f^{-1}(1)}$ is a scenario of acnet.*
2. *For every scenario $acnet'$ of the acyclic net acnet, there is a satisfying assignment f for $Scenario_{acnet}$ such that $acnet|_{f^{-1}(1)} = acnet'$.*

Hence, we can use $Scenario_{acnet}$ to find all the subsets T' of transitions of *acnet* inducing scenarios after translating it into CNF and feeding into a SAT-solver.

3.2 Well-formedness

To characterise well-formed acyclic nets, we introduce formulas capturing the enabledness of transitions, for every $t \in T$:

$$Enabled_t \triangleq \bigwedge_{u \in (^\bullet t)^\bullet} \neg in_u \ \wedge \bigwedge_{p \in {}^\bullet t \setminus M^{init}_{acnet}} \bigvee_{u \in {}^\bullet p} in_u$$

where the second clause is omitted if ${}^\bullet t \subseteq M^{init}_{acnet}$. We then define:

$$NotWellFormed_{acnet} \triangleq Scenario_{acnet} \wedge \bigvee_{t \in T} \left(Enabled_t \wedge \bigvee_{u \in {}^\bullet(t^\bullet)} in_u \right)$$

Intuitively, in the above formula, $Scenario_{acnet}$ 'selects' one of the executions of *acnet* which does not violate well-formedness, and the second part means that there is a transition which can be executed after that violating well-formedness.
The size of the above formula is bounded by $|T| + 6 \cdot |T| \cdot \min\{|F|^2, |P| \cdot |T|\}$.

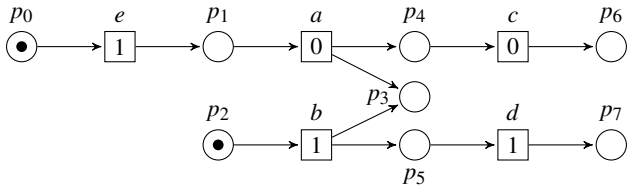

Fig. 4. An acyclic net which is not well-formed.

Proposition 5. *An acyclic net acnet is well-formed iff* $NotWellFormed_{acnet}$ *does not have a satisfying assignment.*

Hence we can use $NotWellFormed_{acnet}$ to find out whether *acnet* is well-formed after translating it into CNF and feeding into a SAT-solver.

Example 3. The acyclic net in Figure 4 is not well-formed. This is so because for the assignment f illustrated in Figure 4 (where $f(e) = f(b) = f(d) = 1$ and $f(a) = f(c) = 0$) we have the following: (i) $Scenario_{acnet}$ evaluates to 1 inducing $\text{scenario}_{acnet}(\{e,b,d\})$; (ii) $Enabled_a[f] = 1$; and (iii) $(\bigvee_{u \in {}^\bullet(a^\bullet)} in_u)[f] = (in_a \vee in_b)[f] = (0 \vee 1) = 1$. Hence, $NotWellFormed_{acnet}[f] = 1$. ♢

3.3 Non-redundant transitions

Definition 10(2) asserts that a well-formed *acnet* has non-redundant transitions if each transition is guaranteed to occur in at least one of its steps sequences. In other words, if for each transition t there is a valid scenario in scenarios(*acnet*) that contains t.

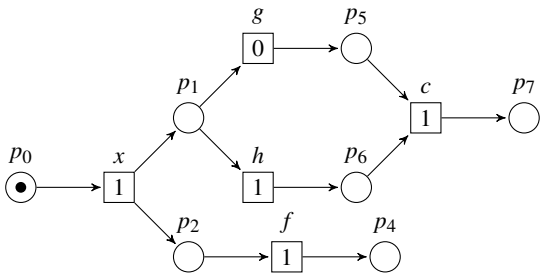

Fig. 5. A well-formed acyclic net with a redundant transition c (see Example 4).

Example 4. Consider the backward deterministic acyclic net in Figure 5. The two transitions g and h are in direct conflict, and so c is in a self-conflict. This means that c can never be executed, i.e., there is no step sequence $\sigma \in \text{sseq}(acnet)$ such that $c \in \bigcup \sigma$. Therefore, c is not included in any scenario of scenarios(*acnet*), and can be seen as an *irrelevant* part of *acnet*.
For the example in Figure 5, we can see that the boolean formula $Scenario_{acnet}$ is not satisfiable for an assignment r such that $r(in_c) = 1$ since

$$\begin{aligned} Causality_{acnet} &= (in_x \to 1) \wedge (in_g \to in_x) \wedge (in_h \to in_x) \wedge \\ &\quad (in_f \to in_x) \wedge (in_c \to (in_g \wedge in_h)) \\ NoForwardConflict_{acnet} &= \neg(in_g \wedge in_h) \wedge \neg(in_h \wedge in_g)\,. \end{aligned}$$

Hence, if $r(in_c) = 1$ and $Causality_{acnet}[r] = 1$ then we have $r(in_g) = r(in_h) = 1$, and so $NoForwardConflict_{acnet}[r] = 0$. This means there is no scenario induced by transitions whose associated variables are assigned 1 and c belongs to such a scenario. ♢

Checking whether a transition t belongs to at least one scenario can be done using the following formula:

$$NonRedundant^{t}_{acnet} \triangleq in_t \wedge Scenario_{acnet} .$$

As a result, a well-formed acyclic net has non-redundant transitions if $NonRedundant_t$ is satisfiable, for each $t \in T$.

Proposition 6. *A well-formed acyclic net acnet has non-redundant transitions iff the formula $NonRedundant^{t}_{acnet}$ is satisfiable, for every $t \in T$.*

3.4 Maximal scenarios

Maximal scenarios are *complete* scenarios, which means that no transitions are enabled. In this case, we assume that *acnet* is a well-formed acyclic net, and define:

$$MaxScenario_{acnet} \triangleq Scenario_{acnet} \wedge \bigwedge_{t \in T} \neg Enabled_t$$

The size of $MaxScenario_{acnet}$ is bounded by $|T|+5\cdot|T|\cdot\min\{|F|^2,|P|\cdot|T|\}$. Intuitively, it is satisfiable for all the scenarios that do not enable any transitions.

Proposition 7. *Let acnet be a well-formed acyclic net.*

1. *If f is a satisfying assignment for $MaxScenario_{acnet}$ then $acnet|_{f^{-1}(1)}$ is a maximal scenario of acnet.*
2. *For every maximal scenario $acnet'$ of acnet, there is a satisfying assignment f for $MaxScenario_{acnet}$ such that $acnet|_{f^{-1}(1)} = acnet'$.*

Hence we can use $MaxScenario_{acnet}$ to find the subsets T' of transitions of *acnet* inducting maximal scenarios after translating it into CNF and feeding into a SAT-solver.

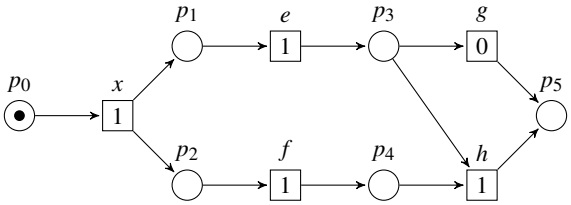

Fig. 6. A well-formed acyclic net in Example 5.

Example 5. Figure 6 shows a well-formed acyclic net that exhibits both forward and backward conflict. Observe that all the transitions in $\{x,e,f,h\}$ satisfy the causality condition, and have no forward nor backward conflict. They also lead to a marking $\{p_5\}$ where no transitions are enabled. Therefore, the set of transitions $\{x,e,f,h\}$ induces a maximal scenario in maxscenarios$(acnet)$. Thus, the $MaxScenario_{acnet}$ formula is satisfied by the assignment indicated in Figure 6 (the situation is the same if g is used instead of h). ♢

3.5 Marked places and deadlocked scenarios

Consider a scenario of a well-formed acyclic net *acnet*. Then a place $p \in P$ is marked when at least one transition in its preset has been executed and no transitions in its postset have been executed. This can be captured as follows:

$$Mark_p \triangleq \bigvee_{t \in {}^\bullet p} in_t \ \wedge \bigwedge_{u \in p^\bullet} \neg in_u$$

where $\bigvee_{t \in {}^\bullet p}$ is omitted if p has no input transitions, and $\bigwedge_{u \in p^\bullet} \neg in_u$ is omitted if p has no output transitions. The size of the above formula is bounded by $2 \cdot |T|$, and it can be used to answer questions such as: *is a specific marking reachable?*

The following formula can be used to check whether there is a reachable marking in which all places in a non-empty set $M \subseteq P$ are marked:

$$Reach^M_{acnet} \triangleq Scenario_{acnet} \ \wedge \bigwedge_{p \in M} Mark_p$$

Proposition 8. *Let acnet be a well-formed acyclic net and M be a non-empty set of its places. Then there is a reachable marking M' satisfying $M \subseteq M'$ iff $Reach^M_{acnet}$ has a satisfying assignment.*

No execution of an acyclic net $acnet = (P, T, F)$ can be extended indefinitely. However, one may consider a scenario as *deadlocked* if it is maximal and some of the places it marks does not belong to M^{fin}_{acnet}. (Verification of deadlock-freeness using SAT-solvers has been studied in, e.g., [14,6,5]). All the deadlocked scenarios can be captured by the following formula:

$$DeadlockScenario_{acnet} \triangleq MaxScenario_{acnet} \ \wedge \bigvee_{p \in P \setminus M^{fin}_{acnet}} Mark_p$$

The size of $MaxScenario_{acnet}$ is bounded by $|T| + 5 \cdot |T| \cdot \min\{|F|^2, |P| \cdot |T|\} + 2 \cdot |P| \cdot |T|$.

3.6 Backward deterministic acyclic nets

In this case, some formulas defined earlier can be simplified:

- Causality: $Causality_{acnet} \triangleq \bigwedge_{t \in T} \left(in_t \rightarrow \bigwedge_{u \in {}^\bullet({}^\bullet t)} in_u \right)$.
- Enabling: $Enabled_t \triangleq \bigwedge_{u \in ({}^\bullet t)^\bullet} \neg in_u \wedge \bigwedge_{u \in {}^\bullet({}^\bullet t)} in_u$.
- Scenario: $Scenario_{acnet} \triangleq Causality_{acnet} \wedge NoForwardConflict_{acnet}$.
- Place marking (for $p \in P$ and $t \in T$ satisfying $\{t\} = {}^\bullet p$): $Mark_p \triangleq in_t \wedge \bigwedge_{u \in p^\bullet} \neg in_u$.
- Checking well-formedness is not needed as *acnet* is well-formed.

4 Communication structured acyclic nets

Communication structured acyclic nets (CSA-nets) add communication to represent the interaction among several separated subsystems [10]. Each CSA-net is a set of *acnet*s

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 [8]). 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.

Definition 12 (communication structured acyclic net). *A* communication structured acyclic net (or CSA-net) *is a tuple* $csan \triangleq (acnet_1, \ldots, acnet_n, Q, W)$ *(*$n \geq 1$*) such that:*

1. $acnet_1, \ldots, acnet_n$ *are well-formed acyclic nets with disjoint sets of nodes (i.e., places and transitions). We also denote:*

$$\begin{array}{ll} P_{csan} \triangleq P_{acnet_1} \cup \cdots \cup P_{acnet_n} & Q_{csan} \triangleq Q \\ T_{csan} \triangleq T_{acnet_1} \cup \cdots \cup T_{acnet_n} & W_{csan} \triangleq W \\ F_{csan} \triangleq F_{acnet_1} \cup \cdots \cup F_{acnet_n} & net_{csan,i} \triangleq acnet_i \quad (for\ \ 1 \leq i \leq n)\ . \end{array}$$

2. Q_{csan} *is a set of* buffer places *and* $W_{csan} \subseteq (Q_{csan} \times T_{csan}) \cup (T_{csan} \times Q_{csan})$ *is a set of arcs adjacent to the buffer places satisfying the following:*
   (a) $Q_{csan} \cap (P_{csan} \cup T_{csan}) = \varnothing$.
   (b) For every buffer place q:
      i. There is at least one transition t such that $tW_{csan}q$.
      ii. If $tW_{csan}q$ *and* $qW_{csan}u$ *then transitions t and u belong to different* $acnet_i$*s.*
   Moreover, for every $x \in P_{csan} \cup T_{csan} \cup Q_{csan}$*:*

$$\begin{array}{l} \mathrm{pre}_{csan}(x) \triangleq \{y \mid y F_{csan} x \ \vee \ y W_{csan} x\} \\ \mathrm{post}_{csan}(x) \triangleq \{y \mid x F_{csan} y \ \vee \ x W_{csan} y\} \end{array}$$

   denote direct predecessors and successors of x.
3. $x_1 \in \mathrm{pre}_{csan}(x_2), \ldots, x_{k-1} \in \mathrm{pre}_{csan}(x_k)$ *and* $x_1 = x_k$ *imply* $\{x_1, \ldots, x_k\} \cap P_{csan} = \varnothing$*, for all* $x_1, \ldots, x_k \in P_{csan} \cup T_{csan} \cup Q_{csan}$.

The set of all CSA*-nets is denoted by CSAN.* ♢

Note that the structure of a CSA-net may include cycles which are a means to implement synchronous communication. Each such cycle can only involve buffer places. In what follows, $csan = (acnet_1, \ldots, acnet_n, Q, W)$ $(n \geq 1)$ is a fixed CSA-net.

Definition 13 (step and marking).

1. $\mathrm{steps}(csan) \triangleq \{U \in \mathbb{P}(T_{csan}) \setminus \{\varnothing\} \mid \forall t, u \in U : t \neq u \implies \mathrm{pre}_{csan}(t) \cap \mathrm{pre}_{csan}(u) = \varnothing\}$ *are the* steps *of csan.*
2. $\mathrm{markings}(csan) \triangleq \mathbb{P}(P_{csan} \cup Q_{csan})$ *are the* markings *of csan.*
3. $M^{init}_{csan} \triangleq M^{init}_{acnet_1} \cup \cdots \cup M^{init}_{acnet_n}$ *is the default* initial *marking of csan, and* $M^{fin}_{csan} \triangleq M^{fin}_{acnet_1} \cup \cdots \cup M^{fin}_{acnet_n}$ *is the default* final *marking of csan.* ♢

Definition 14 (enabled and executed step). *Let M be a marking of csan.*

1. $\mathrm{enabled}_{csan}(M) \triangleq \{U \in \mathrm{steps}(csan) \mid \mathrm{pre}_{csan}(U) \subseteq M \cup (\mathrm{post}_{csan}(U) \cap Q)\}$ *are the steps* enabled *at M.*

2. *A step* $U \in \text{enabled}_{csan}(M)$ *can be* executed *yielding a new marking given by* $M' \triangleq (M \cup \text{post}_{csan}(U)) \setminus \text{pre}_{csan}(U)$. *This is denoted by* $M[U\rangle_{csan} M'$. ♢

The definition that follow introduce notion similar to those introduced earlier for acyclic nets.

Definition 15 ((mixed) step sequence). *Let* $\mu = M_0 U_1 M_1 \ldots M_{k-1} U_k M_k$ *($k \geq 0$) be a sequence such that* $M_0, \ldots, M_k$ *are markings and* $U_1, \ldots, U_k$ *are steps of csan.*

1. μ *is a* mixed step sequence from M_0 to M_k *if* $M_{i-1}[U_i\rangle_{csan} M_i$, *for every* $1 \leq i \leq k$.
2. *If* μ *is a mixed step sequence from* M_0 *to* M_k, *then* $\sigma = U_1 \ldots U_k$ *is a* step sequence from M_0 to M_k. *This is denoted by* $M_0[\sigma\rangle_{csan} M_k$. *Also,* $M_0[\rangle_{csan} M_k$ *denotes that* M_k *is* reachable from M_0.
3. $\text{sseq}(csan) \triangleq \{\sigma \mid M^{init}_{csan}[\sigma\rangle_{csan} M\}$ *are the* step sequences *of csan.*
4. $\text{maxsseq}(csan) \triangleq \{\sigma \in \text{sseq}(csan) \mid \neg\exists U : \sigma U \in \text{sseq}(csan)\}$ *are the* maximal step sequences *of csan.* ♢

Thus, in *csan* the buffer places pass tokens between the different component acyclic nets. Note that just as acyclic nets, CSA-nets may exhibit backward and forward non-determinism. Moreover, they can additionally contain cycles involving buffer places. Within such cycles tokens can be produced and consumed in a single executed step. This provides a mechanism enabling synchronous communication, and is a feature which is not supported by acyclic nets (and the standard Petri net models).

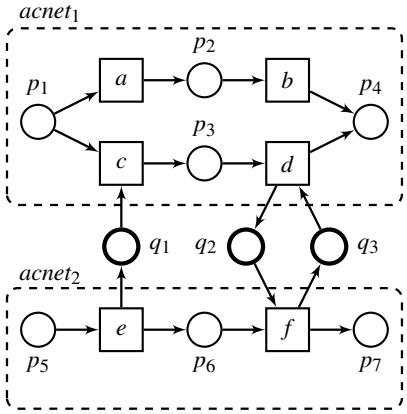

Fig. 7. A communication structured acyclic net.

Example 6. In Figure 7, transitions e and c communicate *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. The initial marking in this case is $M^{init}_{csan} = \{p_1, p_5\}$. The steps are:

$$\text{steps}(csan_1) = \{U \in \mathbb{P}(\{a,b,c,d,e,f\}) \setminus \{\varnothing\} \mid a \in U \implies c \notin U\},$$

and all the maximal step sequences are:

$$\text{maxsseq}(csan) = \{\{a,e\}b, a\{b,e\}, eab, aeb, abe, ec\{d,f\}, \{e,c\}\{d,f\}\}\ .$$

Also, for example, $\text{post}_{acnet}(e) = \{p_6, q_1\}$ and the firing of e will produce tokens in those two places. ♢

Definition 16 (backward deterministic CSA-net). *csan is* backward deterministic *if the following hold:*

1. *$acnet_1, \ldots, acnet_n$ are backward deterministic acyclic nets.*
2. *$|\text{pre}_{csan}(q)| = 1$, for every $q \in Q$.* ♢

Definition 17 (sync-cycle). *A* sync-cycle *of csan is a maximal non-empty set of transitions $\mathbb{S} \subseteq T$ such that, for all $t \neq u \in \mathbb{S}$, $(t,u) \in W^+$. The set of all sync-cycles is denoted by SC^{csan}.* ♢

Note that $\{t\} \in SC^{csan}$, for every $t \in T_{csan}$. The idea behind the notion of sync-cycles is to represent maximum number of synchronised communication sub-systems [10]. In Figure 7, there is one non-singleton sync-cycle $\mathbb{S} = \{d, f\}$.

4.1 Scenarios of CSA-nets

Scenarios of CSA-nets are based on CSA-nets which involve occurrence nets rather than acyclic nets.

Definition 18 (communication structured occurrence net). *csan is a* communication structured occurrence net (or CSO-net) *if the following hold:*

1. *$acnet_1, \ldots, acnet_n$ are occurrence nets.*
2. *$|\text{pre}_{csan}(q)| = 1$ and $|\text{post}_{csan}(q)| \leq 1$, for every $q \in Q$.* ♢

The scenarios of CSA-nets are subnets which are both backward and forward deterministic.

Definition 19 (scenario and maximal scenario).

1. *A* scenario *of csan is a* CSO*-net cson with n component occurrence nets such that:*
   (a) $net_{cson,i} \in \text{scenarios}(net_{cson,i})$, for every $1 \leq i \leq n$.
   (b) $Q_{cson} \subseteq Q$ and $W_{cson} \subseteq W$.
   (c) $\text{pre}_{cson}(t) = \text{pre}_{csan}(t)$ and $\text{post}_{cson}(t) = \text{post}_{csan}(t)$, for every $t \in T_{cson}$.
2. *A* maximal scenario *of csan is a scenario cson such that there is no scenario cson$'$ satisfying $T_{cson} \subset T_{cson'}$.*

The set of all scenarios of csan is scenarios$(csan)$*, and the set of all maximal scenarios of csan is* maxscenarios$(csan)$. ♢

Scenarios represent possible deterministic executions (concurrent histories). Maximal scenarios are complete in the sense that they cannot be extended any further. Note that Definition 19(1.*a*) requires that each component net of *cson* is a scenario (and so, in particular, an occurrence net) of the corresponding component acyclic net of *csan*.

Example 7. Figure 8 represents a CSO-net for *csan* in Figure 7 which is also one of its maximal scenarios whose transitions are $\{e,c,d,f\}$ and no other transitions can be enabled at the final marking $\{p_4, p_7\}$. ♢

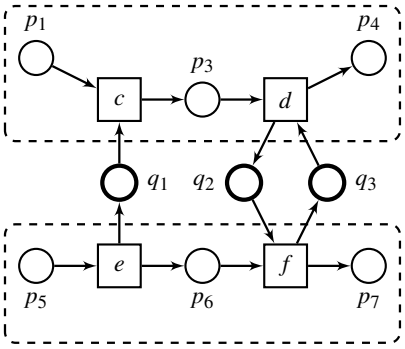

Fig. 8. A communication structured occurrence net.

4.2 Well-formed CSA-nets

Definition 20 (well-formedness and non-redundant transitions). *csan is* well-formed *if* $\text{sseq}(csan) = \bigcup \text{sseq}(\text{scenarios}(csan))$. *Moreover, csan has* non-redundant transitions *if* $T_{csan} = \bigcup\{\bigcup \sigma \mid \sigma \in \text{sseq}(csan)\}$. ♢

As shown in [9], all CSO-nets are well-formed.

Proposition 9. *csan is well-formed iff, for every* $U_1 \ldots U_k \in \text{sseq}(csan)$*:*

1. $\text{post}_{csan}(t) \cap \text{post}_{csan}(u) = \varnothing$*, for every* $1 \leq i \leq k$ *and all distinct* $t, u \in U_i$.
2. $\text{post}_{csan}(U_i) \cap \text{post}_{csan}(U_j) = \varnothing$*, for all* $1 \leq i < j \leq k$.

5 Verifying properties of CSA-nets

The development and intuitive meaning of formulas needed to check the basic properties of CSA-nets are similar as in the case of acyclic nets. Therefore, we will avoid repeating some of the explanations and formal properties. Also, the sizes of formulas introduced for CSA-nets are similar to those developed for acyclic nets with $|T|$ being sometimes replaced by $|SC^{csan}|$ (i.e., the number of syn-cycles of *csan*).

Identifying scenarios

We say that a set of transitions $T' \subseteq T_{csan}$ induces a scenario of *csan* if the following are satisfied:

- *Causality*: all non-initial pre-places of T' received tokens from T'.
- *No Forward Conflict*: transitions in T' are free from forward conflicts.
- *No Backward Conflict*: transitions in T' are free from backward conflicts.

The next definition and proposition make this more formal.

Definition 21 (restricting communication structured acyclic net). *The* restriction *of csan to* $T' \subseteq T_{csan}$ *is* $csan|_{T'} \triangleq (acnet_1|_{T'}, \ldots, acnet_n|_{T'}, Q'|_{T'}, W|_{(Q' \times T') \cup (T' \times Q')})$, *where* $Q' = Q \cap \mathrm{post}_{acnet}(T')$. ♢

Proposition 10. *The following are equivalent for* $T' \subseteq T_{csan}$*:*

1. $csan|_{T'} \in \mathrm{scenarios}(csan)$.
2. $csan|_{T'}$ *is a* CSO*-net and* $\mathrm{pre}_{csan}(T') \setminus M^{init}_{csan} \subseteq \mathrm{post}_{csan}(T')$.

The following boolean variables will be used in the construction of $Scenario_{csan}$ and the translation into SAT problem.

– For every $t \in T_{csan}$, we have a variable in_t tracing that t belongs to a scenario.

The constraints on the above variables are defined, following Proposition 3, as follows:

– To ensure that all non-initial pre-places of T' received tokens from T':

$$Causality_{csan} \triangleq \bigwedge_{t \in T_{csan}} \Big(in_t \rightarrow \bigwedge_{p \in \mathrm{pre}_{csan}(t) \setminus M^{init}_{csan}} \bigvee_{u \in \mathrm{pre}_{csan}(p)} in_u\Big)$$

– To ensure that scenario has no forward conflicts:

$$NoForwardConflict_{csan} \triangleq \bigwedge_{t \in T_{csan}} \bigwedge_{u \in \mathrm{post}_{csan}(\mathrm{pre}_{csan}(t)) \setminus \{t\}} \neg\big(in_t \wedge in_u\big)$$

– To ensure that scenario has no backward conflicts:

$$NoBackwardConflict_{csan} \triangleq \bigwedge_{t \in T_{csan}} \bigwedge_{u \in \mathrm{pre}_{csan}(\mathrm{post}_{csan}(t)) \setminus \{t\}} \neg\big(in_t \wedge in_u\big)$$

Note that sometimes parts of the above formulas may be omitted, similarly as in the case of acyclic nets. Then the formula which characterises all the scenarios of *csan* is:

$$Scenario_{csan} \triangleq Causality_{csan} \wedge NoForwardConflict_{csan} \wedge NoBackwardConflict_{csan}$$

The satisfying assignments of $Scenario_{csan}$ identify precisely all the scenarios of *csan* which is not necessary well-formed.

Example 8. All the transitions in Figure 9 which have value 1 assigned by the indicated assignment f represent a scenario for *csan* because each part of $Scenario_{csan}$ is satisfied:

$$\begin{aligned}
&Causality_{csan}[f] \\
&\quad = ((in_d \rightarrow in_A) \wedge (in_e \rightarrow in_f) \wedge (in_A \rightarrow in_e) \wedge \\
&\qquad (in_B \rightarrow (in_e \wedge in_C)) \wedge ((in_C \rightarrow (in_f \wedge in_B))) \wedge (in_f \rightarrow 1))[f] \\
&\quad = (0 \rightarrow 0) \wedge (1 \rightarrow 1) \wedge (0 \rightarrow 1) \wedge (1 \rightarrow (1 \wedge 1)) \wedge (1 \rightarrow (1 \wedge 1)) \wedge (1 \rightarrow 1) = 1 \\
&NoForwardConflict_{csan}[f] \\
&\quad = (\neg(in_A \wedge in_B) \wedge \neg(in_B \wedge in_A))[f] = \neg(0 \wedge 1) \wedge \neg(1 \wedge 0) = 1 \\
&NoBackwardConflict_{csan}[f] \\
&\quad = (\neg(in_A \wedge in_B) \wedge \neg(in_B \wedge in_A))[f] = \neg(0 \wedge 1) \wedge \neg(1 \wedge 0) = 1\ .
\end{aligned}$$

Hence, $Scenario_{csan}[f] = 1 \wedge 1 \wedge 1 = 1$. ♢

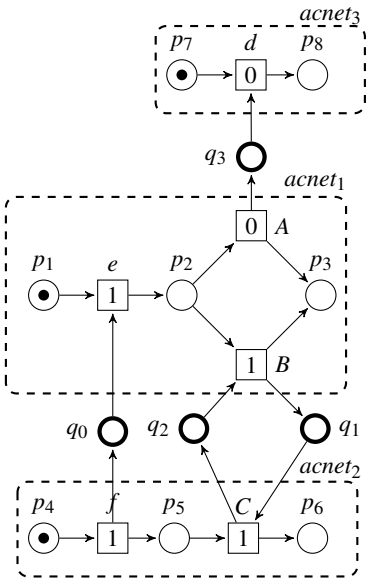

Fig. 9. A CSA-net with a scenario indicated by transitions assigned 1.

The formula $Scenario_{csan}$ represents a scenario in general, which not necessarily maximal. In order to evaluate the maximality of a scenario, we need to evaluate the enabledness property as any scenario is maximal iff there are no enabled steps that can be executed. In this case, however, sync-cycles (which include individual transitions) rather than individual transitions need to be checked for enabledness.

The following formula captures the enabledness of a syn-cycle, for every $\mathbb{S} \in SC^{csan}$:

$$Enabled_{\mathbb{S}} \triangleq \bigwedge_{u \in \mathrm{post}_{csan}(\mathrm{pre}_{csan}(\mathbb{S}))} \neg in_u \ \wedge \bigwedge_{p \in \mathrm{pre}_{csan}(\mathbb{S}) \setminus (M^{init}_{csan} \cup (Q \cap \mathrm{post}_{csan}(\mathbb{S})))} \ \bigvee_{u \in \mathrm{pre}_{csan}(p)} in_u \ .$$

Thus $\mathbb{S}$ is a set of synchronised transitions whose enabledness depends on each other. In Figure 9, B and C are transitions involved in synchronous communication and $\mathbb{S} = \{B,C\}$ is a syn-cycle.

Finally, the following formula represents the set of transitions that induce a maximal scenario of a *well-formed csan*:

$$MaxScenario_{csan} \triangleq Scenario_{csan} \ \wedge \bigwedge_{\mathbb{S} \in SC^{csan}} \neg Enabled_{\mathbb{S}} \ .$$

Evaluating the property of well-formedness is discussed in the next section.

5.1 Well-formedness

Informally speaking, *csan* in not well-formed if there is a subset of transitions that induce a scenario which does not violate well-formedness and there is an enabled sync-

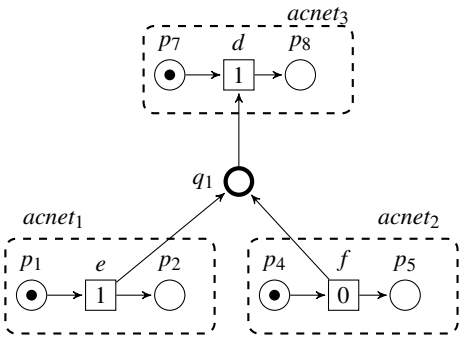

Fig. 10. A communication structured acyclic net which is not well-formed.

cycle which violates well-formedness after its execution and the definition of a scenario is violated as well. The following formula detects that a given *csan* is not well-formed:

$$NotWellFormed_{csan} \triangleq Scenario_{csan} \ \wedge \bigvee_{\mathbb{S}\in SC^{csan}} \big(Enabled_{\mathbb{S}} \ \wedge \bigvee_{u\in \mathrm{pre}_{csan}(\mathrm{post}_{csan}(\mathbb{S}))} in_u\big) .$$

5.2 Non-redundant transitions, marked places, and deadlocked scenarios

The absence of redundant transitions of *csan* can be verified in the same way as for an acyclic net, by checking that, for every $t \in T_{csan}$, the following formula has a satisfying assignment:

$$NonRedundant^{t}_{csan} \triangleq in_t \ \wedge \ Scenario_{csan}$$

The formula for checking whether a place is marked by a scenario is also similar as before:

$$Mark^{csan}_{p} \triangleq \bigvee_{t\in \mathrm{pre}_{csan}(p)} in_t \ \wedge \bigwedge_{u\in \mathrm{post}_{csan}(p)} \neg in_u$$

Moreover, checking whether a set of places is marked by a scenario is achieved by:

$$Reach^{M}_{csan} \triangleq \bigwedge_{p\in M} Scenario_{csan} \ \wedge \ Mark^{csan}_{p}$$

Detecting deadlocked scenarios can then be done using the following formula:

$$DeadlockScenario_{csan} \triangleq MaxScenario_{csan} \ \wedge \bigvee_{p\in P_{csan}\setminus M^{fin}_{csan}} Mark_p$$

Note that it is not required that the buffer places are empty in non-deadlocked markings.

5.3 Backward deterministic CSA-nets

When *csan* is a backward deterministic CSA-net, some formulas defined earlier can be simplified, for example:

$$\begin{aligned} \mathit{Causality}_{csan} &\triangleq \bigwedge_{t \in T_{csan}} \left(in_t \rightarrow \bigwedge_{p \in \mathrm{pre}_{csan}(\mathrm{pre}_{csan}(t))} in_u \right) \\ \mathit{Enabled}_{\mathbb{S}} &\triangleq \bigwedge_{u \in \mathrm{post}_{csan}(\mathrm{pre}_{csan}(\mathbb{S}))} \neg in_u \ \wedge \ \bigwedge_{u \in \mathrm{pre}_{csan}(\mathrm{pre}_{csan}(\mathbb{S})) \setminus \mathbb{S}} in_u \end{aligned}$$

Then, the formula of $\mathit{Scenario}_{csan}$ is reduced to:

$$\mathit{Scenario}_{csan} \triangleq \mathit{Causality}_{csan} \wedge \mathit{NoForwardConflict}_{csan}$$

and the formula for marking place p with $\mathrm{pre}_{csan}(p) = \{t\}$ is:

$$\mathit{Mark}_p \triangleq in_t \ \wedge \bigwedge_{u \in \mathrm{post}_{csan}(p)} \neg in_u$$

6 Concluding remarks

We presented fundamentals of SAT-based verification of CSA-nets (we also intend to discuss the acyclicity constraint later on). In the future work, we plan to extend the set of formulas capturing behavioural properties of CSA-nets, and so enhance the applicability of the proposed model checking method.

The ongoing work is concerned with an implementation of formulas developed here in the SONCRAFT tool [11]. This would allow comparisons of the efficiency of the proposed model checking technique with other approaches (note that verification problems considered here are NP-complete; see, e.g., [3]). In particular, a comparison with model checking based on finite prefixes of net unfoldings [4,5] after adapting it to CSA-nets and their step sequence execution semantics. However, even the unfolding of acyclic nets, where the forward and backward conflict are allowed, would in the worst case generate exponential finite prefixes [6]. The method proposed in this paper does not suffer from a similar problem.

Finally, we expect the model checking approach presented in this paper to be amenable to further, relatively straightforward, extensions. In particular, to structured acyclic nets based on the *behavioural structured occurrence nets* [9], where the dynamic behaviour of a concurrent system is represented at different levels of abstraction.

Acknowledgement

The authors acknowledge financial support provided by University of Hafr Al Batin. Also, we would like to thank the reviewers for their insightful comments on the submitted paper.

References

1. Talal Alharbi. Analysing and visualizing big data sets of crime investigations using structured occurrence nets (PhD thesis), 2016.
2. Anirban Bhattacharyya, Bowen Li, and Brian Randell. Time in structured occurrence nets. In Lawrence Cabac, Lars Michael Kristensen, and Heiko Rölke, editors, *Proceedings of the International Workshop on Petri Nets and Software Engineering 2016, Toruń, Poland, June 20-21, 2016*, volume 1591 of *CEUR Workshop Proceedings*, pages 35–55. CEUR-WS.org, 2016.
3. Javier Esparza. Decidability and complexity of Petri net problems - an introduction. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, volume 1491 of *Lecture Notes in Computer Science*, pages 374–428. Springer, 1996.
4. Javier Esparza and Keijo Heljanko. *Unfoldings: A Partial-Order Approach to Model Checking (Monographs in Theoretical Computer Science. An EATCS Series)*. Springer, 2008.
5. Victor Khomenko. Model checking based on prefixes of Petri net unfoldings (PhD thesis), 2003.
6. Victor Khomenko, Alex Kondratyev, Maciej Koutny, and Walter Vogler. Merged processes: a new condensed representation of Petri net behaviour. *Acta Informatica*, 43(5):307–330, 2006.
7. Victor Khomenko, Maciej Koutny, and Alexandre Yakovlev. Detecting state encoding conflicts in STG unfoldings using SAT. *Fundam. Informaticae*, 62(2):221–241, 2004.
8. Jetty Kleijn, Maciej Koutny, and Marta Pietkiewicz-Koutny. Regions of Petri nets with a/sync connections. *Theor. Comput. Sci.*, 454:189–198, 2012.
9. Maciej Koutny and Brian Randell. Structured occurrence nets: A formalism for aiding system failure prevention and analysis techniques. *Fundam. Informaticae*, 97(1-2):41–91, 2009.
10. Bowen Li. Visualisation and analysis of complex behaviours using structured occurrence nets (phd thesis), 2017.
11. Bowen Li, Brian Randell, Anirban Bhattacharyya, Talal Alharbi, and Maciej 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*, pages 70–74. IEEE Computer Society, 2018.
12. Paolo Missier, Brian Randell, and Maciej Koutny. Modelling provenance using structured occurrence networks. In Paul Groth and James Frew, editors, *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*, pages 183–197. Springer, 2012.
13. Ivan Poliakov, Victor Khomenko, and Alexandre Yakovlev. Workcraft - A framework for interpreted graph models. In Giuliana Franceschinis and Karsten Wolf, editors, *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*, pages 333–342. Springer, 2009.
14. César Rodríguez and Stefan Schwoon. Verification of Petri nets with read arcs. In Maciej Koutny and Irek Ulidowski, editors, *CONCUR 2012 - Concurrency Theory - 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings*, volume 7454 of *Lecture Notes in Computer Science*, pages 471–485. Springer, 2012.
15. Workcraft. `https://workcraft.org/`, 2018.