Acta Informatica manuscript No.
(will be inserted by the editor)

B.F. van Dongen · J. Desel ·
W.M.P. van der Aalst

Aggregating Causal Runs into Workflow nets

the date of receipt and acceptance should be inserted later

Abstract This paper provides three algorithms for constructing system nets from sets of partially-ordered causal runs. The three aggregation algorithms differ with respect to the assumptions about the information contained in the causal runs. Specifically, we look at the situations where labels of conditions (i.e. references to places) or events (i.e. references to transitions) are unknown. Since the paper focusses on aggregation in the context of process mining, we solely look at workflow nets, i.e. the class of Petri nets with unique start and end places. The difference of the work presented here and most work on process mining is the assumption that events are logged as partial orders instead of linear traces. Although the work is inspired by applications in the process mining and workflow domains, the results are generic and can be applied in other application domains.

Keywords: Process mining, Petri net Synthesis, Aggregation, Runs, Process nets.

1 Introduction

This paper proposes different approaches to "discover" process models from observing runs, i.e., runs (also known as causal nets or occurrence nets, see[8])

B.F. van Dongen, W.M.P. van der Aalst
Department of Technology Management,
Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
E-mail: {b.f.v.dongen,w.m.p.v.d.aalst}@tm.tue.nl

J. Desel
Lehrstuhl für Angewandte Informatik,
Katholische Universität Eichstätt-Ingolstadt,
D-85071, Eichstätt, Germany.
E-mail: joerg.desel@ku-eichstaett.de

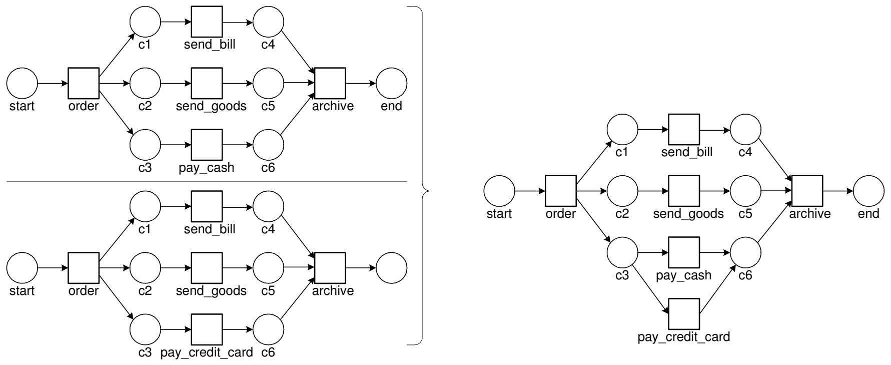
are aggregated into a single Petri net that captures the observed behaviour. This is useful in many domains where processes are studied based on their recorded behaviour. Some examples:

– Discovering administrative processes by following the document flows in the organization with the goal to improve efficiency.
– Auditing processes in organizations in order to make sure that they conform to some predefined rules.
– Constructing enterprise models by observing transaction logs or document flows in enterprise systems such as SAP, Peoplesoft and Oracle.
– Monitoring the flow of SOAP messages between web-services to see how different services interact.
– Observing patient flows in hospitals to improve careflows and to verify medical guidelines.

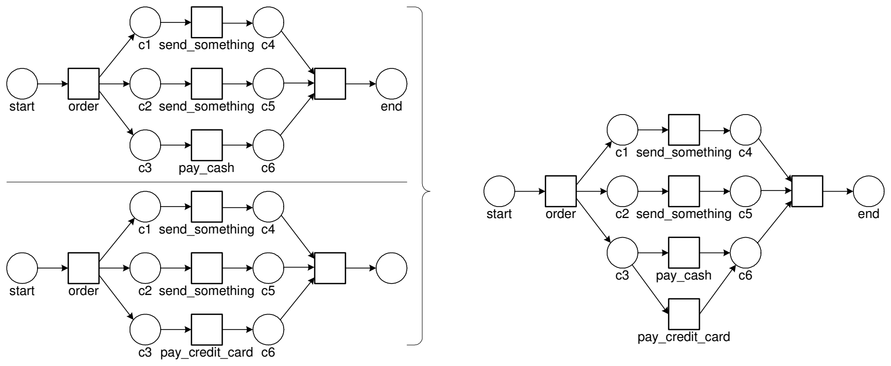
There are many techniques to discover sequential process models based on event logs (also knows as transaction logs, audit trails, etc). Recently, people working on process mining techniques [4] started to tackle situations where processes may be concurrent and the set of observations is incomplete. Note that the set of possible sequences is typically larger than the number of process instances thus making it unrealistic to assume that all possible combinations of behaviour have been observed.

In many applications, the event log is linear, e.g., sorted based on timestamps and an approach based on runs is not applicable. However, there are processes where it is possible to monitor causal dependences (e.g., by analyzing the dataflows). In the examples mentioned before, it is easy to identify situations where activities are causally linked by documents or explicit messages and these could be monitored. Interestingly, we also encountered some systems that actually log behaviour using a representation similar to runs. The ad-hoc workflow management system InConcert of Tibco (formerly Xerox) allows end users to define and modify process instances (e.g., a customer order) while capturing the causal dependencies between the various activities. The representation used directly corresponds to runs. The analysis tool ARIS PPM (Process Performance Monitor) of IDS Scheer can extract runs represented as so-called *instance EPCs* (Event-driven Process Chains) from systems such as SAP R/3 and Staffware. These examples show that in real-life systems and processes runs can be recorded or already are being recorded thus motivating the work presented in this paper.

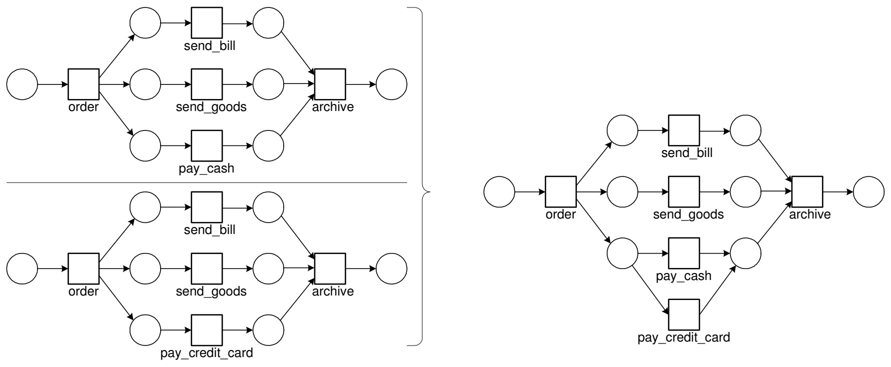
To introduce the main topic of this paper we use the examples shown in Figure 1. The left-hand side of this figure shows several runs. These are the behaviours that have been observed by extracting information from e.g. some enterprise information system. The right-hand side shows the models that we would like to discover by aggregating the runs shown on the left-hand side. Figure 1 (a) shows the easiest situation. Here we assume that in the run all event and condition labels have been recorded in some event log. Note that runs are represented by acyclic Petri nets without any choices, i.e., the unfolding of a Petri net that may contain loops and choices. In Figure 1 (a) there are two runs and it is easy to see that the aggregated model can indeed reproduce these two runs. In this case no other runs are possible. However,



(a) Known event and condition labels.



(b) Known condition labels and unknown or non-unique event labels.



(c) Known event labels and unknown condition labels.

Fig. 1 Example describing the three problems tackled in this paper.

it is important to see that not all possible runs need to be present. In most application domains the number of possible runs is larger than the actual number of process instances, e.g., there may be less patients in a hospital that the number of possible combinations of treatments. Figure 1 (b) describes a more complex problem where not all event labels are recorded or where the same label may refer to different transitions. For example, in Figure 1 (b) the archive event is no longer visible and the two send events (`send_goods` and `send_bill`) cannot be distinguished, since both of them are recorded as `send_something`. Figure 1 (c) illustrates the most challenging problem, i.e., the event labels are given but the condition labels are not recorded at all. Nevertheless, it is clear that the Petri net on the right is the most likely candidate process to exhibit such behaviour.

In this paper we will tackle the problem of aggregating runs into a Petri net that can generate these runs. We will show that it is possible to do this for the three situations depicted in Figure 1.

2 Related Work

The generation of system nets from their causal runs has been investigated before. The first publication on this topic is [19]. Here the basis is assumed to be the set of all runs. These runs are folded, i.e., events representing the occurrence of the same transition are identified, and so are conditions representing a token on the same place. In [9] a similar folding approach is taken, but there the authors start with a set of causal runs, as we do in the present paper. [9] does not present algorithms in details for the aggregation of runs but rather concentrates on correctness criteria for the derived system net.

The problem tackled in this paper is closely related to the so-called synthesis problem of Petri nets (see [10] and [14] for the synthesis of elementary net systems and [6] for more general cases). In this work, the behaviour is given in the form of state graphs (where the events are known but the states are anonymous). In process mining, the observed behaviour is not complete and it is not known, which process executions lead to identical global states. More recently, [18] extracts Petri nets from models which are based on Message Sequence Charts (MSCs), a concept quite similar to causal runs. Less related is the work presented in [15], where a special variant of MSCs is used to generate a system implementation.

In a recent paper [17], regions are defined for partial orders of events representing runs. These regions correspond to places of a Place/Transition net, which can generate these partial orders. In contrast to our work, the considered partial orders are any linearizations of causal orders, i.e., two ordered events can either occur in a sequence (then there is a causal run with a condition "between" the events) or they can occur concurrently. Consequently, conditions representing tokens on places are not considered in these partial orders whereas our approach heavily depends on these conditions.

The research domain *process mining* is relatively new. A complete overview of recent process mining research is beyond the scope of this paper. Therefore, we limit ourselves to a brief introduction to this topic and refer to [4,5] and our website http://www.processmining.org for a more complete overview. The goal of process mining is to extract information about processes from event logs. One of the aspects of process mining focusses on finding a process specification, based on a set of executions of that process. In process mining a process log is taken as a starting point and a variety of algorithms have been proposed to generate a process model based on this log. Typically, such a log is considered to consist of cases (i.e. process instances, for example one insurance claim in a process dealing with insurance claims) and all tasks in each case are totally ordered (typically based on the timestamps). In this paper, we take a different approach. We start by looking at so-called runs. These runs are a *partial* ordering on the tasks within each case. However, in addition to the partial ordering of tasks, we may have information about the state of the system from which the logs originated, i.e. for each event the pre and post conditions are known. This closely relates to the process mining algorithms presented in [11] and [12]. However, in these papers only causal dependencies between events and no state information is assumed to be known.

In this paper, we provide three algorithms for the aggregation of runs. First, we assume we indeed have full knowledge of each event, its preconditions and its postconditions. Then, we assume that we cannot uniquely identify events, i.e. the label of an event may refer to multiple transitions. Finally, we provide an algorithm that assumes less knowledge about pre- and postconditions. However, before we elaborate on our results, we first provide some preliminary definitions that we use throughout the paper.

3 Preliminaries

In this section, we introduce some basic definitions used in the remainder of this paper and formalize the starting point for process mining. In the introduction, we stated that we start with a partial order on a set of events. Typically, a partial order is represented by a graph, and therefore we introduce some concepts related to graphs, such as a complete subgraph and a graph coloring. A graph-coloring is a way to label the nodes of a graph in such a way that no two neighboring nodes (i.e. nodes connected by an edge) have the same color.

Definition 3.1. (Subgraph)
Let $G = (N, E)$ be a graph, i.e. N is the set of nodes and $E \subseteq N \times N$ is the set of edges. Let $N' \subseteq N$. We say that $G' = (N', E \cap (N' \times N'))$ is a subgraph of G.

Definition 3.2. (Undirected path in a graph)
Let $G = (N, E)$ be a directed graph. Let $a \in N$ and $b \in N$. We define an undirected path from a to b in the standard way as a sequence of nodes denoted by $< n_1, \ldots, n_k >$ with $k \geq 1$ such that $n_1 = a$ and $n_k = b$ and $\forall_{i \in \{1 \ldots k-1\}} ((n_i, n_{i+1}) \in E \vee (n_{i+1}, n_i) \in E)$.

Definition 3.3. (Complete graph)
Let $G = (N, E)$ be a graph, i.e. N is the set of nodes and $E \subseteq N \times N$ is the set of edges. We say that G is a *complete* graph if and only if $E = (N \times N)$.

Definition 3.4. (Connected graph)
Let $G = (N, E)$ be a graph. We say that G is a *connected* graph if and only if for all $n_1, n_2 \in N$ holds that there is an undirected path from n_1 to n_2. A set of vertices $N' \subseteq N$ generates a *maximal connected subgraph* if it is a maximal set of vertices generating a connected subgraph.

Definition 3.5. (Graph coloring)
Let $G = (N, E)$ be a graph. Let μ be a set of colors. A function $f : N \to \mu$ is a coloring function if and only if for all $(n_1, n_2) \in E$, either $n_1 = n_2$ or $f(n_1) \neq f(n_2)$.

Lemma 3.6. (Colorings on subgraphs can be combined)
Let $G = (N, E)$ be a graph and $E_1, E_2 \subseteq E$, such that $E_1 \cup E_2 = E$. Furthermore, let $f : N \to \mu$ be a coloring function on the graph (N, E_1) as well as a coloring function on the graph (N, E_2). Then f is also a coloring function on G.

Proof Let $(n_1, n_2) \in E$ and $n_1 \neq n_2$. Since $E = E_1 \cup E_2$, we either have $(n_1, n_2) \in E_1$ or $(n_1, n_2) \in E_2$. Since f is a coloring function on both (N, E_1) and (N, E_2), $f(n_1) \neq f(n_2)$. $\square$

In graphs, we would like to be able to talk about predecessors and successors of nodes. Therefore, we introduce a special notation for that.

Definition 3.7. (Pre-set and Post-set)
Let $G = (N, E)$ be a directed graph and let $n \in N$. We define $\overset{G}{\bullet} n = \{m \in N \mid (m, n) \in E\}$ as the pre-set and $n \overset{G}{\bullet} = \{m \in N \mid (n, m) \in E\}$ as the post-set of n with respect to the graph G. If the context is clear, the superscript G may be omitted, resulting in $\bullet n$ and $n \bullet$.

As stated in the introduction, our starting point is not only a partial order of events within a case, but also information about the state of a case. Since we want to be able to represent both events and states, Petri nets provide a natural basis for our approach. In this paper, we use a specific class of Petri nets, called Place/Transition nets.

Definition 3.8. (Place/Transition net)
$N = (P, T, F, M_0)$ is a marked place/transition net (or P/T-net) if:

- P is a finite set of places,
- T is a finite, non-empty set of transitions, such that $P \cap T = \emptyset$ and $T \neq \emptyset$,
- $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation of the net,
- $M_0 : P \to \mathbb{N}$ represents the initial marking of the net, where a marking is a bag over the set of places P, i.e. it is a function from P to the natural numbers $\mathbb{N}$.

Note that any P/T net $N = (P, T, F, M_0)$ defines a directed graph $((P \cup T), F)$. In this paper, we restrict ourselves to P/T nets where for all transitions t holds that $\bullet t \neq \emptyset$ and $t \bullet \neq \emptyset$.

Definition 3.9. (Bag Notations)
Let S be a set. A *bag* over S is a function from S to the natural numbers $\mathbb{N}$. We use square brackets for the enumeration of the elements of a bag, e.g. $[2a, b, 3c]$ denotes the bag with two a's, one b, and three c's. The sum of two bags ($X \uplus Y$), the presence of an element in a bag ($a \in X$), and the notion of subbags ($X \leq Y$) are defined in a straightforward way and they can handle a mixture of sets and bags. Furthermore, $\biguplus_{a \in A} \big(f(a)\big)$ denotes the sum over the bags that are a result of function f.

Petri nets specify processes. The behaviour of a Petri net is given in terms of causal nets, representing process instances (i.e. cases). Therefore, we introduce some concepts taken from [8]. First, we introduce the notion of a causal net, this is a specification of one process instance of some process specification.

Definition 3.10. (Causal net)
A P/T net (C, E, K, S_0) is called a causal net, if and only if:

- For every place $c \in C$ holds that $| \bullet c| \leq 1$ and $|c \bullet| \leq 1$,
- The transitive closure of K is irreflexive, i.e. it is a partial order on $C \cup E$,
- For all places $c \in C$ holds that $S_0(c) = 1$ if $\bullet c = \emptyset$ and $S_0(c) = 0$ if $\bullet c \neq \emptyset$.

In causal nets, we refer to places as *conditions* and to transitions as *events*.

A causal net is typically generated by a process (remember that it represents an instance of a process). Therefore, each transition and place in a causal net should refer to a transition and place of some process specification respectively. This reference is made by mapping the conditions and events of a causal net onto places and transitions of a Petri net. We call the combination of a causal net and such a mapping a *run*.

Definition 3.11. (Run)
A run (N, α, β) of a Place/Transition net (P, T, F, M_0) is a causal net $N = (C, E, K, S_0)$, together with two mappings $\alpha : C \rightarrow P$ and $\beta : E \rightarrow T$, such that:

- For each event (transition) $e \in E$, the mapping α induces a bijection from $\bullet e$ to $\bullet \beta(e)$ and a bijection from $e \bullet$ to $\beta(e) \bullet$
- $\alpha(S_0) = M_0$ where α is generalized to markings by $\alpha : (C \rightarrow \mathbb{N}) \rightarrow (P \rightarrow \mathbb{N})$, such that $\alpha(S_0)(p) = \sum_{c | \alpha(c) = p} S_0(c)$.

To avoid confusion, the P/T net (P, T, F, M_0) is called *system net* in the sequel.

When we take a system net and take all possible runs for that system net, we say that we have the causal behaviour of that system net.

Definition 3.12. (Causal behaviour)
The causal behaviour of a P/T net (P, T, F, M_0) is defined as its set of runs.

In this paper, we take a set of runs as a starting point. From these runs, we generate a system net describing the behaviour of all individual runs. Note that we do not assume to have all runs as a starting point. The causal behaviour of a P/T net is not necessarily a *finite* set of runs, therefore we cannot assume that we have all runs. Furthermore, in process mining we have to deal with logs which typically show representative examples of possible behaviour, rather than all possible behaviour.

4 Aggregation of Runs

Generally, when looking at runs, it is assumed that the system net is used to generate these runs. However, what if this is not the case? In this section, we will introduce an approach that takes a set of runs as a starting point. From this set of runs, a system net is constructed. However, to construct a system net, we need to find a mapping from all the events and conditions in the causal nets to the transitions and places in the system net. From Definition 3.11, we know that there should exist a bijection between all places in the pre- or post-set of an event in some causal net, and the pre- or post-set of a transition in a system net. *Therefore, two conditions belonging to the pre- or post-set of an event should not be mapped onto the same label.* This restriction is in fact merely another way to express the fact that our P/T nets do not allow for more than one edge between a place and a transition or vice versa. More general, we define a labelling function on the nodes of a graph as a function that does not give the same label to two nodes that have a common successor, or a common predecessor.

Definition 4.1. (Labelling function)
Let μ be a set of labels. Let $G = (N, E)$ be a graph. Let $R = \{(n_1, n_2) \subseteq N \times N \mid n_1 \overset{G}{\bullet} \cap n_2 \overset{G}{\bullet} \neq \emptyset \vee \overset{G}{\bullet} n_1 \cap \overset{G}{\bullet} n_2 \neq \emptyset\}$. We define $f : N \rightarrow \mu$ to be a *labelling function* if and only if f is a coloring function on the graph (N, R).

In this paper, we focus on the aggregation of runs that originate from a Petri net with a clearly defined starting state and completion state, i.e. processes that describe a life-cycle of some case (e.g. a customer order, a payment transaction, a job application, a production order, etc.). Note that this assumption is very natural in the context of workflow management systems [1,3,13]. However, it applies to many other domains where processes are instantiated for specific cases. Hence, we will limit ourselves to a special class of Petri nets, namely Workflow-nets or WF-nets. A WF-net is defined as follows:

Definition 4.2. (Workflow nets)
Let $N = (P, T, F, M_0)$ be a P/T-net. N is a *workflow net* (WF-net) if and only if:

1. *object creation*: P contains an input place p_{ini} such that $\bullet p_{ini} = \emptyset$,
2. *object completion*: P contains an output place p_{out} such that $p_{out}\bullet = \emptyset$,
3. *connectedness*: there is a path from p_{ini} to every node and from every node to p_{out},
4. *initial marking*: $M_0 = [p_{ini}]$, i.e. the initial marking marks only the input place p_{ini}.

A WF-net is a Petri net with one input place. When looking at runs of a WF-net, we can therefore conclude that there is always exactly one condition containing a token and all the other conditions do not contain tokens. A set of causal nets fulfilling this condition is called a *causal set*.

Definition 4.3. (Causal set)
Let $n \in \mathbb{N}$ and let $\Phi = \{(C_i, E_i, K_i, S_i) \mid 0 \leq i < n\}$ be a set of causal nets. We call this set a *causal set* if and only if for all $0 \leq i < n$ holds that:

- All sets C_i, E_i, K_i, C_j, E_j and K_j are disjoint.
- $\sum_{c \in C_i} S_i(c) = 1$, i.e. there is exactly one condition with an empty preset,
- For all $e \in E_i$ with $c \in \bullet e$ such that $S_i(c) = 1$ holds that $\{c\} = \bullet e$,i.e. each event in the postset of one of these conditions has only this initially marked condition in its preset,
- For all $c \in C$ with $c\bullet = \emptyset$ holds that $\forall_{e \in \bullet c} e\bullet = \{c\}$, i.e. each event in the preset of a condition with empty postset (representing a token on the place p_{out}) has only this condition in its postset.

The concept of constructing a system net from a causal set is called *aggregation*. This concept can be applied if we assume that each causal net in the given set can be called a run of some system net. From Definition 3.11, we know that we need two mappings α and β satisfying the two properties mentioned. Using the definition of a system net and the relation between system nets and runs, we can conclude that any aggregation algorithm should have the following functionality:

- It should provide the set of places P of the system net.
- It should provide the set of transitions T of the system net.
- It should provide the flow relation F of the system net.
- It should provide the initial marking M_0 of the system net.
- For each causal net in the causal set, it should provide the mappings $\alpha_i : C_i \rightarrow P$ and $\beta_i : E_i \rightarrow T$, in such a way that for all causal nets, $\alpha_i(S_i)$ is the same (i.e. they have the same initial marking) and they induce bijections between pre- and post-sets of events and their corresponding transitions.

From Definition 3.11, we know that each event that appears in a causal net has a corresponding transition in the original system net. More important, however, is the fact that bijections exist between the pre- and post-sets of this event and the corresponding transitions. In order to express this in terms of labelling functions of causal nets, we formalize this using the notion of *transition equivalence*.

Definition 4.4. (Transition equivalence)
Let μ, ν be two sets of labels, such that $\mu \cap \nu = \emptyset$. Let $\Phi = \{N_i = (C_i, E_i, K_i, S_i) \mid 0 \leq i < n\}$ be a causal set, and let $\Psi = \{(\alpha_i : C_i \rightarrow \mu, \beta_i : E_i \rightarrow \nu) \mid 0 \leq i < n\}$ be a set of labelling functions of (C_i, E_i, K_i, S_i). We define (Φ, Ψ) to *respect transition equivalence* if and only if for each $e_i \in E_i$ and $e_j \in E_j$ with $\beta_i(e_i) = \beta_j(e_j)$ the following holds:

- for each $(p_i, e_i) \in K_i$ we have a $(p_j, e_j) \in K_j$ such that $\alpha_i(p_i) = \alpha_j(p_j)$,
- for each $(e_i, p_i) \in K_i$ we have a $(e_j, p_j) \in K_j$ such that $\alpha_i(p_i) = \alpha_j(p_j)$.

In the remainder of this paper, we will introduce three aggregation algorithms.

5 Aggregation with Known Labels

In this section, we present an aggregation algorithm that assumes that we know all mapping functions, and that these mapping functions adhere to the definition of a run. To illustrate the aggregation process, we make use of a running example throughout the paper. Consider Figure 2 where four part of runs are shown. It is important to realize that these are not complete runs, but merely parts of runs. We do assume however that the events A,B,C,D,E,F and G do not appear in any other part of each run.

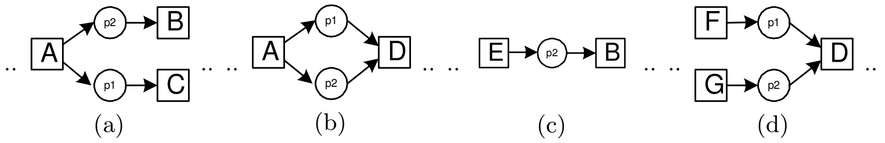


Fig. 2 Four examples of parts of runs.

Our first aggregation algorithm is called the *ALK* aggregation algorithm (short for "All Labels Known"). This algorithm assumes all information such as presented in Figure 2 to be present, i.e. it assumes known labels for events and known labels for conditions. These labels refer to concrete transitions and places in the aggregated system net.

Definition 5.1. (*ALK* aggregation algorithm)
Let μ, ν be two sets of labels, such that $\mu \cap \nu = \emptyset$. Let Φ be a causal set of size n and let $(C_i, E_i, K_i, S_i) \in \Phi$ correspond to a causal net with $0 \leq i < n$. Furthermore, let $\{(\alpha_i : C_i \rightarrow \mu, \beta_i : E_i \rightarrow \nu) \mid 0 \leq i < n\}$ be a set of labelling functions respecting transition equivalence, such that for all causal nets $\alpha_i(S_i)$ is the same. We construct the system net (P, T, F, M_0) belonging to these runs as follows:

- $P = \bigcup_{0 \leq i < n} rng(\alpha_i)$ is the set of places (note that $P \subseteq \mu)$[1],
- $T = \bigcup_{0 \leq i < n} rng(\beta_i)$ is the set of transitions (note that $T \subseteq \nu$),
- $F = \bigcup_{0 \leq i < n} \{(\alpha_i(c), \beta_i(e)) \in P \times T \mid (c, e) \in K_i \cap (C_i \times E_i)\} \cup$
  $\bigcup_{0 \leq i < n} \{(\beta_i(e), \alpha_i(c)) \in T \times P \mid (e, c) \in K_i \cap (E_i \times C_i)\}$
  is the flow relation,
- $M_0 = \alpha_0(S_0)$ is the initial marking.

The result of the *ALK* aggregation algorithm from Definition 5.1 for the parts presented in Figure 2 is shown in Figure 3. Another example is given in Figure 1(a).

It is easy to see that the aggregated net shown in Figure 3 can actually generate the runs shown in Figure 2. We show that this is always the case after applying the *ALK* aggregation algorithm.

[1] With rng we denote the range of a function, i.e. $rng(f) = \{f(x) \mid x \in dom(f)\}$

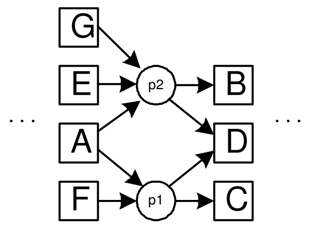


Fig. 3 The aggregated Petri net.

Property 5.2. (*ALK* algorithm is correct)
To prove the *ALK* algorithm is correct, we show that for $0 \leq i < n$ holds that $N_i = (C_i, E_i, K_i, S_i)$, (N_i, α_i, β_i) is indeed a run of $\sigma = (P, T, F, M_0)$, (i.e. the requirements stated in Definition 3.11 are fulfilled).

Proof Since we assumed that all $N_i = (C_i, E_i, K_i, S_i)$ are causal nets, we need to prove the following for each α_i and β_i.

1. α_i is a function from C_i onto P. This trivially follows from Definition 5.1.
2. β_i is a function from E_i onto T. This trivially follows from Definition 5.1.
3. $\alpha_i(S_i) = M_0$ holds by definition, since it holds for S_0 and for all causal nets, $\alpha_i(S_i)$ is the same.
4. For each event $e \in E_i$, the mapping α_i induces a bijection from $\bullet e$ to $\bullet \beta_i(e)$ and a bijection from $e\bullet$ to $\beta_i(e)\bullet$.

   Let $e \in E_i$. We start by showing that $\alpha_i(\overset{N_i}{\bullet} e) = \overset{\sigma}{\bullet} \beta_i(e)$ and $\alpha_i(e \overset{N_i}{\bullet}) = \beta_i(e) \overset{\sigma}{\bullet}$. Assume $p \in \alpha_i(\overset{N_i}{\bullet} e) \setminus \overset{\sigma}{\bullet} \beta_i(e)$, i.e. there exists a $c \in C_i$ with $(c, e) \in K_i$, such that $p = \alpha_i(c)$, $\beta_i(e) = t$ and $(p, t) \notin F$. Clearly this contradicts with the definition of F in Definition 5.1. Now assume $p \in \overset{\sigma}{\bullet} \beta_i(e) \setminus \alpha_i(\overset{N_i}{\bullet} e)$, i.e. there is a $(p, t) \in F$ such that $\beta_i(e) = t$ and there is no $c \in C_i$ with $\alpha_i(c) = p$, such that $(c, e) \in K_i$. If this is the case in all causal nets for $0 \leq i < n$, then this leads to a contradiction since this would imply $(p, t) \notin F$ (cf. Definition of F in Definition 5.1). If there is a $0 \leq j < n$, such that $(c', e') \in K_j$ with $\beta_j(e') = t$ and $\alpha_j(c') = p$, then there has to be a $c \in C_i$ such that $(c, e) \in K_i$, since $\alpha_i(\overset{N_i}{\bullet} e) = \alpha_j(\overset{N_i}{\bullet} e')$ (cf. Definition 4.4). Combined with the fact that α_i and β_i are labelling functions, $\alpha_i(\overset{N_i}{\bullet} e) = \overset{\sigma}{\bullet} \beta_i(e)$ and $\alpha_i(e \overset{N_i}{\bullet}) = \beta_i(e) \overset{\sigma}{\bullet}$ yields the bijection. Similar arguments apply for the post-set.

□

The aggregation algorithm presented in Definition 5.1 is a rather trivial aggregation over a set of runs. However, it is assumed that the mapping functions α_i and β_i are known for each causal net, in such a way that Definition 3.11 is followed. Furthermore, we assume two sets of labels μ and ν to be known. However, when applying these techniques in the context of process mining, it is often not realistic to assume that all of these are present. Therefore, in the remainder of this paper, we relax some of these assumptions to obtain more usable process mining algorithms.

6 Aggregation with Duplicate or Missing Transition Labels

In this section, we will assume that the causal set used to generate the system net and the labelling functions, does not respect transition equivalence. We introduce an algorithm to change the labelling function for events in such a way that this property holds again. In the domain of process mining, the problem of so-called "duplicate transitions" (i.e. several transitions with the same label) is well-known (cf. [2,7,16]). Therefore, there is a need for algorithms to find out which events actually belong to which transition. In this section, we assume that we have causal nets with labelling functions, where some events have the same label, even though they may refer to different transitions, which is shown in Figure 4. Note that this figure is similar to Figure 2, except that we now labelled events F and G both with a new label X.

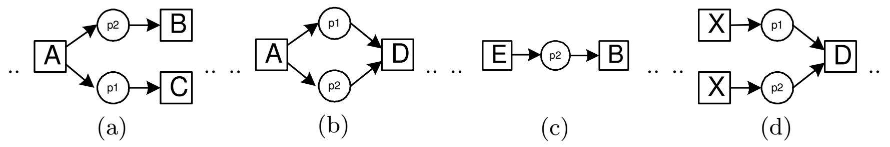


Fig. 4 Four examples of parts of runs.

In terms of an aggregation algorithm, the problem of duplicate labels translates to the situation where the property of transition equivalence (Property 4.4) is not satisfied. Since the aggregation algorithm presented in the previous section only works if this property holds, we provide an algorithm to redefine the labelling functions for events. Furthermore, we will prove that after applying this algorithm, the desired property holds.

Definition 6.1. (Relabelling algorithm)
Let μ, ν be two sets of labels, such that $\mu \cap \nu = \emptyset$. Let $\Phi = \{N_i \mid 0 \leq i < n \wedge N_i = (C_i, E_i, K_i, S_i)\}$ be a causal set, and let $\Psi = \{(\alpha_i : C_i \to \mu, \beta_i : E_i \to \nu) \mid 0 \leq i < n\}$ be a set of labelling functions in (C_i, E_i, K_i, S_i), such that $\alpha_i(S_i)$ is the same for all causal nets. Furthermore, assume that μ and ν are minimal, i.e. $\bigcup_{0 \leq i < n} rng(\alpha_i) = \mu$ and $\bigcup_{0 \leq i < n} rng(\beta_i) = \nu$. Let $E^\star = \bigcup_{0 \leq i < n} E_i$ be the set of all events in the causal set. We define the relabelling algorithm as follows:

1. Define $\bowtie \subseteq E^\star \times E^\star$ as an equivalence relation on the elements of $E^\star$ in such a way that $e_i \bowtie e_j$ with $e_i \in E_i$ and $e_j \in E_j$ if and only if $\beta_i(e_i) = \beta_j(e_j)$, $\alpha_i(\overset{N_i}{\bullet} e_i) = \alpha_j(\overset{N_i}{\bullet} e_j)$, and $\alpha_i(e_i \overset{N_i}{\bullet}) = \alpha_j(e_j \overset{N_i}{\bullet})$.
2. For each $e \in E^\star$, we say $eqvl(e) = \{e' \in E^\star \mid e \bowtie e'\}$.
3. Let ν' be the set of equivalence classes of $\bowtie$, i.e. $\nu' = \{eqvl(e) \mid e \in E^\star\}$.
4. For all causal nets (C_i, E_i, K_i, S_i) and labelling functions α_i, define a labelling function $\beta'_i : E_i \to \nu'$ such that for an event e_i, $\beta'_i(e_i) = eqvl(e_i)$, i.e. it returns the equivalence class of $\bowtie$ containing e_i.

After re-labelling the events, the part of the run shown in Figure 4(d) is relabelled to include the pre- and postconditions. Figure 5(a) shows the

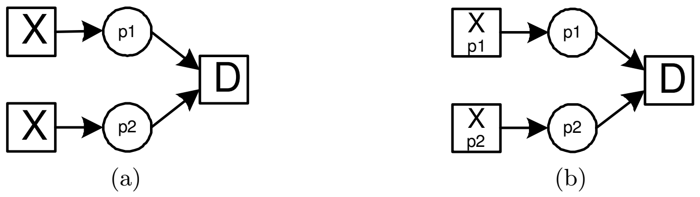


Fig. 5 The original and relabelled part of Figure 4(d).

fragment before relabelling, whereas Figure 5(b) shows the fragment after relabelling. However, in Figure 5(b) we only show the relabelling with respect to the post conditions.

Applying the *ALK* algorithm of Definition 5.1 to the relabelled runs yields the result as shown in Figure 6. Note that we do not show the ν' labels explicitly, i.e. B refers to the equivalence class of events labelled B.

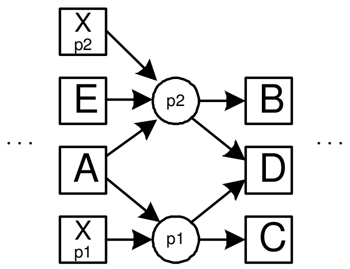


Fig. 6 A part of the aggregated net.

What remains to be shown is that our algorithm does not only work for our small running example, but also in the general case. Note that the only difference between the assumptions in Definition 5.1 and Definition 6.1 is the requirement with respect to transition equivalence. Therefore, if suffices to prove that after applying the relabelling algorithm on a causal set, we can establish transition equivalence.

Property 6.2. (Transition equivalence holds after relabelling) Let μ, ν be two sets of labels, such that $\mu \cap \nu = \emptyset$. Let $\Phi = \{(C_i, E_i, K_i, S_i) \mid 0 \leq i < n\}$ be a causal set, and let $\Psi = \{(\alpha_i : C_i \to \mu, \beta_i : E_i \to \nu) \mid 0 \leq i < n\}$ be a set of labelling functions in (C_i, E_i, K_i, S_i), such that $\alpha_i(S_i)$ is the same for all causal nets. After applying the relabelling algorithm, the property of transition equivalence holds for (Φ, Ψ'), with $\Psi' = \{(\alpha_i : C_i \to \mu, \beta'_i : E_i \to \nu') \mid 0 \leq i < n\}$, and β'_i as defined in Definition 6.1.

Proof We prove that Property 4.4 holds for (Φ, Ψ') after applying the relabelling function. Assume (C_i, E_i, K_i, S_i) and (C_j, E_j, K_j, S_j) are two causal nets from Φ. The new function β'_i is indeed a function, since for each event

$e_i \in E_i$ there exists exactly one equivalence class containing e_i. Furthermore, let $e_i \in E_i$ and $e_j \in E_j$, such that $\beta'_i(e_i) = \beta'_j(e_j)$. We know that $e_i \bowtie e_j$ and from the definition of $\bowtie$, we know that $\alpha_i(\bullet e_i) = \alpha_j(\bullet e_j)$ and $\alpha_i(e_i\bullet) = \alpha_j(e_j\bullet)$, which directly implies transition equivalence. $\square$

The algorithm presented above is capable of finding events that have the same label, but correspond to different transitions in the system net. When *no transition labels are known at all*, it can be applied to *find all transition labels*, by using an initial $\nu = \{\tau\}$ and initial mapping functions β_i, mapping everything onto τ. However, in that case, no distinction can be made between events that have the same pre- and post set, but should have different labels. After applying this relabelling algorithm, the aggregation algorithm of Section 5 can be used to find the system net belonging to the given causal nets.

7 Aggregation with Unknown Place Labels

In Section 6, we have shown a way to identify the transitions in a system net, based on the labels of events in causal nets. However, what if condition labels are not known? In this section, we take one step back. We assume all events to refer to the correct transition and we try to identify the labels of conditions. We introduce an algorithm to aggregate causal nets to a system net, and we prove that the original causal nets are indeed runs of that system net. In Figure 7, we again show our small example of the aggregation problem, only this time there are no labels for conditions $p1$ and $p2$, which we did have in figures 2 and 4.

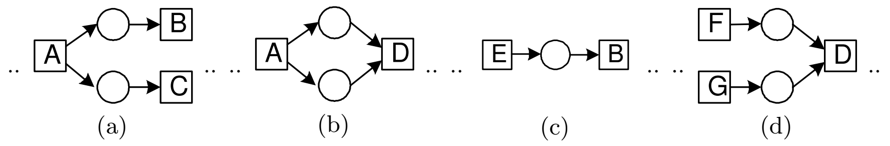


Fig. 7 Four examples of parts of runs.

Consider the four runs of Figure 7. Remember that they are *parts* of causal nets, in such a way that the tasks A, B, C, D, E, F and G do not appear in any other way in another causal net. In contrast to the algorithms presented in previous sections, we cannot always derive a unique aggregated system net for causal nets if we do not have labels for the conditions. Instead, we define an *aggregation class*, describing a class of WF-nets that could have generated these causal nets. Table 1 shows some requirements all WF-nets in the aggregation class of our example should satisfy.

The information in Table 1 is derived using the concept of a segment, which can be considered to be the context of a condition in a causal net.

Table 1 Information derived from runs shown in Figure 7.

Fragment	Conclusions
Fig. 7 (a)	$A\bullet = \bullet B \uplus \bullet C$
Fig. 7 (b)	$A\bullet = \bullet D$
Fig. 7 (c)	$E\bullet = \bullet B$
Fig. 7 (d)	$F\bullet \uplus G\bullet = \bullet D$

Definition 7.1. (Segment)
Let $N = (C, E, K, S_0)$ be a causal net. Furthermore, let $N' = (C', E_{in}, E_{out})$ be such that $C' \subseteq C$, $E_{in} \cup E_{out} \subseteq E$ and $E_{in} \neq \emptyset$ and $E_{out} \neq \emptyset$. We call N' a *segment* if and only if for all $c \in C'$ holds that $\bullet c \subseteq E_{in}$ and $c\bullet \subseteq E_{out}$ and for all $e \in E_{in}$ holds that $\bullet e \cap C' = \emptyset$ and $e\bullet \subseteq C'$ and for all $e \in E_{out}$ holds that $\bullet e \subseteq C'$ and $e\bullet \cap C' = \emptyset$. Furthermore, the subgraph of N made up by $C' \cup E_{in} \cup E_{out}$ is connected. We call the events in E_{in} the input events and the events in E_{out} the output events.

For the fragments of Figure 7, it is easy to see that each of them contains only one segment, where the input events are the events on the left hand side and the output events are the events on the right hand side. The meaning of a segment is as follows. If we have a run and a segment in that run, then we know that after each of the events in the input set of the segment occurred, all the events in the output set occurred in the execution represented by this run. This translates directly to a marking in a system net, since the occurrence of a set of transitions would lead to some marking (i.e. a bag over places), which enables another set of transitions. Furthermore, each transition only produces one token in each output place. Combining this leads to the fact that for each segment in a causal net the bag of places following the transitions corresponding to the input events of the segment should be the same as the bag of places preceding the transitions corresponding to the output set of events, as indicated in Table 1.

Clearly, when looking only at these fragments, what we are looking for are the places that should be put between tasks A, E, F and G on the one hand, and B, C and D on the other hand. Therefore, we only focus on this part of the causal nets. For this specific example, there are two possibilities, both of which are equally correct, namely the two WF-net fragments shown in Figure 8.

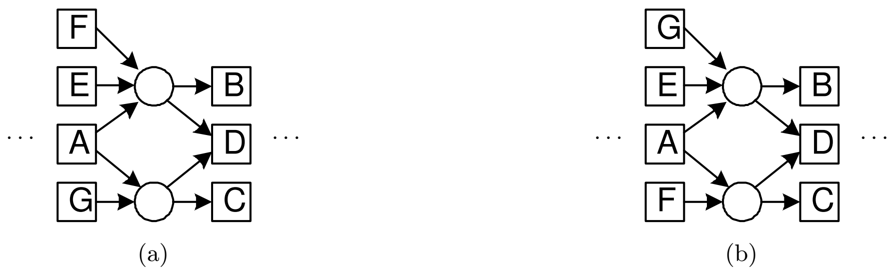


Fig. 8 Two part of aggregated nets.

From the small example, we have seen that it is possible to take a set of causal nets without labels for any of the conditions (but with labels for all the events) and to define a class of WF-nets that could be system nets of the causal nets. In the remainder of this section, we show that this is indeed possible for all causal set s. For this, we first introduce the *NCL* algorithm.

7.1 *NCL* Algorithm

Before actually presenting the *NCL* algorithm (which stands for "No Condition Labels"), we first take a look at a more intuitive example. Consider Figure 9, where we present three causal nets, each of which corresponds to a paper review process. In the first causal net, three reviewers are invited to review the paper and after the three reviews are received, the paper is accepted. In the second causal net, only two reviews are received (the third one is not received on time), but the paper is rejected nonetheless (apparently the two reviewers that replied rejected the paper). In the third example only one review is received in time, and therefore an additional reviewer is invited, which hands in his review in time, but does not accept the paper.

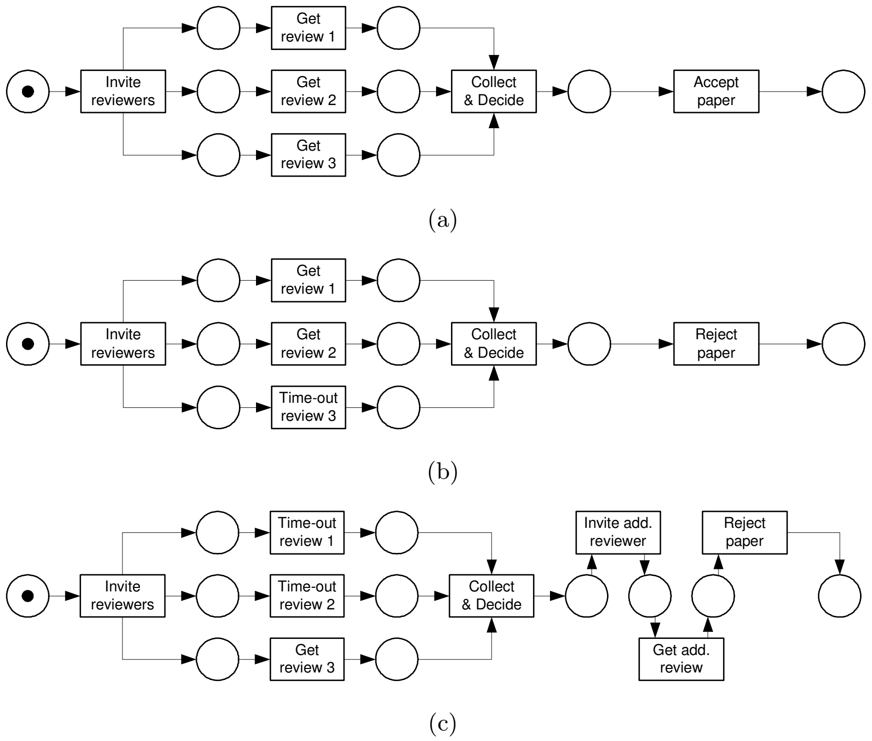


Fig. 9 Three causal nets of a review process of a paper.

As we stated before, we define an aggregation class of a causal set, that contains all WF-nets that are capable of generating the causal nets in the causal set. The information needed for this aggregation class comes directly from the causal nets, using segments. In Table 2, we present the conclusions we can draw based on the three causal nets. Note that in this table, just like in Table 1, we consider *bags* of pre- and post-sets of transitions in the aggregation class. The information in this table is obtained from the causal nets in the following way. Consider for example Figure 9(a), where `invite reviewers` is followed by `Get review 1`, `Get review 2` and `Get review 3`. This implies that the bag of output places of `invite reviewers`, should be the same as the sum over the bags of the input places of `Get review 1`, `Get review 2` and `Get review 3`.

Consider the information presented in Table 2. We formalize the conclusions sketched there in the concept of an aggregation class.

Definition 7.2. (Aggregation class)
Let $\Phi = \{(C_i, E_i, K_i, S_i) \mid 0 \leq i < n\}$ be a causal set, and let $\sigma = (P, T, F, M_0)$ be a marked WF-net. For each causal net, let $\beta_i : E_i \rightarrow T$ be a mapping from the events of that causal net to T, such that β_i is a labelling function for (C_i, E_i, K_i, S_i). We define A_Φ, the *aggregation class* of Φ, as the set of all pairs $(\sigma, \mathcal{B})$, such that the following conditions are satisfied:

1. $T = \bigcup_{0 \leq i < n} Rng(\beta_i)$ is the set of transitions, i.e. each transition appears as an event at least once in some causal net,
2. $\mathcal{B}$ is the set of all labelling functions, i.e. $\mathcal{B} = \{\beta_i \mid 0 \leq i < n\}$. We use $\beta_i \in \mathcal{B}$ to denote the labelling function for events belonging to $(C_i, E_i, K_i, S_i) \in \Phi$.
3. For all $p \in P$ holds that $\overset{\sigma}{\bullet}p \cup p\overset{\sigma}{\bullet} \neq \emptyset$,
4. $M_0 = [p_{ini}]$ and $\overset{\sigma}{\bullet}p_{ini} = \emptyset$,
5. For each causal net $\gamma = (C_i, E_i, K_i, S_i)$, with $e \in E_i$ and $\beta_i(e) = t$ holds that if $S_i(\overset{\gamma}{\bullet}e) = 1$ then $p_{ini} \in \overset{\sigma}{\bullet}t$,
6. For each causal net $\gamma = (C_i, E_i, K_i, S_i)$, with $e \in E_i$ and $\beta_i(e) = t$ holds that $|t\overset{\sigma}{\bullet}| = |e\overset{\gamma}{\bullet}|$ and $|\overset{\sigma}{\bullet}t| = |\overset{\gamma}{\bullet}e|$,
7. For each causal net $\gamma = (C_i, E_i, K_i, S_i)$, with $e \in E_i$, $\beta_i(e) = t$ and $T' \subseteq T$ holds that $|t\overset{\sigma}{\bullet} \cap \bigcup_{t' \in T'}(\overset{\sigma}{\bullet}t')| \geq \sum_{e' \in E_i, \beta(e') \in T'} |e\overset{\gamma}{\bullet} \cap \overset{\gamma}{\bullet}e'|$,
8. For each causal net $\gamma = (C_i, E_i, K_i, S_i)$, with $e \in E_i$, $\beta_i(e) = t$ and $T' \subseteq T$ holds that $|\bigcup_{t' \in T'}(t'\overset{\sigma}{\bullet}) \cap \overset{\sigma}{\bullet}t| \geq \sum_{e' \in E_i, \beta(e') \in T'} |e'\overset{\gamma}{\bullet} \cap \overset{\gamma}{\bullet}e|$,
9. For each causal net $\gamma = (C_i, E_i, K_i, S_i)$ and any segment (C'_i, E_{in}, E_{out}) of γ, holds that $\biguplus_{e \in E_{in}} \left(\beta_i(e)\overset{\sigma}{\bullet}\right) = \biguplus_{e \in E_{out}} \left(\overset{\sigma}{\bullet}\beta_i(e)\right)$.

Figure 10 is used to gain more insight into part 9 of Definition 7.2. In the lower causal net of that figure, there is a token travelling from A to D and from B to C. The upper causal net on the other hand only connects A and C. Assuming that these are the only causal nets in which these transitions appear, we know that the condition between A and D and between B and C should represent a token in the same place, since there is a segment $(\{c_1, c_2, c_3\}, \{A, B\}, \{C, D\})$ in the lower causal net and therefore, $A\bullet \uplus B\bullet = \bullet C \uplus \bullet D = [p_1, 2p_2]$.

Table 2 Information derived from review example.

Causal net	Conclusions on transitions in the aggregation class		
Fig. 9 (a)	$\bullet$“Invite reviewers”	$=$	$[p_{ini}]$
	“Invite reviewers”$\bullet$	$=$	$\bullet$“Get review 1” $\uplus$ $\bullet$“Get review 2” $\uplus$ $\bullet$“Get review 3”
	“Get review 1” $\bullet$ $\uplus$ “Get review 2” $\bullet$ $\uplus$ “Get review 3”$\bullet$	$=$	$\bullet$“Collect & Decide”
	“Collect & Decide”$\bullet$	$=$	$\bullet$“Accept paper”
	$\lvert$“Accept paper” $\bullet$ $\rvert$	$=$	1
Fig. 9 (b)	$\bullet$“Invite reviewers”	$=$	$[p_{ini}]$
	“Invite reviewers”$\bullet$	$=$	$\bullet$“Get review 1” $\uplus$ $\bullet$“Get review 2” $\uplus$ $\bullet$“Time-out review 3”
	“Get review 1” $\bullet$ $\uplus$ “Get review 2” $\bullet$ $\uplus$ “Time-out review 3”$\bullet$	$=$	$\bullet$“Collect & Decide”
	“Collect & Decide”$\bullet$	$=$	$\bullet$“Reject paper”
	$\lvert$“Reject paper” $\bullet$ $\rvert$	$=$	1
Fig. 9 (c)	$\bullet$“Invite reviewers”	$=$	$[p_{ini}]$
	“Invite reviewers”$\bullet$	$=$	$\bullet$“Time-out review 1”$\uplus$ $\bullet$“Time-out review 2”$\uplus$ $\bullet$“Get review 3”
	“Time-out review 1”$\bullet$ $\uplus$ “Time-out review 2”$\bullet$ $\uplus$ “Get review 3”$\bullet$	$=$	$\bullet$“Collect & Decide”
	“Collect & Decide”$\bullet$	$=$	$\bullet$“Invite add. reviewer”
	“Invite add. reviewer”$\bullet$	$=$	$\bullet$“Get add. review”
	“Get add. review”$\bullet$	$=$	$\bullet$“Reject paper”
	$\lvert$“Reject paper” $\bullet$ $\rvert$	$=$	1

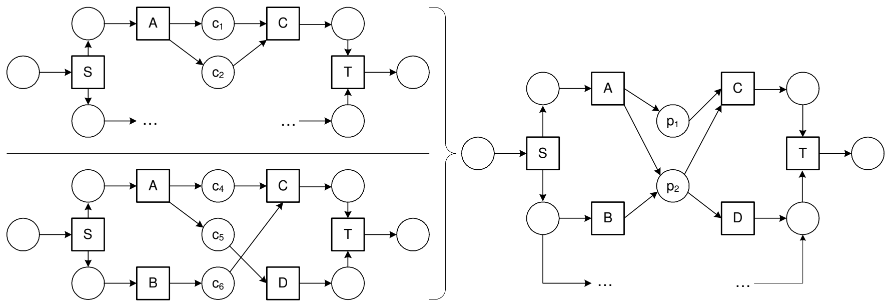


Fig. 10 Example explaining the use of bags.

Definition 7.2 defines a *finite* class of Workflow nets for a causal set. What remains to be given are the conditions under which it is a finite *non-empty* class of Petri nets and the proof that each Petri net with its mappings is indeed a system net for the causal set. To prove this, we first introduce the concept of a *condition graph*.

Definition 7.3. (Condition graph)

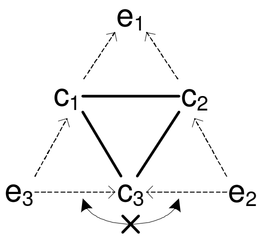
Let $N = (C, E, K, S)$ be a causal net. We define the condition graph $\Delta_N = (C, A)$ as an undirected graph, such that $A = \{(c_1, c_2) \in C \times C \mid \exists_{e \in E}\{c_1, c_2\} \subseteq \overset{N}{\bullet} e \vee \{c_1, c_2\} \subseteq e\overset{N}{\bullet}\}$. Note that since a condition graph is undirected, $(c_1, c_2) \in A$ implies that $(c_2, c_1) \in A$.

We use condition graphs to prove that each Petri net with its mappings in the aggregation class of a causal set is indeed a system net for that causal set. For this, we first introduce some lemmas on these condition graphs that show the relation between condition graphs and causal nets. We start by showing that pre- and post-sets of events correspond to complete subgraphs in the condition graph, i.e. subgraphs where each pair of nodes is connected by an edge.

Lemma 7.4. (Pre- and post sets relate to complete subgraphs in condition graphs)

Let $N = (C, E, K, S)$ be a causal net and $\Delta_N = (C, A)$ its condition graph. We show that for all $e \in E$, holds that Δ_N restricted to $\overset{N}{\bullet} e$ is a complete subgraph and Δ_N restricted to $e\overset{N}{\bullet}$ is a complete subgraph. Furthermore, for each complete subgraph (C', A'), there exists an $e \in E$, such that $C' \subseteq \overset{N}{\bullet} e$ or $C' \subseteq e\overset{N}{\bullet}$.

Proof Since for all $\{c_1, c_2\} \subseteq \overset{N}{\bullet} e$, holds that $(c_1, c_2) \in A$ by definition, the first part is correct. The same applies to $e \overset{N}{\bullet}$.
Now assume (C', A') is a complete subgraph. Assume $\{c_1, c_2\} \subseteq C'$. Since we are looking at a complete subgraph, we know $(c_1, c_2) \in A'$, therefore there exists an $e_1 \in E$, such that $\{c_1, c_2\} \subseteq \overset{N}{\bullet} e_1$ or $\{c_1, c_2\} \subseteq e_1 \overset{N}{\bullet}$. Assume $\{c_1, c_2\} \subseteq \overset{N}{\bullet} e_1$ (The proof is symmetrical for $e_1 \overset{N}{\bullet}$). Now assume $c_3 \in C'$ such that $c_1 \neq c_3$ and $c_2 \neq c_3$. Let $c_3 \not\subseteq \overset{N}{\bullet} e_1$. We show that this leads to a contradiction. Since for all $c \in C$ holds that $|c \overset{N}{\bullet}| \leq 1$, and $\{c1, c2\} \subseteq \overset{N}{\bullet} e_1$, we know that there must be an $e_2 \in E$, such that $\{c_2, c_3\} \subseteq e_2 \overset{N}{\bullet}$.



Similarly, we know that there is an $e_3 \in E$, such that $\{c_1, c_3\} \subseteq e_3 \overset{N}{\bullet}$. However, since $|\overset{N}{\bullet} c_3| \leq 1$, this implies that $e_2 = e_3$ and thus $\{c_1, c_2, c_3\} \subseteq e_2 \overset{N}{\bullet}$. □

Using the fact that each pre- and post-set correspond to a complete subgraph, we can infer that each segment in a causal net corresponds to a connected subgraph in the condition graph, i.e. a subgraph such that there is a path between each two nodes. Furthermore, we show that these connected subgraphs are maximal, i.e. all nodes in the subgraph are only connected to nodes inside the subgraph.

Lemma 7.5. (Segments correspond to maximal connected subgraphs in condition graphs)
Let $N = (C, E, K, S)$ be a causal net and $\Delta_N = (C, A)$ its condition graph. Let (C', E_{in}, E_{out}) be a segment in N. We show that $(C', A \cap (C' \times C'))$ is a maximal connected subgraph of Δ_N.

Proof From Definition 7.1 we know that the graph $(C' \cup E_{in} \cup E_{out}, K \cap ((C' \cup E_{in} \cup E_{out}) \times (C' \cup E_{in} \cup E_{out})))$ is a connected graph. Now, let $c \in C'$ be a condition in the segment and assume that $\{e_{in}\} = \overset{N}{\bullet} c$ and $\{e_{out}\} = c \overset{N}{\bullet}$. From Lemma 7.4, we know that $e_{in} \overset{N}{\bullet}$ and $\overset{N}{\bullet} e_{out}$ make up a complete subgraph in Δ_N and since $c \in \overset{N}{\bullet} e_{out} \cap e_{in} \overset{N}{\bullet}$ that these two complete subgraphs make up a connected subgraph. By induction over the elements of C', it is easy to see that C' makes up a connected subgraph in Δ_N. Therefore, each segment defines a complete subgraph G' in Δ_N. Furthermore, let $G' = (C', A')$ be the connected subgraph of ΔN corresponding to the segment. Let $c \in C \setminus C'$ and assume there exists a $c' \in C'$, such that $(c, c') \in A$. This implies that there is an $e \in E$, such that $\{c, c'\} \subseteq \overset{N}{\bullet} e$ or $\{c, c'\} \subseteq e \overset{N}{\bullet}$. However, this implies that $e \in E_{in}$ or $e \in E_{out}$ either of which imply that $c \in C'$. Therefore, such a c does not exist and G' is maximal. □

At this point, we look at the definitions of Section 3 again. If we assume that we have a system net and the causal behaviour of this system net, we can derive the next lemma using Definition 3.5.

Lemma 7.6. (System nets color condition graphs)
Let $\sigma = (P, T, F, M_0)$ be a system net and (N, α, β) be a run of that system net, with $N = (C, E, K, S_0)$. Furthermore, let $\Delta_N = (C, A)$ be the condition graph of N. The mapping $\alpha : C \to P$ is a coloring function of Δ_N, with the set of colors being P.

Proof Let $n_1, n_2 \in C$ be two nodes in Δ_N with $n_1 \neq n_2$. For α to be a coloring, $\alpha(n_1) \neq \alpha(n_2)$ should hold if $(n_1, n_2) \in A$. Assume $(n_1, n_2) \in A$. This means that there is an $e \in E$ such that $\{n_1, n_2\} \subseteq \overset{N}{\bullet} e$ or $\{n_1, n_2\} \subseteq e \overset{N}{\bullet}$. From Definition 3.11, we know that α induces a bijection from $\overset{N}{\bullet} e$ to $\overset{\sigma}{\bullet} \beta(e)$ and from $e \overset{N}{\bullet}$ to $\beta(e) \overset{\sigma}{\bullet}$. Therefore, $\alpha(n_1) \neq \alpha(n_2)$. □

We have shown that system nets color condition graphs. However, we can go one step further and introduce the concept of a conditional coloring, which is a coloring on the condition graph, such that the coloring function, when applied to the conditions in a causal net, induces local bijections for the input and output sets of events.

Definition 7.7. (Conditional coloring)
Let Φ be a causal set and let A be the aggregation class of Φ. Moreover, let $(\sigma, \mathcal{B}) \in A$, with $\sigma = (P, T, F, M_0)$ and let $N = (C, E, K, S) \in \Phi$ be a causal net and $\Delta_N = (C, A)$ be the condition graph of N. Assume $\alpha : C \to P$ is a function, such that α is a coloring on Δ_N and for all $c \in C$ holds that $\alpha(c) \in \{p \in P \mid \beta(\overset{N}{\bullet} c) \subseteq \overset{\sigma}{\bullet} p \wedge \beta(c \overset{N}{\bullet}) \subseteq p \overset{\sigma}{\bullet}\}$. [2] We call α a conditional coloring of Δ_N.

The concept of a conditional coloring we introduced here is often referred to in mathematics as a list coloring.

Lemma 7.8. (Conditional coloring induces bijections)
Let Φ be a causal set and let A be the aggregation class of Φ, and let $(\sigma, \mathcal{B}) \in A$, with $\sigma = (P, T, F, M_0)$. Let $N = (C, E, K, S) \in \Phi$ be a causal net and $\Delta_N = (C, A)$ be the condition graph of N. Let $\alpha : C \to P$ be a conditional coloring of Δ_N. We show that for all $e \in E$, α induces a bijection from $\overset{N}{\bullet} e$ to $\overset{\sigma}{\bullet} \beta(e)$ and from $e \overset{N}{\bullet}$ to $\beta(e) \overset{\sigma}{\bullet}$.

Proof The requirements stated in Definition 7.2, imply that $|\overset{N}{\bullet} e| = |\overset{\sigma}{\bullet} \beta(e)|$. Furthermore, since Δ_N restricted to $\overset{N}{\bullet} e$ is a complete graph (Lemma 7.4), and α is a coloring function (Lemma 7.6), we know that $|\alpha(\overset{N}{\bullet} e)| = |\overset{N}{\bullet} e|$ Since all elements in $\overset{N}{\bullet} e$ are mapped to different colors. Combining both implies that $|\alpha(\overset{N}{\bullet} e)| = |\overset{\sigma}{\bullet} \beta(e)|$.

For all $c \in \overset{N}{\bullet} e$ holds that $\alpha(c) \in \{p \in P \mid \beta(\overset{N}{\bullet} c) \subseteq \overset{\sigma}{\bullet} p \wedge \beta(c \overset{N}{\bullet}) \subseteq p \overset{\sigma}{\bullet}\}$ (Definition 7.7) and $c \overset{N}{\bullet} = \{e\}$, because N is a causal net we know that $\alpha(c) \in \{p \in P \mid \beta(e) \in p \overset{\sigma}{\bullet}\}$ and thus $\alpha(c) \in \overset{\sigma}{\bullet} \beta(e)$. Since this holds for all $c \in \overset{N}{\bullet} e$, we can conclude that $\alpha(\overset{N}{\bullet} e) \subseteq \overset{\sigma}{\bullet} \beta(e)$. By combining the above, we can conclude that $\alpha(\overset{N}{\bullet} e) = \overset{\sigma}{\bullet} \beta(e)$, and thus that α induces a bijection from $\overset{N}{\bullet} e$ to $\overset{\sigma}{\bullet} \beta(e)$. A similar proof holds for the mapping from $e \overset{N}{\bullet}$ to $\beta(e) \overset{\sigma}{\bullet}$. □

[2] Note that β is generalized, i.e. for a set E holds that $\beta(E) = \{\beta(e) \mid e \in E\}$.

At this point we still need to prove the following for an arbitrary WF-net in the aggregation class. For each causal net in a causal set, we should be able to color its condition graph using a conditional coloring. If we are able to construct such a coloring, we have satisfied the first requirement stated in Definition 3.11.

Lemma 7.9. (Conditional coloring exists)
Let Φ be a causal set, let A be the aggregation class of Φ, and let $(\sigma, \mathcal{B}) \in A$, with $\sigma = (P, T, F, M_0)$. Let $N = (C, E, K, S) \in \Phi$ be a causal net and $\Delta_N = (C, A)$ be the condition graph of N. Let $\beta_i \in \mathcal{B}$ be a labelling function belonging to N_i. We show that we can construct a mapping $\alpha : C \rightarrow P$, such that α is a conditional coloring of Δ_N.

Proof First, we look at the initial condition, i.e. the initially marked source condition. Assume $c \in C$ such that $\overset{N}{\bullet} c = \emptyset$. We call $c \overset{N}{\bullet} = \{e\}$. From the definition of a causal set (Def. 4.3), we know that $\{c\} = \overset{N}{\bullet} e$ and thus that there is no $c' \in C$ with $c \neq c'$ and $(c, c') \in A$. We know that $\overset{\sigma}{\bullet} \beta_i(e) = \{p_{ini}\}$ (Def. 7.2). By setting $\alpha(c) = p_{ini}$, we have a correct coloring for the initial condition c in N.

Second, we look at the final conditions, i.e. the sink conditions. Assume $c \in C$ such that $c \overset{N}{\bullet} = \emptyset$. We call $\overset{N}{\bullet} c = \{e\}$. From the definition of a causal set (Def. 4.3), we know that $\{c\} = e \overset{N}{\bullet}$ and thus that there is no $c' \in C$ with $c \neq c'$ and $(c, c') \in A$. We know that $|\beta_i(e) \overset{\sigma}{\bullet}| = 1$ (Def. 7.2). We say that $\beta_i(e) \overset{\sigma}{\bullet} = \{p\}$. By setting $\alpha(c) = p$, we have a correct coloring for any final condition c in N.

Finally, we split the graph up into two subgraphs. Let $A_{in} = \{(c_1, c_2) \in A \mid \overset{N}{\bullet} c_1 = \overset{N}{\bullet} c_2\}$ and let $A_{out} = \{(c_1, c_2) \in A \mid c_1 \overset{N}{\bullet} = c_2 \overset{N}{\bullet}\}$. Using the definition of a condition graph it is easy to see that $A_{in} \cup A_{out} = A$. We now show that for each subgraph $\delta_{in}(N) = (C, A_{in})$ and $\delta_{out}(N) = (C, A_{out})$ we can construct at least one conditional coloring. Then, we show that there is at least one conditional coloring that is the same for both subgraphs after which we can use Lemma 3.6 to show that this is a conditional coloring on the complete graph.

Consider the subgraph $\delta_{in}(N) = (C, A_{in})$. Using Lemma 7.4, it is easy to see that this graph consists of several complete components and that each component is a complete graph. Let $e \in E$. We know that $e \overset{N}{\bullet} \subseteq C$ and that $e \overset{N}{\bullet}$ defines a complete component in $\delta_{in}(N)$. Now, let $V_1, \ldots, V_n$ be maximal sets, such that for each $0 < i \leq n$ holds that $V_i \subseteq E \overset{N}{\bullet}$ and for all $c_1, c_2 \in V_i$ holds that $c_1 \overset{N}{\bullet} = c_2 \overset{N}{\bullet}$. For each V_i and $c \in V_i$, we say that $V_{i,in} = \{e\}$ and $V_{i,out} = c \overset{N}{\bullet}$.

From Definition 7.7, we know that for each $c \in V_i$ must hold that $\alpha(c) \in \{p \in P \mid \beta(\{e\}) \subseteq \overset{\sigma}{\bullet} p \wedge \beta(V_{i,out}) \subseteq p \overset{\sigma}{\bullet}\}$. Using rule 7 of Definition 7.2, we first prove a necessary condition for this. Assume $\beta(\{e\}) = \{t\}$, and $\beta(V_{i,out}) = T' = \{t'\}$. Rule 7 shows us that $|t \overset{\sigma}{\bullet} \cap \overset{\sigma}{\bullet} t'| \geq \sum_{e' \in V_{i,out}} |e \overset{N}{\bullet} \cap \overset{N}{\bullet} e'|$. From the definition of partition V, we know that $\sum_{e' \in V_{i,out}} |e \overset{N}{\bullet} \cap \overset{N}{\bullet} e'| = |V_i|$. Furthermore, $t \overset{\sigma}{\bullet} \cap \overset{\sigma}{\bullet} t' = \{p \in P \mid \beta(V_{i,in}) \subseteq \overset{\sigma}{\bullet} p \wedge \beta(V_{i,out}) \subseteq p \overset{\sigma}{\bullet}\}$. Therefore we know that there are at least enough colors available for each partition

V_i. The same way of reasoning can be used to show that there are at least enough colors available for each set of partitions $\upsilon \subseteq \{V_1, \ldots, V_n\}$. (The latter requires the use of rule 8 instead of 7 of Definition 7.2). Therefore, there exists at least one conditional coloring for the entire subgraph $\delta_{in}(N)$. Similarly, this can be shown for $\delta_{out}(N)$.

At this point, we have shown that we can construct conditional colorings for two subgraphs of $\delta(N)$, namely $\delta_{in}(N)$ and $\delta_{out}(N)$. The final part of the proof uses rule 8 of Definition 7.2, since we now have to show that *the same* conditional coloring can be constructed for both subgraphs. For this purpose, we consider a segment (C', E_{in}, E_{out}) in N. Since segments correspond to connected components of $\delta(N)$, it is sufficient to show that the same conditional coloring can be constructed for $\delta_{in}(N)$ and $\delta_{out}(N)$, restricted to C', which we call $\delta'_{in}(N)$ and $\delta'_{out}(N)$. From the definition of a segment, it is clear that this restriction does not disturb the structure of $\delta_{in}(N)$ and $\delta_{out}(N)$, i.e. in both graphs, each connected component is still a complete subgraph. Now consider a possible conditional coloring on $\delta'_{in}(N)$. Each color given to a condition in that graph refers to a place in the causal net. However, multiple conditions can be mapped onto each place, namely one condition for each token that was produced in that place by an element of E_{in}. The same holds for $\delta'_{out}(N)$, i.e. multiple condition can be mapped onto each place, namely one condition for each token that was consumed by a succeeding element of E_{out}. Since rule 9 of Definition 7.2 states that the tokens produced by E_{in} are the tokens consumed by E_{out}, it must be possible to construct the same conditional coloring α for both $\delta'_{in}(N)$ and $\delta'_{out}(N)$. Using Lemma 3.6, we then know that this coloring α is a conditional coloring on $\delta(N)'$, i.e. the restriction of $\delta(N)$ to C'.

Since we can now provide a conditional coloring on each connected component of $\delta(N)$, we have shown that we can construct a conditional coloring on the entire graph $\delta(N)$. □

To clarify the rather complex proof of Lemma 7.9 we use an example. Consider a causal net containing the fragment of a WF-net presented in Figure 11. We numbered the conditions 1 through 8 to be able to distinguish them. Now, assume that the two Petri nets presented in Figure 12 are parts of two alternative system nets appearing in the aggregation class of that causal net.

The proof of Lemma 7.9 depends on the condition graph of a run. Therefore, in Figure 13 we present the condition graph of the run presented in Figure 11. Note that we labelled the edges to show from which event the edge was derived.

In Lemma 7.9, the condition graph of Figure 13 (i.e. $\delta(N)$ in the lemma) is split up into two subgraphs, namely one for the input side of events (i.e. $\delta_{in}(N)$, see Figure 14) and one for the output sides of events (i.e. $\delta_{out}(N)$, see Figure 15).

Then the proof continues, by showing that for each of these two subgraphs it is possible to provide a conditional coloring. Figure 16 shows the possible labels for each subgraph and both Petri nets from Figure 12. It is easy to see that this indeed leads to several possible colorings in each graph.

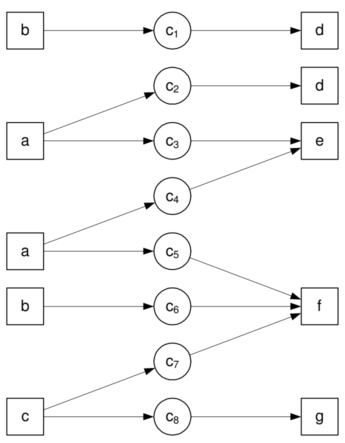


Fig. 11 A part of a run containing two segments.

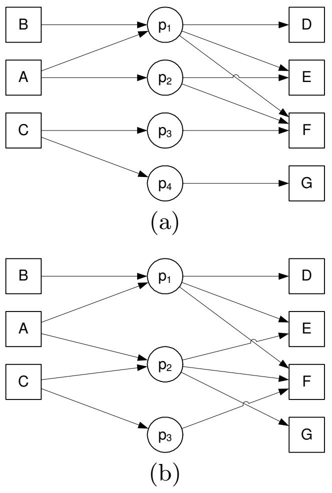


Fig. 12 Two parts of system nets in an aggregation class.

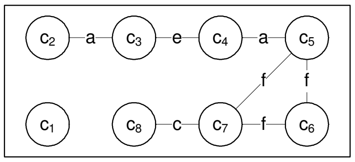
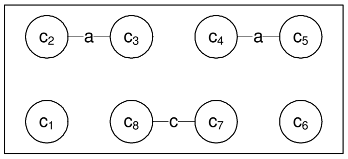


Fig. 13 Part of the condition graph of the run of Figure 11.



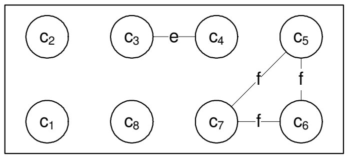


Fig. 14 Input subgraph of Figure 13. **Fig. 15** Output subgraph of Figure 13.

Finally, at this point it is proven that it is always possible to construct two coloring functions on the input and output subgraph that give the same label to each condition in both graphs. If we look at Figure 16 and we take the input subgraph shown in Figure 12(a) (i.e. the left-top figure) then it is easy to see that it is possible to label c_4 with p_2 and c_5 with p_1. This however is not possible in the output subgraph, since neighbour c_6 has to be mapped onto p_1. Instead there is only one mapping that is the same for both subgraphs. The last part of the proof uses the fact that for each segment the input enables the output. This implies that the tokens that is placed in p_1 has to be consumed from there again. Therefore, if we would label c_5 with p_1 then this would be the same as saying that transition A produces a token in p_1 which is consumed by transition F again. However, transition F also consumes another token from p_1, namely the one corresponding to c_6, i.e.

coming from transition B. This violates the fact that only one edge can exist between a place and a transition.

Figure 17 shows the only possible conditional coloring of the condition graph of Figure 13, using the labels provided by the system net 12(a) and Figure 18 shows the only possible conditional coloring of the condition graph of Figure 13, using the labels provided by the system net 12(b). Note that in general additional conditional colorings may be possible.

From figures 17 and 18, we can conclude that both system nets depicted in Figure 12 are indeed capable of producing the causal net of Figure 11, since we can construct a conditional coloring on the condition graphs.

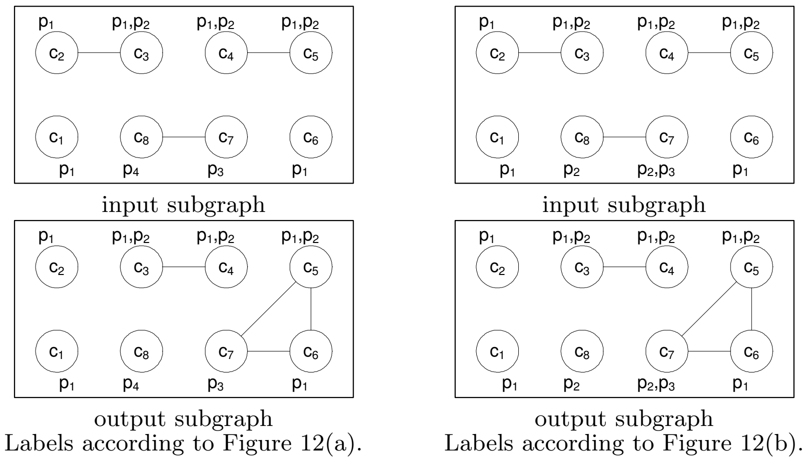


Fig. 16 Possible conditional colorings for the subgraphs of figures 14 and 15.

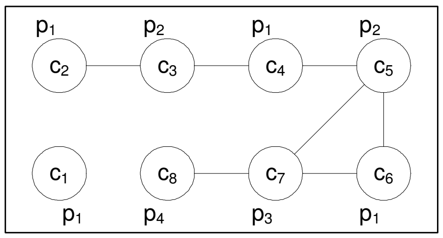


Fig. 17 The conditional coloring of Figure 13 according to Figure 12(a).

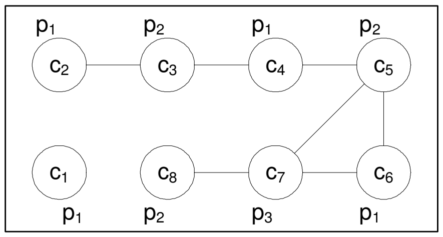


Fig. 18 The conditional coloring of Figure 13 according to Figure 12(b).

What remains to be shown is that the conditional coloring also fulfills the last part of the definition of a run, namely the demand with respect to the initial marking. Furthermore, it is interesting to note that we can conclude that at least three places are needed in the system net and that for example the place between C and G could also be p_1.

Lemma 7.10. (Initial marking can be mapped)
Let Φ be a causal set, let A be the aggregation class of Φ and let $(\sigma, \mathcal{B}) \in A$ with $\sigma = (P, T, F, M_0)$. Let $N = (C, E, K, S) \in \Phi$ be a causal net and $\Delta_N = (C, A)$ be the condition graph of N. Let $\alpha : C \to P$, such that α is a conditional coloring of Δ_N. We show that $\alpha(S) = M_0$.

Proof From Definition 7.2, we know that $M_0 = [p_{ini}]$. Furthermore, from Definition 4.3, we know that there is exactly one $c \in C$ with $S(c) = 1$. Moreover, using Lemma 7.9, we conclude that $\alpha(c) = p_{ini}$ and thus $\alpha(S) = [p_{ini}] = M_0$. □

Finally, we can combine everything and state that each WF-net in an aggregation class is indeed a system net of a causal set.

Property 7.11. (Aggregation class contains system nets)
Let Φ be a causal set, let A be the aggregation class of Φ and let $(\sigma, \mathcal{B}) \in A$ with $\sigma = (P, T, F, M_0)$. Let $N = (C, E, K, S) \in \Phi$ be a causal net with event labelling function $\beta \in \mathcal{B}$, condition graph $\Delta_N = (C, A)$ and $\alpha : C \to P$ a conditional coloring of Δ_N. (N, α, β) is a run of σ.

Proof This proof is given by combining Lemma 7.8, (which shows that for all $e \in E$, α induces a bijection from $\overset{N}{\bullet} e$ to $\overset{\sigma}{\bullet}\beta(e)$ and from $e\overset{N}{\bullet}$ to $\beta(e)\overset{\sigma}{\bullet}$) and Lemma 7.10 (which shows that $\alpha(S) = M_0$). □

We have shown that it is possible to take a set of causal nets and construct a system net such that each causal net is a run of that system net, as long as the causal nets have one initially marked condition. What we did not show are the conditions under which the aggregation class is not empty. These conditions however, cannot be given based on a set of causal nets. Even if these causal nets belong to one causal set, this is still not enough. What we *can* show however, is that if we start from a *sound* WF-net as a system net, generate a set of runs and remove the labels of places, the original WF-net is in the aggregation class. For the full definition of soundness, we refer to [1].

Property 7.12. (A system net is in the aggregation class of its runs)
Let $\sigma = (P, T, F, M_0)$ be a sound WF-net. We consider σ to be a system net. Let $B = \{(N_i, \alpha_i, \beta_i) \mid 0 \leq i < n\}$ be the causal behaviour of that system net, such that each (N_i, α_i, β_i) is a run of that system net, with $N_i = (C_i, E_i, K_i, S_i)$. Let $\mathcal{B} = \{\beta_i \mid 0 \leq i < n\}$ and $\Phi = \{N_i \mid 0 \leq i < n\}$ be a causal set. We show that $(\sigma, \mathcal{B}) \in A(\Phi)$.

Proof We show that all conditions of Definition 7.2 are satisfied.

1. $T = \bigcup_{0 \leq i < n} Rng(\beta_i)$ is the set of transitions. Since the WF-net is sound, there are no dead transitions thus implying that in its causal set each transition appears as an event at least once.

2. $\mathcal{B}$ is the set of all labelling functions, i.e. $\mathcal{B} = \{\beta_i \mid 0 \leq i < n\}$.
3. For all $p \in P$ holds that $\overset{\sigma}{\bullet} p \cup p \overset{\sigma}{\bullet} \neq \emptyset$. Since every sound WF-net is connected, this condition is satisfied,
4. $M_0 = [p_{ini}]$ and $\overset{\sigma}{\bullet} p_{ini} = \emptyset$. Since σ is a WF-net, there is exactly one place $p_{ini} \in P$, such that $\overset{\sigma}{\bullet} p_{ini} = \emptyset$ and $M_0 = [p_{ini}]$,
5. For each causal net N_i, with $e \in E_i$ and $\beta_i(e) = t$ and $\overset{N_i}{\bullet} e = \{c\}$, holds that if $S_i(c) = 1$ then $p_{ini} \in \overset{\sigma}{\bullet} t$. Since $S_i(c) = 1$, we know that $\overset{N_i}{\bullet} c = \emptyset$. Now assume $\alpha_i(c) = p$. The fact that for all $e' \in E_i$, α_i induces local bijections from $e' \overset{N_i}{\bullet}$ to $\beta_i(e') \overset{\sigma}{\bullet}$ implies that $\overset{\sigma}{\bullet} p = \emptyset$ and since σ is a workflow net, this implies that $p = p_{ini}$. Moreover, the fact that for all α_i induces local bijections from $\overset{N_i}{\bullet} e$ to $\overset{\sigma}{\bullet} t$ implies that $p_{ini} \in \overset{\sigma}{\bullet} t$,
6. For each causal net N_i, with $e \in E_i$ and $\beta_i(e) = t$ holds that $|t \overset{\sigma}{\bullet}| = |e \overset{N_i}{\bullet}|$ and $|\overset{\sigma}{\bullet} t| = |\overset{N_i}{\bullet} e|$. Since α_i induces bijections from $e \overset{N_i}{\bullet}$ to $t \overset{\sigma}{\bullet}$ and from $\overset{N_i}{\bullet} e$ to $\overset{\sigma}{\bullet} t$, this condition is satisfied,
7. For each causal net N_i, with $e \in E_i$, $\beta_i(e) = t$ and $T' \subseteq T$ holds that $|t \overset{\sigma}{\bullet} \cap \bigcup_{t' \in T'} (\overset{\sigma}{\bullet} t')| \geq \sum_{e' \in E_i, \beta(e') \in T'} |e \overset{N_i}{\bullet} \cap \overset{N_i}{\bullet} e'|$. Let $e \in E_i$ with $\beta_i(e) = t$ and let $T' \subseteq T$. Assume that there $|t \overset{\sigma}{\bullet} \cap \bigcup_{t' \in T'} (\overset{\sigma}{\bullet} t')| = m$, i.e. there are m places between t and T'. Furthermore, assume that $\sum_{e' \in E_i, \beta(e') \in T'} |e \overset{N_i}{\bullet} \cap \overset{N_i}{\bullet} e'| < m$. Since for all $e' \in E_i$ with $beta_i(e_i) = t_i$, α_i induces local bijections from $\overset{N_i}{\bullet} e_i$ to $\overset{\sigma}{\bullet} t_i$, we know that there are at least two $c_1, c_2 \in e \overset{N_i}{\bullet}$ that are mapped onto the same $p \in P$. However, since $p \in t \overset{\sigma}{\bullet}$ this violates the local bijection property of α_i,
8. For each causal net N_i, with $e \in E_i$, $\beta_i(e) = t$ and $T' \subseteq T$ holds that $|\bigcup_{t' \in T'} (t' \overset{\sigma}{\bullet}) \cap \overset{\sigma}{\bullet} t| \geq \sum_{e' \in E_i, \beta(e') \in T'} |e' \overset{N_i}{\bullet} \cap \overset{N_i}{\bullet} e|$. The proof for this property is similar to the previous one.
9. For each causal net N_i and any segment (C'_i, E_{in}, E_{out}) of N_i holds that $\biguplus_{e \in E_{in}} \left(\beta_i(e) \overset{\sigma}{\bullet}\right) = \biguplus_{e \in E_{out}} \left(\overset{\sigma}{\bullet} \beta_i(e)\right)$. This property relates to soundness. If one set of transitions produces tokens then these tokens will be consumed by another set of transitions (i.e. no tokens are "left behind" in the execution of a sound WF-net). The only exception is the transition that produces a token in the output place, but that transition cannot produce any tokens in any other place. Therefore, in each run, the input events of a segment will enable the output events of that segment.

□

In this section, we have presented the *NLC* algorithm, that takes a set of runs without condition labels as a starting point. From these runs, an aggregation class of WF-nets is defined. Moreover, we have shown that if the runs were generated from some sound WF-net, then the WF-net itself is in that aggregation class. We conclude this paper with an elaborate example of the application of the *NCL* algorithm.

8 Example

In this section, we again give an example of the aggregation process of a set of causal nets. Consider the four causal nets presented in Figure 19. These causal nets originate from a workflow system in which two activities need to be performed. These activities are labelled `L` and `R`. However, in the workflow design, there are several options. First, the system initializes the two activities trough event `Init LR`. Then, a person can decide to perform both activities at once, which is represented by the event `Do LR`. When both activities have been performed, the workflow can be finished through event `exit LR`. However, in a typical workflow environment, people can make mistakes and therefore, in Figure 19(b), both activities have been undone, thus generating events `Undo L` and `Undo R`. Finally, in Figure 19(c) and (d) it is shown that the workflow system allows for the two activities to be executed separately, through `Do L` and `Do R`. To keep things interesting, the last causal net belongs to a case in the workflow system that is not finished yet. However, this set of causal nets still conforms to the definition of a causal set (i.e. Definition 4.3).

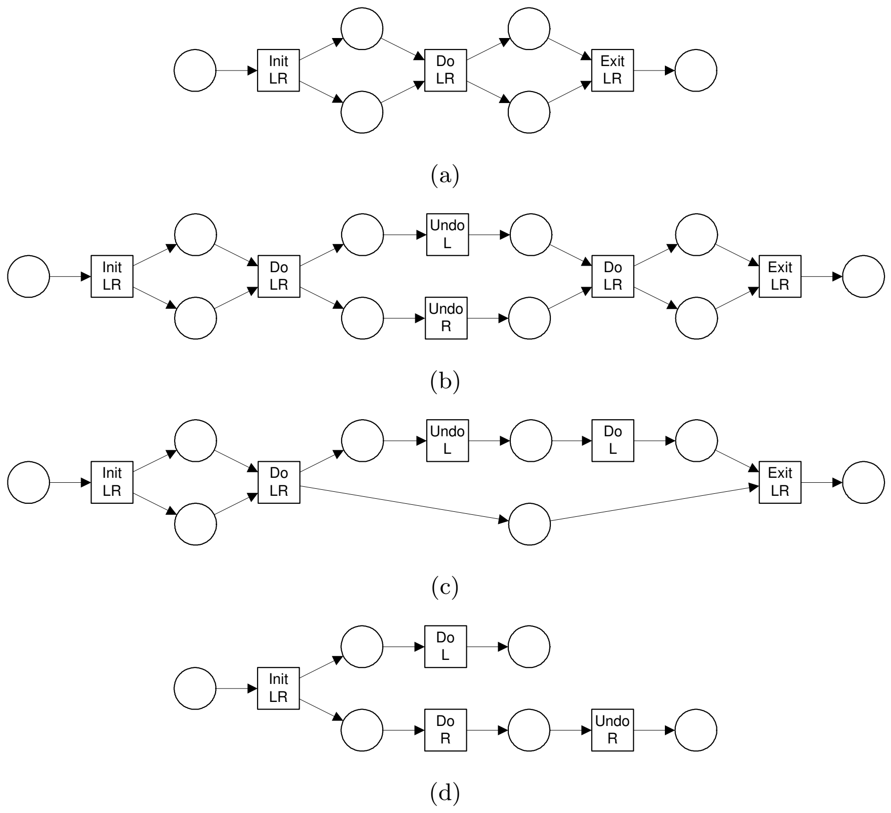


Fig. 19 Four causal nets without condition labels.

Using the *NCL* algorithm given by Definition 7.2, we can generate the aggregation class of the causal set of Figure 19. In fact, this aggregation class only contains one workflow net, namely the workflow net shown in Figure 20.

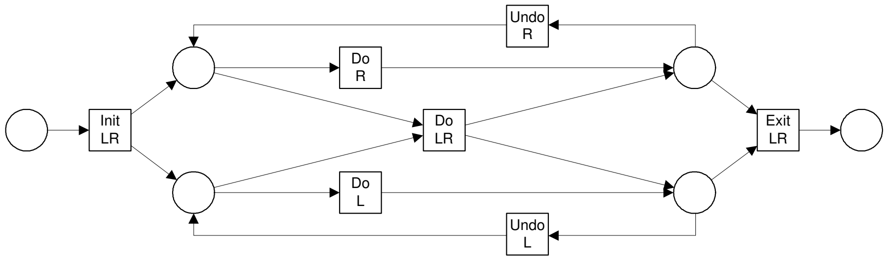


Fig. 20 The only element of the aggregation class of the four nets of Figure 19.

It is important to note that the workflow net in Figure 20 actually allows for more behaviour than is shown in the four causal nets of Figure 19. It is for example possible to execute a long sequence of `Do L` and `Undo L`, which is not shown in the causal nets. Therefore, this example once more shows that each net in the aggregation class can actually generate the runs of the causal set it was constructed from, but it might be able to generate more runs.

9 Conclusion and Future Work

In this paper, we looked at process mining from a new perspective. Instead of starting with a set of traces, we started with runs, which are partial orders on events. We presented three algorithms to generate a Petri net from these runs. The first algorithm assumes that for each run, all labels of both conditions and events are known. The second algorithm relaxes this by assuming that some transitions can have the same label (i.e. duplicate labels are allowed in the system net). This algorithm can also be used if only condition/place-labels were recorded. Finally, we provided an algorithm that does not require condition labels, i.e. the event/transition labels are known, the condition/place labels are unknown and duplicate transition labels are not allowed.

The results presented in this paper hold for a sub-class of Petri nets, so-called WF-nets. However, the first two algorithms presented here can easily be generalized to be applicable to any Petri net. For the third algorithm this can also be done, however, explicit knowledge about the initial marking would be required. When taking a set of runs as a starting point, this knowledge is not present in the general case.

References

1. W.M.P. van der Aalst. Workflow Verification: Finding Control-Flow Errors using Petri-net-based Techniques. In W.M.P. van der Aalst, J. Desel, and

A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 161–183. Springer-Verlag, Berlin, 2000.

2. W.M.P. van der Aalst, A.K. Alves de Medeiros, and A.J.M.M. Weijters. Genetic Process Mining. In *Proceedings of the 26th International Conference on Applications and Theory of Petri Nets*, volume 3536 of *Lecture Notes in Computer Science*, 2005.
3. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods and Systems*. Academic Service, Schoonhoven, 1997.
4. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
5. W.M.P. van der Aalst and A.J.M.M. Weijters, editors. *Process Mining*, Special Issue of Computers in Industry, Volume 53, Number 3. Elsevier Science Publishers, Amsterdam, 2004.
6. E. Badouel and Ph. Darondeau. Theory of regions. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets Part 1 (Basic Models)*, volume 1491 of *Lecture Notes in Computer Science*, pages 529–586. Springer-Verlag, Berlin, 1998.
7. A.K. Alves de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. Genetic Process Mining: A Basic Approach and its Challenges. In *BPM - BPI workshop*, 2005.
8. J. Desel. Validation of Process Models by Construction of Process Nets. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 110–128. Springer-Verlag, Berlin, 2000.
9. J. Desel and T. Erwin. Hyprid specifications: looking at workflows from a run-time perspective. *Computer Systems Science and Engineering*, 5:291–302, 2000.
10. J. Desel and W. Reisig. The synthesis problem of petri nets. *Acta informatica*, 33:297–315, 1996.
11. B.F. van Dongen and W.M.P. van der Aalst. Multi-phase Process mining: Building Instance Graphs. In P. Atzeni, W. Chu, H. Lu, S. Zhou, and T.W. Ling, editors, *Conceptual Modeling - ER 2004*, volume 3288 of *Lecture Notes in Computer Science*, pages 362–376. Springer-Verlag, Berlin, 2004.
12. B.F. van Dongen and W.M.P. van der Aalst. Multi-phase Process mining: Aggregating Instance Graphs into EPCs and Petri Nets. In *PNCWB 2005 workshop*, pages 35–58, 2005.
13. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005.
14. A. Ehrenfeucht and G. Rozenberg. Partial (Set) 2-Structures - Part 1 and Part 2. *Acta Informatica*, 27(4):315–368, 1989.
15. David Harel, Hillel Kugler, and Amir Pnueli. Synthesis revisited: Generating statechart models from scenario-based requirements. In Hans-Jörg Kreowski, Ugo Montanari, Fernando Orejas, Grzegorz Rozenberg, and Gabriele Taentzer, editors, *Formal Methods in Software and Systems Modeling*, volume 3393 of *Lecture Notes in Computer Science*, pages 309–324. Springer, 2005.
16. J. Herbst and D. Karagiannis. Workflow mining with InWoLvE. *Comput. Ind.*, 53(3):245–264, 2004.
17. Robert Lorenz and Gabriel Juhás. Towards synthesis of petri nets from scenarios. In Susanna Donatelli and P. S. Thiagarajan, editors, *ICATPN*, volume 4024 of *Lecture Notes in Computer Science*, pages 302–321. Springer, 2006.
18. A. Roychoudhury and P.S Thiagarajan. Communicating transaction processes. In J. Lilius, F. Balarin, and R. Machado, editors, *Proceedings of Third International Conference on Application of Concurrency to System Design (ACSD2003)*, pages 157–166. IEEE Computer Society, 2003.
19. E. Smith. On net systems generated by process foldings. In G. Rozenberg, editor, *Advances in Petri Nets*, volume 524 of *Lecture Notes in Computer Science*, pages 253–276. Springer-Verlag, Berlin, 1991.