On the α-Reconstructibility of Workflow Nets

Eric Badouel

Inria Rennes-Bretagne Atlantique
Campus universitaire de Beaulieu
F35042 Rennes Cedex, France
eric.badouel@inria.fr

Abstract. The process mining algorithm α was introduced by van der Aalst *et al.* for the discovery of workflow nets from event logs. This algorithm was presented in the context of structured workflow nets even though it was recognized that more wokflow nets should be reconstructible. In this paper we assess α algorithm and provide a more precise description of the class of workflow nets reconstructible by α.

Keywords: Process Mining, Workflows, Net Synthesis

1 Introduction

One of the purpose of *process mining* [11] is to construct or to reconstruct from an event log a business process model that can generate this event log. The game is to dig out of event logs sufficient informations on the structure of their generating model. As a technique for *model discovery*, process mining has some connections with machine learning.

Process mining may be used for the purpose of modelling. For instance, after collecting over a long period of time information on the health history of many patients, including the diagnosis and treatment steps, one may want to extract from this record an accurate model of the workflow system of an hospital. *Reverse engineering*, which consists of reconstructing from representative use cases an existing but partially unknown system, is another activity of model discovery that can be achieved by process mining. According to [11], process mining can also be used for *conformance* checking or *enhancement* of business process models. For instance, *process-aware systems* record run-time informations used to detect discrepancies between expected and actual behaviours and to refactor these systems.

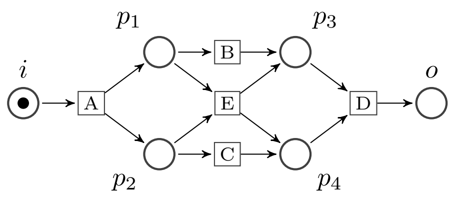
In this paper we focus our attention on model discovery. We fix a subclass of net systems, the so-called *workflow nets*, as the class of target models for process mining. Section 2 introduces the model of workflow nets and we present and illustrate algorithm α [12, 13]. The mining algorithm α tries to reconstruct the places of a workflow net by taking solely into account those pairs of transitions that follow each other in some execution sequence. This assumption has strong impacts, considered in Section 4, on the class of reconstructible workflow nets. In particular, we point out that their inner places are *boundary* places, a condition that we study in more details in Section 5. On that basis we obtain a characterization of the class of workflow nets that are discovered by α. This class contains the structured workflow nets of [12, 13].

2 Discovering Workflow Nets from Event Logs

A log is a finite set of execution sequences of a workflow system. Given a log of the system to be discovered (i.e., constructed or reconstructed), each event reported in this log refers to an *activity*, i.e., a particular step in the workflow system, and to a specific *case*, i.e., a process instance. Additional informations pertaining to events are generally included, for instance a time stamp or the identity of the performer. Usually cases have little or no connection with one another, and time stamps will mainly serve the purpose of indicating the correct ordering of the activities. An event log may then be abstracted to a *set of sequences of activities.* Each sequence represents all activities of a case from the time when it enters the system till the time when it leaves the system.

As the target representation of process mining, we consider a subclass of elementary net systems, called *workflow nets.* The goal of process mining is to synthesize from an event log a workflow net that can reproduce all sequences of activities traced in this log, from the inception of a case to its termination.

Example 1. The workflow net displayed below specifies all possible behaviours of an isolated case in some workflow system. In other words, the firing sequences of the workflow net represent all activities pertaining to a case from the time when the case enters into the system (the input place i is marked) until the case terminates and exits from the system (the output place o is marked). The possible sequences of activities are thus $ABCD, ACBD, AED$.



A workflow net contains an *input* place i and an *output* place o. The input place i is initially marked, and this initial marking represents the entry of a new case in the system. The output place o gets marked when the case comes to completion, and this final marking represents the exit of the case from the system. The current marking of the workflow net represents the current status of a case. It is assumed that the execution of a case can always reach termination, and that it cannot interfere with the execution of any subsequent case. The latter property, called *soundness* in [13], can be formalized as follows: when the output place is marked, all other places must be empty. The marking in which the output place and no other place is marked is called the *terminal marking.* Therefore, in a (sound) workflow net, the terminal marking must be reachable from any other reachable marking.

Moving the token from the output place back to the initial place is a way to simulate the termination of a case and the inception of a new case. A workflow net with such an implicit feedback may be seen as a cyclic generator, that can iterate in sequence all scenarios of execution of all cases. Instead of adding a feedback transition from the output place to the input place, one might as well coalesce the input place and the output place (confusing thus the initial and terminal markings). With this representation, the two crucial properties of workflow nets N may be reformulated equivalently as follows: *(i)* the net N' obtained by coalescing the input place and the output place of N is *reversible* (the initial marking may be reached from any reachable marking) and *(ii)* the initial (or terminal) marking is the only reachable marking of N' containing the input (or output) place. This is essentially the definition of workflow nets which we adopt below.

Definition 1. *A* workflow net *is a contact-free, initially life, place simple and connected elementary net system $N = (P, T, F, M_0)$ where P contains an input place i and an output place o (the remaining places $p \in P \setminus \{i, o\}$ are called* inner places*), such that the following conditions hold:*

1. $^\bullet i = o^\bullet = \emptyset$
2. $(\forall p \in P \setminus \{i, o\}) \quad {}^\bullet p \neq \emptyset \quad \wedge \quad p^\bullet \neq \emptyset$
3. $M_0 = \{i\}$.
4. *The* closed *net system $N' = (P', T, F', \{\iota\})$ obtained from N by replacing places i and o with a unique place ι such that ${}^\bullet\iota = {}^\bullet o$ and $\iota^\bullet = i^\bullet$ is reversible, and its initial marking $M'_0 = \{\iota\}$ is the unique reachable marking of N' in which the place ι is marked.*

Let us recall that a net system is initially life if any transition is enabled in some reachable marking, and it is life if, for any transition t and for any reachable marking M, the transition t is enabled in some marking reachable from M. Notice that the closed net system N' being both initially life and reversible is life. Let us also recall that a *contact situation* for an elementary net system is given by an accessible marking M and a transition t such that ${}^\bullet t \subseteq M$ and $M \cap t^\bullet \neq \emptyset$, i.e., transition t is disabled in marking M because of one of its output places. An elementary net is *contact-free* if it has no contact situation. i.e., $M \cap t^\bullet = \emptyset$ for every accessible marking M and transition t such that ${}^\bullet t \subseteq M$. Any elementary net system is equivalent to a contact-free net obtained by adding complementary places where needed [1]. Contact-free elementary net systems are equivalent to one-safe net systems and, up to this correspondence, Def. 1 is an equivalent reformulation of the definition of *sound workflow nets* given in [13].

[1]Places p and $\overline{p}$ are *complementary* places when ${}^\bullet\overline{p} = p^\bullet$, $\overline{p}^\bullet = {}^\bullet p$, and $p \in M_0 \Leftrightarrow \overline{p} \notin M_0$, then for every marking M accessible (from the initial marking M_0) one has $p \in M \Leftrightarrow \overline{p} \notin M$. If a place has no complementary place one can formally add one such place to the net system without modifying its behaviour (both net systems have isomorphic reachability graph). One can then eliminate the contact situations by adding complementary places to those (output) places involved in some contact situation.

Given a workflow net $N = (P, T, F, M_0)$, the *full log* of N is the language $\mathcal{L}(N) = \{u \in T^* \mid \{i\} [u\rangle \{o\}\}$, i.e., the full log of N is the set of all firing sequences from the initial marking $\{i\}$ to the final marking $\{o\}$. A *log* of N is any subset $W \subseteq \mathcal{L}(N)$ such that every transition $t \in T$ occurs in at least one execution sequence in W. A *workflow log* is any log of a workflow net. Since workflow nets have no dead transitions, the full log of a workflow net is actually a log of this workflow net.

Example 2 (Exple. 1 continued). Consider the log $\{ABCD, ACBD, AED\} \subseteq \mathcal{L}(N)$ of the workflow net N shown in Exple.1. Every execution sequence in this log starts with event A and ends with event D. In between, one is left the choice to perform either the event E or the events B and C, which are concurrent since they occur in the log in both orders BC and CB. Using these structural informations extracted from the log, the α algorithm can reconstruct the workflow net N from the considered log. All three execution sequences in this log are actually needed by algorithm α: every activity ought to be reported in at least one execution sequence(thus AED is needed), and for any pair of concurrent events, at least two execution sequences exhibiting the two possible orderings are needed (thus $ABCD$ and $ACBD$ are needed).

The set of all execution sequences of a workflow net may grow exponentially with the number of events, owing to their possible concurrency. Therefore the execution sequences reported in an event log usually form a small but hopefully representative set of samples of all possible behaviours. In order to discover a workflow net N from some small log $W \subset \mathcal{L}(N)$, a process mining algorithm must carry out some non-trivial generalization over the execution sequences in this log. For that purpose algorithm α is presented as the composition $\alpha = Syn \circ Abs$ of an abstraction function Abs and a synthesis function Syn. The role of the function Abs is to extract from a log W the relevant relations between the events that occur in this log. The role of the function Syn is to reflect, as faithfully as possible, these relations in the structure of a synthesized net system.

Definition 2. *The α-abstraction of a workflow log $W \subset T^*$ is the triple $Abs(W) = \langle I_W, C_W, O_W \rangle$ where:*

1. $I_W = \{t \in T \mid \exists u \in T^* \quad t \cdot u \in W\}$ *is the set of transitions starting some execution sequence in the log;*
2. $O_W = \{t \in T \mid \exists u \in T^* \quad u \cdot t \in W\}$ *is the set of transitions ending some execution sequence in the log;*
3. $C_W = \left\{t \cdot t' \in T^2 \mid \exists u, v \in T^* \quad u \cdot t \cdot t' \cdot v \in W\right\}$ *is the set of pairs of transitions appearing consecutively in some execution sequence in the log.*

Definition 3. *$W \subseteq \mathcal{L}(N)$ is said to be a* complete *log of workflow net N when $Abs(W) = Abs(\mathcal{L}(N))$*

Hence a complete log W contains already all the information needed by algorithm α: the net systems $\alpha(W)$ and $\alpha(\mathcal{L}(N))$ are identical up to a bijective renaming of their set of places. This relation, called *isomorphism*, will be denoted by $\cong$.

Hence a workflow net is α-reconstructible (i.e., $N \cong \alpha(\mathcal{L}(N))$) if and only if it can be discovered from any of its complete log $W \subseteq \mathcal{L}(N)$ (i.e., $N \cong \alpha(W)$).

From the theory of event structures [14, 15], we know that the behaviour of a net system can be captured, up to net unfolding, by the basic relations of *causality*, *conflict* and *concurrency* between events. When unfolding a net to an event structure, the events are not in bijective correspondence with the transitions of the net, since two occurrences of the same transition may be distinguished by their past history. Nevertheless, given a workflow net N with set of transitions T and a log W of N, one may derive from the α-abstraction of this log three relations $\rightarrow_W$, $\sharp_W$, $\|_W$ between the transitions of N (the activities reported in the log), approximating loosely the relations of causality, conflict and concurrency in the associated event structure. These relations are the following:

$$\begin{array}{lrcl}
\text{causality:} & t \rightarrow_W t' & \Leftrightarrow & t \cdot t' \in C_W \ \wedge \ t' \cdot t \notin C_W \\
\text{conflict:} & t \,\sharp_W\, t' & \Leftrightarrow & t \cdot t' \notin C_W \ \wedge \ t' \cdot t \notin C_W \\
\text{concurrency:} & t \,\|_W\, t' & \Leftrightarrow & t \cdot t' \in C_W \ \wedge t' \cdot \ t \in C_W
\end{array}$$

From these relations between transitions, one may derive again the following relations between sets of transitions.

Definition 4. *Let $W \subseteq T^*$ be a workflow log. For any sets of transitions $A, B \subseteq T$, let $A \prec_W B$ when the following three conditions hold:*

1. $(\forall a \in A)(\forall b \in B) \quad a \ \rightarrow_W \ b,$
2. $(\forall a_1, a_2 \in A) \quad a_1 \sharp_W a_2,$ *and*
3. $(\forall b_1, b_2 \in B) \quad b_1 \sharp_W b_2$

Let $A \prec_W^m B$ when A and B are maximal sets with the property $A \prec_W B$, i.e., $A \prec_W^m B \Leftrightarrow (A \prec_W B) \wedge (A' \prec_W B' \wedge A \subseteq A' \wedge B \subseteq B' \Rightarrow A = A' \wedge B = B')$.

Note that the definition of $\prec_W$ may be applied to any workflow log, and in particular to the full log $\mathcal{L}(N)$ of a workflow net N. Using the above relations, the synthesis function *Syn* used in the α algorithm may be defined as follows.

Definition 5. *Let $\langle I_W, C_W, O_W \rangle$ be the α-abstraction of a workflow log $W \subset T^*$. Then $\alpha(W) = Syn(\langle I_W, C_W, O_W \rangle)$ is the elementary net system (P, T, F, M_0) defined as follows:*

1. $P = \{i, o\} \cup \{p_{A,B} \mid A, B \subseteq T \wedge A \prec_W^m B\},$
2. ${}^\bullet i = \emptyset,$ *and* $i^\bullet = I_W,$
3. ${}^\bullet o = O_W,$ *and* $o^\bullet = \emptyset,$
4. ${}^\bullet p_{A,B} = A,$ *and* ${p_{A,B}}^\bullet = B,$
5. $M_0 = \{i\}.$

Example 3. The algorithm α reconstructs the net system of Exple.1 from its set of firing sequences $W = \{ABCD, ACBD, AED\}$. We have $I_W = \{A\}$, $O_W = \{D\}$, and $C_W = \{AB, BC, CD, AC, CB, BD, AE, ED\}$. For any pair of transitions t and t' one of the following exclusive conditions holds: $t \rightarrow_W t'$,

$t' \rightarrow_W t$, $t \sharp_W t'$, or $t \|_W t'$. In the given example we obtain the following classification:

	B	C	D	E
A	$A \rightarrow_W B$	$A \rightarrow_W C$	$A \sharp_W D$	$A \rightarrow_W E$
B		$B \|_W C$	$B \rightarrow_W D$	$B \sharp_W E$
C			$C \rightarrow_W D$	$C \sharp_W E$
D				$E \rightarrow_W D$

The maximal elements of $\prec_W$ are obtained from the smallest ones, namely $\{t\} \prec_W \{t'\}$ for $t \rightarrow_W t'$, by progressively adjoining new elements to each of these two sets while preserving the conditions imposed on $\prec_W$. We obtain:

$$\{A\} \prec_W^m \{B, E\} \qquad \{A\} \prec_W^m \{C, E\} \qquad \{B, E\} \prec_W^m \{D\} \qquad \{C, E\} \prec_W^m \{D\}$$

providing respectively the places p_1, p_2, p_3, and p_4 of the net system displayed in Exple.1.

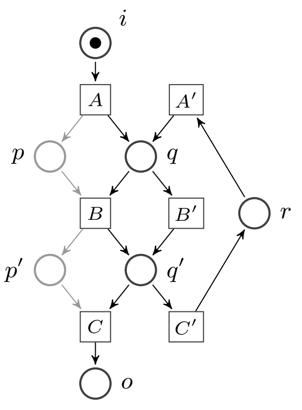
3 Some Observations about the Algorithm α

The reader may feel uncomfortable with the fact that the synthesized places are associated only from the maximal elements of relation $\prec_W$. Of course, associating a place p with every pair such that $A \prec_W B$ would produce a net with a large number of places. We may expect many of these additional places (if not all) to be implicit, so that one can remove them without having an impact on the behaviour of the system.

Definition 6. *A place p of a (contact-free) net system $N = (P, T, F, M_0)$ is a* (structurally) implicit place *if for every reachable marking M and transition $t \in p^\bullet$, ${}^\bullet t \setminus \{p\} \subseteq M \Rightarrow p \in M$.*

The next example shows however that some of these places *are not implicit* and therefore the decision to restrict to the maximal elements of the relation $\prec_W$ has an impact of the expressivity of algorithm α.

Example 4. Let us consider the workflow net N depicted next. Its language is $\mathcal{L}(N) = A\,(B'C'A')^*\,B\,(C'A'B')^*\,C$. A complete log is $W = \{ABC, AB'C'A'BC'A'B'C\}$. Places q, q' and r correspond to maximal elements of relation $\prec_W$, namely $\{A, A'\} \prec_W^m \{B, B'\}$, $\{B, B'\} \prec_W^m \{C, C'\}$, and $\{C'\} \prec_W^m \{A'\}$ respectively. Whereas ${}^\bullet p = \{A\} \prec_W^m \{B\} = p^\bullet$ and ${}^\bullet p' = \{B\} \prec_W^m \{C\} = p'^\bullet$ are not. Nevertheless these places are not implicit places since they disable the firing of transition B (respectively C) in the marking reached after firing $ABC'A'$ (resp. AB') where place q (resp. place q') is marked. The net $N' = \alpha(W)$ synthesized by algorithm α is obtained from N by suppressing these two places p and p'. Its language is $\mathcal{L}(N') = A\,(B + B')\,(C'A'\,(B + B'))^*\,C$.



Algorithm α is *sober* in the sense that for any complete log $W \subseteq \mathcal{L}(N)$ of a workflow net N any larger log $W \subseteq W' \subseteq \mathcal{L}(N)$ is also complete and the minimal size of complete logs of workflow nets is asymptotically negligible w.r.t. the size of their languages. Indeed when N ranges over workflow nets with set of transitions T, the size of $Abs(\mathcal{L}(N))$ is in $O(|T|^2)$. Moreover, a firing sequence of N contained in a log W may contribute several pairs of transitions in C_W. Therefore, one may expect to find complete logs of N with size even smaller than $O(|T|^2)$. Sobriety means that one can assume $W \subseteq \mathcal{L}(N)$ to be a complete log of workflow net N as soon as it contains a reasonable number of its execution sequences. Process reconstruction then amounts to the following:

Workflow net discovery

Problem Reconstruct a workflow net N from one of its complete log $W \subseteq \mathcal{L}(N)$.
Solution N is α-reconstructible if and only if $N \cong \alpha(W)$ if and only if the following two conditions hold: *(i)* $\alpha(W)$ is a workflow net, and *(ii)* $W \subseteq \mathcal{L}(\alpha(W))$, i.e., the synthesized net can reproduce each execution sequence in W.

The reader may have expected any net constructed by algorithm α to be an α reconstructible workflow net. Or more precisely, that α algorithm computes a closure operation providing the best approximation of a given log by a workflow net. This is unfortunately not the case as illustrated by the next example.

Example 5. Algorithm α relies only on the local informations reported in $Abs(\mathcal{L}(N))$. The price to pay for the locality of observations is that one cannot detect a situation where a transition t is an immediate cause of t' (there exists a non implicit place [2] in $t^\bullet \cap {}^\bullet t'$) but t' cannot occur immediately after t because it depends on others transitions that can be fired only after t. The following example illustrates this situation. In the language of the workflow net of Fig. 1, namely $W = \{ACD, BCE\}$, event D never follows immediately event A even though A is an immediate cause of D. Note however that p is not an implicit place because it disables D to be fired after the sequence BC: q is marked but not p after firing this sequence. The workflow net derived by algorithm α from W, the language of the workflow net of Fig. 1 is displayed in Fig. 2. This net reproduces the original workflow net of Fig. 1 but for the two places p and p'. Due to the absence of these two places the dependency of D on A (and similarly the dependency of E on B) is not inferred by algorithm α and the resulting net system contains two additional execution sequences ACE and BCD. In this case algorithm α looses precision by generalizing too much (it is underfitted). N_1 is not α-recontructible. Still N_1 is isomorphic to $\alpha(W')$ where $W' = \{ACD, BCE, AD, BE\}$; however

[2] Since implicite place have no impact on the net behaviour an implicit place in $t^\bullet \cup {}^\bullet t'$ can represent a fictitious dependency between t and t'. We say that t is an *immediate cause* of t' when $t^\bullet \cup {}^\bullet t'$ contains a non implicit place.

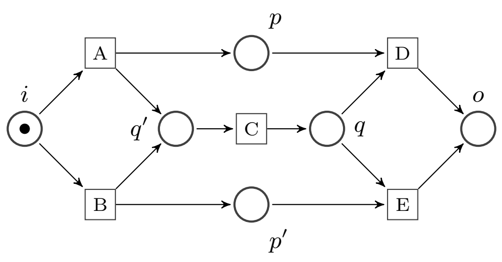


Fig. 1. a workflow net N_1

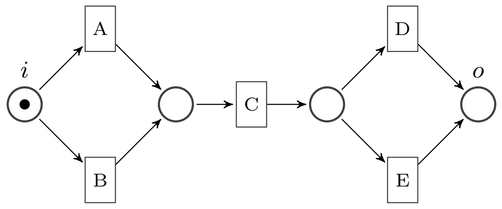


Fig. 2. the workflow net N_2 constructed by algorithm α from the language of the workflow net N_1 of Fig. 1

$W' \not\subseteq \mathcal{L}(N_1) = \{ACD, BCE\}$. Thus a net produced from a log does not necessarily reproduce every execution sequence contained in the log.

The preceding example shows that α cannot be associated with a Galois connection $W \subseteq \mathcal{L}(N) \Leftrightarrow N \leq \alpha(W)$ where $\leq$ is some order (or pre-order) relation on net systems. If it were the case we would have $W \subseteq \mathcal{L}(\alpha(W))$ for every $W \subseteq T^*$, in contradiction with the above example. Thus α cannot be used to provide the best net system N, according to some order, or pre-order relation, whose language contains a given log. For that reason we put stress on *process discovery* and in this context sobriety is a crucial property since we need to be assured that the log given as input is a complete log of the net to be discovered.

An alternative process mining algorithm based on regions in elementary net systems [7, 8, 6, 1] was proposed in [3, 4]. It is not the purpose of this paper to provide a detailed presentation of this algorithm for which we refer the reader to the above mentionned references and [2]. We just want to stress on the differences between the two approaches: *process discovery* (α algorithm) versus *process approximation* (ω algorithm). The latter algorithm produces a workflow net $\omega(W)$ whose language is the least language of a workflow net containing the log $W \subseteq T^*$.

Example 6. Let us consider the elementary net system N_1 depicted on left part of Fig. 3 with language $W = \{ACDE, BDCE\}$. This net is not a workflow net because places p_2 (respectively p_3) gets marked after the firing of sequence $ACDE$ (resp. $BDCE$) hence these places will not be synthesized by algorithm

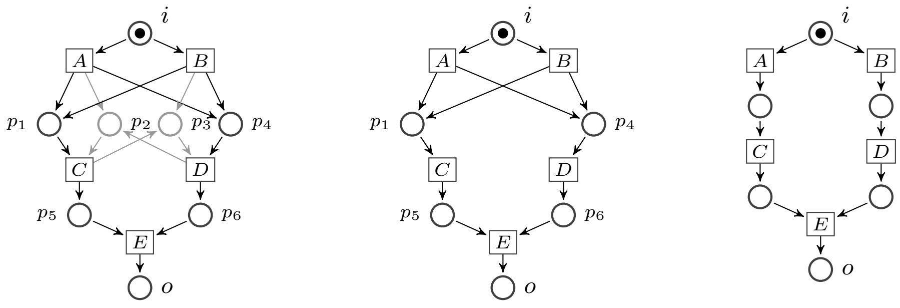


Fig. 3. an elementary net system (left) with language $W = \{ACDE, BDCE\}$, its workflow net approximation given by algorithm ω (center), and the workflow net synthesized by α from W (right)

ω. The net $N_2 = \omega(W)$ [3] shown in the center of Fig. 3 reproduces net N_1 without these two places; its language $\mathcal{L}(N_2) = \{ACDE, ADCE, BCDE, BDCE\}$ is the least language of a workflow net containing W. If we now apply the α-algorithm to the log W, we obtain $A\sharp_W B$, $A\sharp_W D$, $B\sharp_W C$, $C\|_W D$ and the immediate causalities $A \rightarrow_W C$, $B \rightarrow_W D$, $C \rightarrow_W E$, and $D \rightarrow_W E$. The resulting net $N_3 = \alpha(W)$, depicted on the right-part of Fig. 3, is a workflow net but $W = \{ACDE, BDCE\} \not\subseteq \mathcal{L}(N_3) = \{ACE, BDE\}$. α may fail to find a workflow net realizing all computation sequences in a given log W, i.e., a workflow net N such that $W \subseteq \mathcal{L}(N)$. In that case, α provides no solution of any kind. In contrast, ω always produces the optimal solution, i.e., a workflow net with the least possible language containing W.

Algorithm ω is also able to discover the workflow net N_1 of Exple. 5. This extended expressivity is however at the price of a higher computational complexity due to the fact that regions are global properties of the log whereas α considers only very local informations that are easy to extract: the α-algorithm is much faster and less space consuming that the ω-algorithm.

Since α does not provide useful information when it fails to discover a workflow net it is the more so important to be able to identify the class of α-reconstructible workflow nets. This was the motivation for restricting the α-algorithm to the context of *structured workflow nets*. The main result announced in [13] and proven in the report [12] is that structured workflow nets without short loops (a condition that we introduce in the next section) are α-reconstructible. Let us briefly introduce structured workflow nets. The reader may have observed that the workflow net N_1 of Exple. 5 is not a *free-choice* net [5]. The two choices between events A and B and between events D and E

[3]More precisely, the ω algorithm [2] is presented with two variants. In the first case, places are the so called ω-regions –regions that may appear as extensions of places of workflow nets–. A simplified, and language equivalent, version is obtained by restricting the inner places to be *minimal* regions. It is the simplified version that is depicted in Fig. 3.

are not independent: if one chooses A (resp. B), then one must choose D (resp. E). In fact, one cannot choose between events D and E at run time, since both events are never jointly enabled. Structured workflow nets satisfy a property slightly stronger that the free-choice property, that already excludes such interferences between conflict (the sharing of an input place by two transitions) and synchronization (the sharing of two input places by a transition).

Definition 7. *A workflow net $N = (P, T, F, M_0)$ is a structured workflow net if it has no structurally implicit places and the following condition holds:*

$$\forall t \in T \quad |{}^\bullet t| > 1 \Rightarrow (\forall p \in {}^\bullet t \quad |{}^\bullet p| = 1 \ \wedge \ |p^\bullet| = 1) \tag{SWN}$$

i.e., if a transition t requires the synchronization of several conditions (places), then each of these conditions has a unique cause ($|{}^\bullet p| = 1$) and a unique consequence ($|p^\bullet| = 1$), hence it cannot induce a conflict between t and another transition t'.

The main result established in [12] is the following.

Theorem 1. *Structured workflow nets without short loops are α-reconstructible.*

Adding structurally implicit places to a net preserves its language, and removing places from a net satisfying condition (SWN) cannot invalidate this condition. Therefore, one can state the following corollary to Th. 1.

Corollary 1. *A workflow net N without short loops and satisfying condition (SWN) is always language equivalent to some α-reconstructible workflow net N'.*

Condition (SWN) is a structural condition, hence it can be checked very efficiently. It was argued [13] that the class of nets satisfying this condition supports all basic routing patterns and building blocks used to construct workflow systems in practice. The conditions characterizing structured workflow nets (Def. 7) are sufficient, but however not necessary, to ensure α-reconstructibility. Actually, an α-reconstructible net may contain structurally implicit places (Exple. 7 below) and it may not satisfy condition (SWN) (Exple. 1).

Example 7. Consider transitions C, F, G in the workflow net N depicted on the left of Fig. 4. If we let $W = \mathcal{L}(N)$, then we get $C \sharp_W F$, $C \rightarrow_W G$, and $F \rightarrow_W G$.

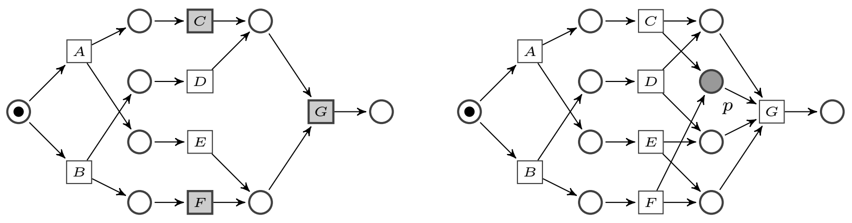


Fig. 4. a place p of an α-reconstructible net which is a structurally implicit place

Therefore, the α-algorithm necessarily produces a place in $C^\bullet \cap F^\bullet \cap {}^\bullet G$. This is indeed the place p that appears in the net $N' = \alpha(\mathcal{L}(N))$ shown on the right of Fig. 4. Another place in $D^\bullet \cap E^\bullet \cap {}^\bullet G$ appears symmetrically in N'. Now N and N' are language equivalent, and therefore, N' is α-reconstructible, and p is clearly a structurally implicit place of N'.

4 α Reconstructible Workflow Nets

In view of the preceding discussion we try in this section to obtain a more precise understanding of α-reconstructibility of workflow nets. The purpose of algorithm α is to deduce places of the net on the basis of the information $Abs(W)$ extracted from the event log. Notice that for every place p of an elementary net one has *(i)* $\forall a, a' \in {}^\bullet p \; a \sharp_N a'$, *(ii)* $\forall b, b' \in p^\bullet \; b \sharp_N b'$, and *(ii)* $\forall a \in {}^\bullet p \; \forall b \in p^\bullet \; a \rightarrow_N b$ where

$$\begin{aligned} t \rightarrow_N t' &\Leftrightarrow t^\bullet \cap {}^\bullet t' \neq \emptyset \\ t \,\sharp_N\, t' &\Leftrightarrow ({}^\bullet t \cap {}^\bullet t') \cup (t^\bullet \cap t'^\bullet) \neq \emptyset \\ t \,\|_N\, t' &\Leftrightarrow ({}^\bullet t \cup t'^\bullet) \cap ({}^\bullet t \cup t'^\bullet) = \emptyset \end{aligned}$$

In order to correctly infer the places of a workflow net N from the abstraction $Abs(\mathcal{L}(N))$ of its language (or of any of its complete log W) the above relations of causality, conflict and concurrency associated respectively with N and W should fit at best.

Proposition 1. *Let $W = \mathcal{L}(N)$ be the full log of a workflow net.*

1. *$t \rightarrow_W t' \Rightarrow t \rightarrow_N t'$, $t \|_W t' \Rightarrow t \|_N t'$, and $t \sharp_N t' \Rightarrow t \sharp_W t'$*
2. *if t and t' are co-enabled, i.e. there exists some reachable marking M such that $M[t\rangle$ and $M[t'\rangle$, then $t \|_N t' \Leftrightarrow t \|_W t'$ and $t \sharp_N t' \Leftrightarrow t \sharp_W t'$*

Proof. These properties are easily derived from the firing rule of elementary net systems using the fact that a workflow net is contact-free. More precisely one can successively check that

1. $(M[t \cdot t'\rangle \wedge M[t'\rangle) \Leftrightarrow (t \|_N t' \wedge {}^\bullet t \cup {}^\bullet t' \subseteq M \wedge M \cap (t^\bullet \cup t'^\bullet) = \emptyset)$ and one has $M[t' \cdot t\rangle$ and $M_{t \cdot t'} = M_{t' \cdot t}$ in this case.
2. $M[t \cdot t'\rangle \Leftrightarrow \neg(t \sharp_N t') \wedge {}^\bullet t \cup ({}^\bullet t' \setminus t^\bullet) \subseteq M \wedge M \cap (t^\bullet \cup (t'^\bullet \setminus {}^\bullet t)) = \emptyset$
3. From $M[t \cdot t'\rangle$ and $t^\bullet \cap {}^\bullet t' = {}^\bullet t \cap t'^\bullet = \emptyset$ it follows that $M[t'\rangle$ and hence $t \|_N t'$.
4. If N is contact-free, then $(M[t \cdot t'\rangle \wedge t^\bullet \cap {}^\bullet t' = \emptyset) \Rightarrow M[t'\rangle$
5. Then $t \sharp_N t' \Rightarrow t \sharp_W t'$, and $t \sharp_N t' \Leftrightarrow t \sharp_W t'$ if t and t' are co-enabled, i.e., $M[t\rangle$ and $M[t'\rangle$ for some reachable marking M.
6. The following relations hold if N is contact-free:
   (a) $t \rightarrow_W t' \Rightarrow t \rightarrow_N t'$,
   (b) $t \|_W t' \Rightarrow t \|_N t'$,
   (c) if t and t' are co-enabled then $t \|_N t' \Leftrightarrow t \|_W t'$. □

The following example illustrates the mismatch between the structural ($\rightarrow_N$) and behavioural ($\rightarrow_W$) relations of causality that can arise from contact situations.

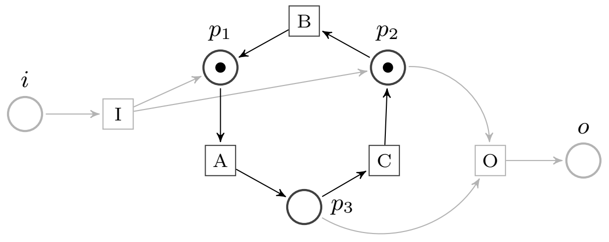


Fig. 5. a workflow with (many) contact situations, the marking indicated in this picture is the marking reached after the firing of the initial transition I

Example 8. The language of the workflow net depicted in Fig. 5 is $I(ABC)^*AO$. A complete log of this language is given by $W = \{IABCAO\}$ from which we deduce the following causal dependencies: $A \rightarrow_W B$, $B \rightarrow_W C$, and $C \rightarrow_W A$. The relations induced by the net system report (fake) immediate causalities in the reverse direction: $B \rightarrow_N A$, $A \rightarrow_N C$, and $C \rightarrow_N B$.

It can be inferred from the non emptiness of $t^\bullet \cap {}^\bullet t'$ that transition t is an immediate cause of t' only if one can find at least one place $p \in t^\bullet \cap {}^\bullet t'$ that is used as a ressource for t' and not for the purpose of disabling transition t. This is the rationale for requiring workflow nets to be contact-free.

The inclusion $\rightarrow_{\mathcal{L}(N)} \subseteq \rightarrow_N$ shows that the causal relation inferred from the observations $t \cdot t' \in C_{\mathcal{L}(N)}$ and $t' \cdot t \notin C_{\mathcal{L}(N)}$ (according to Def. 2) implies the existence of a connecting place $p \in t^\bullet \cap {}^\bullet t'$. As just noticed contact-freeness avoids the production of connecting places representing fake dependencies. But this restriction is not sufficient to guarantee that the converse inclusion $\rightarrow_N \subseteq \rightarrow_{\mathcal{L}(N)}$ holds. In particular, as illustrated by the following example, cyclic dependencies given by short loops is another obstacle to α-reconstructibility.

Definition 8. *Two transitions of a (contact-free) elementary net system form a* short loop *if $t^\bullet \cap {}^\bullet t' \neq \emptyset$ and $t'^\bullet \cap {}^\bullet t \neq \emptyset$.*

Example 9. The workflow net N of Fig. 6 has a short loop involving transitions B and C. A complete log of N is $W = \{ABCBD\}$. The abstraction of this log

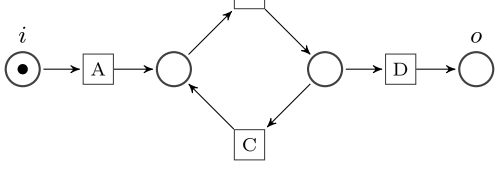


Fig. 6. a workflow net with a short loop

is $Abs(W) = \{\{A\}, \{AB, BC, CB, BD\}, \{D\}\}$. Since both short sequences BC

and CB belong to C_W, one gets $B \parallel_W C$. Thus cyclic dependencies within short loops are lost when applying the α-algorithm. The net system synthesized by the function Syn from $Abs(W)$ is shown in Fig. 7. This net is not a workflow net since the transition C is isolated, hence α-algorithm fails in that case.

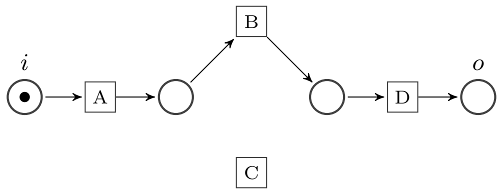


Fig. 7. the net system constructed by the algorithm α from the complete log $W = \{ABCBD\}$ of the workflow net of Fig. 6

Proposition 2. *If N is a workflow net without short loops then*

$$t^\bullet \cap {}^\bullet t' \neq \emptyset \quad \Rightarrow \quad t' \cdot t \notin C_{\mathcal{L}(N)}$$

Proof. $t^\bullet \cap {}^\bullet t' \neq \emptyset$ implies $t'^\bullet \cap {}^\bullet t = \emptyset$ (no short loops). Suppose by way of contradiction that $t' \cdot t \in C_{\mathcal{L}(N)}$. Then there exists a reachable marking M such that $M[t' \cdot t\rangle$. Since $t'^\bullet \cap {}^\bullet t = \emptyset$ and the net is contact-free we have $M[t\rangle$ and thus $t \|_N t'$ which is in contradiction with $t^\bullet \cap {}^\bullet t' \neq \emptyset$. $\square$

In view of Proposition 2, in order to establish $\rightarrow_N \subseteq \rightarrow_{\mathcal{L}(N)}$ (and thus $\rightarrow_N = \rightarrow_{\mathcal{L}(N)}$ since the converse implication holds by Prop. 1) it remains to ensure that $t^\bullet \cap {}^\bullet t' \neq \emptyset$ implies $t \cdot t' \in C_{\mathcal{L}(N)}$. This condidition can be restated as the following requirement on places.

Definition 9. *An inner place p of a workflow net is said to be a* boundary *place when:* $\quad \forall t \in {}^\bullet p \quad \forall t' \in p^\bullet \qquad t \cdot t' \in C_{\mathcal{L}(N)}$.

For instance place p (or similarly place p') of workflow net N_1 depicted in Fig. 1 is not a boundary place since ${}^\bullet p = \{A\}$, $p^\bullet = \{D\}$, and there is no firing sequence of this net in which A is immediately followed by D. Indeed, the two non-boundary places p and p' cannot be discovered by algorithm α (Fig. 2).

Corollary 2. *Let N be a workflow net without short loops and all of whose inner places are boundary places, then $\rightarrow_N = \rightarrow_{\mathcal{L}(N)}$.*

From the definition of the structural relations of causality $\rightarrow_N$ and conflict $\sharp_N$ associated with a net system N, it follows that for every place p one has

1. $\forall t \in {}^\bullet p \, \forall t' \in p^\bullet \;\; t \rightarrow_N t'$,
2. $\forall t, t' \in {}^\bullet p \;\; t \sharp_N t'$, and
3. $\forall t, t' \in t^\bullet \;\; t \sharp_N t'$.

Using $\sharp_N \subseteq \sharp_{\mathcal{L}(N)}$ (by Prop. 1) and $\rightarrow_N \subseteq \rightarrow_{\mathcal{L}(N)}$ (by Cor. 2) we deduce the following result.

Corollary 3. *$^\bullet p \prec p^\bullet$ for every place of a workflow net N without short loops and whose inner places are boundary places, where $\prec = \prec_{\mathcal{L}(N)}$ (see Def. 4)* [4].

Elements of the (graph of) relation $\prec = \{(A, B) \in \wp(T) \times \wp(T) \mid A \prec B\}$ are generalizations of the (extents of) places of the net to be discovered. In analogy with regions in transition systems, we might called them α-regions. The pair $(^\bullet p, p^\bullet)$, called the *α-extent* of place $p \in P$, belongs (by above Cor. 3) to relation $\prec$ when p is a boundary place. Moreover when equipped with the component-wise order relation $(A, B) \sqsubseteq (A', B') \Leftrightarrow A \subseteq A' \wedge B \subseteq B'$ the set $\prec$ is downward-closed:

$$(A \prec B \ \wedge\ A' \subseteq A \ \wedge\ B' \subseteq B) \ \Rightarrow\ A' \prec B'$$

The fact that p is a boundary place yields that $A \prec B$ for every $A \subseteq {}^\bullet p$ and $B \subseteq p^\bullet$. We say that place p *justifies* the corresponding observation $A \prec B$. In order to synthesize a net from $Abs(\mathcal{L}(N))$ we have to find enough abstract places (α-regions) for justifying every element $A \prec B$ in that relation. This led us to consider the following class of workflow nets:

Definition 10. *A workflow net without short loops and all of whose inner places are boundary places is said to be an α-workflow net when a place p exists such that $A \subseteq {}^\bullet p$ and $B \subseteq p^\bullet$ whenever $A \prec B$.*

The main result of this section is the following theorem, which sums up the various elements presented so far in the course of this section.

Theorem 2. *An α-workflow net N whose places have incomparable α-extents (i.e., $^\bullet p \subseteq {}^\bullet q \ \wedge\ p^\bullet \subseteq q^\bullet \quad \Rightarrow \quad p = q$) is α-reconstructible; i.e., $N \cong \alpha(W)$ for any complete log $W \subseteq \mathcal{L}(N)$ of N.*

Proof. Let N be an α-workflow net whose places have incomparable extents for the order relation $\sqsubseteq$. We have $i^\bullet = I_{\mathcal{L}(N)}$ and these transitions can occur only as the first elements of execution sequences, symetrically $^\bullet o = O_{\mathcal{L}(N)}$ and these transitions can occur only as the last transitions of execution sequences. Thus we also have $i^\bullet = I_W$ and $^\bullet o = O_W$ since $W \subseteq \mathcal{L}(N)$ contains at least one occurrence of each transition. Let us now proceed to a comparison of the respective sets of inner places of nets N and $\alpha(W)$. Cor. 3 states that the extents of inner places of N are elements of relation $\prec = \prec_W$. The condition stated in Def. 10 ensures that all maximal elements of $\prec_W$ are extents of places. Therefore the extents of places are exactly the maximal elements of $\prec_W$, i.e. they are the places of $\alpha(W)$, if and only if they are incomparable for order relation $\sqsubseteq$. $\square$

Notice that, by definition, the places of $\alpha(W)$ have incomparable extents. Thus *an α-workflow net is α-reconstructible if and only if it has incomparable places.* More significantly, we establish in the next section a converse to Theo. 2 which allows to *identify α-reconstructible workflow nets with those α-workflow nets whose places have incomparable extents.*

[4] We have also $\prec = \prec_W$ for any complete log $W \subseteq \mathcal{L}(N)$ of N.

Example 10 (example 4 continued). The extents of places p and p' of Exple. 4 are not maximal elements w.r.t. $\sqsubseteq$ since $({}^\bullet p, p^\bullet) \sqsubseteq ({}^\bullet q, q^\bullet)$ and $({}^\bullet p', p'^\bullet) \sqsubseteq ({}^\bullet q', q'^\bullet)$. The language of the net $N' = \alpha(W)$ synthesized from its complete log W is not the least language of a workflow net containing W (since N, which is also an α-workflow net, gives in that respect a better approximation). But N', unlike N, is α-reconstructible.

5 Boundary Places of a Workflow Net

In this section we provide a characterization of the boundary places and we describe their relationship with the non structurally implicit places. For some classes of net systems these two notions coincide.

Proposition 3. *Let N be a workflow net that satisfies Condition (SWN). An inner place of N is a boundary place if and only if it is a non structurally implicit place.*

Proof. The key argument is the observation that for any pair of transitions t and t' of a workflow net that satisfies Condition (SWN) the set $t^\bullet \cap {}^\bullet t'$ contains at most one place. Let p be a boundary place of N, then there exists accessible markings M and M' such that $M[t\rangle M'[t'\rangle$ and marking M satisfies $p \notin M$ and ${}^\bullet t' \setminus \{p\} \subseteq M$ (since none of these places belong to $t^\bullet$ by the preceding remark). Hence p is a non structurally implicit place. Conversely let p be a non structurally implicit place of N.

1. If $|p^\bullet| > 1$ then for every $t' \in p^\bullet$ one has ${}^\bullet t' = \{p\}$. Since the net is initially life every $t \in {}^\bullet p$ is enabled in some reachable marking M, and since the net is contact-free t' is enabled in the marking M' such that $M[t\rangle M'$. Hence $t \cdot t' \in C_{\mathcal{L}(N)}$ and p is a boundary place.
2. If $p^\bullet = \{t'\}$ and ${}^\bullet t' = \{p\}$ then the same argument applies: $t \in {}^\bullet p$ is enabled in some reachable marking M, and since the net is contact-free t' is enabled in the marking M' such that $M[t\rangle M'$. Hence $t{\cdot}t' \in C_{\mathcal{L}(N)}$ and p is a boundary place.
3. If $p^\bullet = \{t'\}$ and $|{}^\bullet t'| > 1$ then $p'^\bullet = \{t'\}$ for every $p' \in {}^\bullet t'$ and $|{}^\bullet p| = 1$, say ${}^\bullet p = \{t\}$. Let M be a reachable marking such that ${}^\bullet t' \setminus \{p\} \subseteq M$ and $p \notin M$. Since places in ${}^\bullet t'$ are inner places they should be emptied by a transition enabled in a marking M' reachable from M. This transition is necessarily t' and the firing of t' requires that p be filled, hence t be fired beforehand, and t' can then fired immediately after t. Hence p is a boundary place. □

The input and output places are non implicit and we could equivalently have defined structured workflow nets as workflow nets satisfying Condition (SWN) and all whose inner places are boundary places. However in the general case these two notions differ. For instance the place p and p' in the workflow net of Exple. 5 are neither boundary places nor structurally implicit places, and the place p in net N' of Exple. 7 is both a boundary place and a structurally implicit place.

Let us now proceed to the characterization of boundary places. The language of a workflow net N is closed under the congruence $\sim$ generated by the relations $t \cdot t' \sim t' \cdot t$ pertaining to pairs of concurrent transitions t and t' $(t \|_N t')$. An equivalence class of maximal execution sequences of N is called a *process* of N. Processes of a workflow net N may be represented equivalently as follows.

Definition 11. *A* process *of a workflow net $N = (P, T, F, M_0)$ is a pair $\mathcal{R} = (R, \ell)$ consisting of a net $R = (P_R, T_R, F_R)$ and two labelling functions $\ell : T_R \to T$ and $\ell : P_R \to \wp(P)$ satisfying the following conditions:*

1. *There is a place i_R such that ${}^\bullet i_R = \emptyset$, and $\ell(i_R) = \{i\}$ where i is the input place of the workflow net N.*
2. *There is a place o_R such that ${o_R}^\bullet = \emptyset$, and $\ell(o_R) = \{o\}$ where o is the output place of the workflow net N.*
3. $\forall p_R \in P_R \setminus \{i_R, o_R\} \qquad |{}^\bullet p_R| = 1 \quad$ *and* $\quad |{p_R}^\bullet| = 1$.
4. $\forall t_R \in T_R \qquad {}^\bullet t_R \neq \emptyset \quad$ *and* $\quad {t_R}^\bullet \neq \emptyset$.
5. *The underlying graph of R is acyclic.*
6. $\{\ell(p_R) \mid p_R \in {}^\bullet t_R\}$ *is a partition of* ${}^\bullet \ell(t_R)$.
7. $\{\ell(p_R) \mid p_R \in {t_R}^\bullet\}$ *is a partition of* $\ell(t_R)^\bullet$.

The above definition differs slightly from the usual definition of processes as occurrence nets [10]. The sole difference is that here, *each place in a process R is mapped by ℓ to a set of places of N, playing indistinguishable roles in this process.* As a result, processes are free from equivalent places. In the sequel, we let $\ell(M_R) = \bigcup \{\ell(p_R) \mid p_R \in M_R\}$ denote the marking of N associated by ℓ with the marking M_R of R.

Proposition 4. *Processes $\mathcal{R} = (R, \ell)$ of a workflow net N are in bijective correspondence with the equivalences classes of complete execution sequences of N modulo permutation of concurrent transitions.*

The proof of this proposition is delayed until some additional results have been proved starting with the following lemma.

Lemma 1. *If $\mathcal{R} = (R, \ell)$ is a process of a workflow net N then*

$$\{i_R\}\,[t_1 \cdots t_k\rangle\,\{o_R\} \;\; \textit{in } R \quad \Leftrightarrow \quad \{i\}\,[\ell(t_1) \cdots \ell(t_k)\rangle\,\{o\} \;\; \textit{in } N$$

Proof. More generally for $X \subseteq T_R$ we let $\ell(X) = \bigcup \{\ell(p_R) \mid p_R \in X\}$, in particular $\ell({}^\bullet t_R) = {}^\bullet \ell(t_R)$ and $\ell({}^\bullet t_R) = {}^\bullet \ell(t_R)$ for every $t_R \in T_R$. $M_R[t_R\rangle M'_R$ in R iff $M_R \setminus M'_R = {}^\bullet t_R$ and $M'_R \setminus M_R = {t_R}^\bullet$ iff $\ell(M_R \setminus M'_R) = \ell(M_R) \setminus \ell(M'_R) = {}^\bullet \ell(t_R)$ and $\ell(M'_R \setminus M_R) = \ell(M'_R) \setminus \ell(M_R) = \ell(t_R)^\bullet$ iff $\ell(M_R)[\ell(t_R)\rangle \ell(M'_R)$ in N. Thus $\{i_R\}\,[t_1 \cdots t_k\rangle\,\{o_R\}$ in R iff $\{i\}\,[\ell(t_1) \cdots \ell(t_k)\rangle\,\{o\}$ in N. $\square$

We recall that if two firing sequences of the form $u \cdot t \cdot t' \cdot v$ and $u \cdot t' \cdot t \cdot v$ are both enabled in the initial marking of an elementary net system N, then one necessarily has $t \|_N t'$ and these sequences are permutation equivalent.

Corollary 4. *Let $\mathcal{R} = (R, \ell)$ be a process of a workflow net N. The set $\mathcal{L}_\mathcal{R} = \{\ell(t_1) \cdots \ell(t_n) \mid t_1 \cdots t_n \in \mathcal{L}(R)\}$ where $\mathcal{L}(R)$ is the set of firing sequences of R starting in $\{i_R\}$ and ending in $\{o_R\}$ is closed by permutation of concurrent transitions.*

Lemma 2. *Any firing sequence $u = t_1 \cdots t_k \in \mathcal{L}(N)$ of a workflow net N can be associated with a process $\mathcal{R} = (R, \ell)$ of N defined as follows. We let $T_R = \left\{\tilde{t}_1, \cdots, \tilde{t}_k\right\}$ with $\ell(\tilde{t}_i) = t_i$. For $1 \leq i < j \leq k$ we let*

$$P_{i,j} = \{p \in t_i{}^\bullet \cap {}^\bullet t_j \mid \forall i < j' < j \;\; p \notin {}^\bullet t_{j'}\}$$

We further let P_R stand for the set $\left\{p'_{i,j} \mid P_{i,j} \neq \emptyset\right\}$ together with two additional places, i_R and o_R, where

- *$\ell(i_R) = i$, and $\ell(o_R) = o$,*
- *$\ell(\tilde{p}_{i,j}) = P_{i,j}$,*
- *${}^\bullet\tilde{t}_1 = i_R$, and ${}^\bullet\tilde{t}_j = \{\tilde{p}_{i,j} \mid 1 \leq i < j\}$ for $1 < j \leq k$, and*
- *$\tilde{t}_i{}^\bullet = \{\tilde{p}_{i,j} \mid i < j \leq k\}$ for $1 \leq i < k$, and $\tilde{t}_k{}^\bullet = \{o_R\}$.*

Proof. Indeed the non empty sets $P_{i,j}$ for $j \in \{i+1, \ldots, k\}$ form a partition of $t_i{}^\bullet$ and the non empty sets $P_{i,j}$ for $i \in \{1, \ldots, j-1\}$ form a partition of ${}^\bullet t_j$ and $\mathcal{R} = (R, \ell)$ satisfies the conditions in Def. 11. □

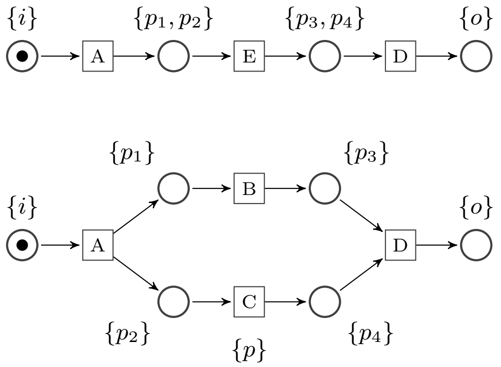
Proof (of Prop. 4). In view of the two preceding lemmas and Corollary 4 it just remains to prove that any two sequences in $\mathcal{L}(R)$ are permutation equivalent. We note that R is a workflow net, in particular it is contact-free. Moreover, since it contains no conflict it is a persistent net: once a transition is enabled it remains enabled until it is actually fired. Since the net is acyclic a place is never filled twice and a transition is fired exactly once along any maximal firing sequence. It follows by induction that any two sequences in $\mathcal{L}(R)$ are permutation equivalent. □

Proposition 5. *An inner place p of a workflow net is a boundary place if and only if for every pair of transitions $t \in {}^\bullet p$ and $t' \in p^\bullet$ there exists a process $\mathcal{R} = (R, \ell)$ of N and a* ***non (structurally) implicit*** *place $p_R \in P_R$ in this process with $p_R \in t_R{}^\bullet \cap {}^\bullet t'_R$ such that $\ell(t_R) = t$, $\ell(t'_R) = t'$ and $p \in \ell(p_R)$.*

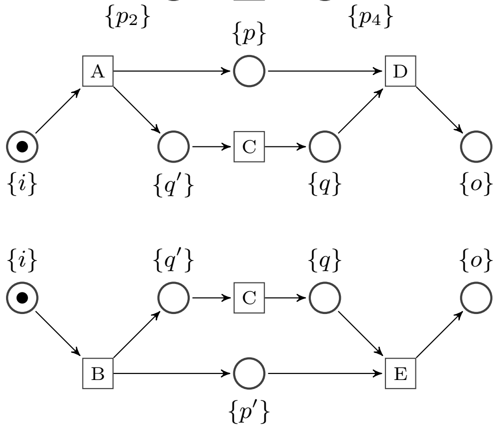
Proof. Let p a boundary place of a workflow net N. For every $t \in {}^\bullet p$ and $t' \in p^\bullet$ there exists a firing sequence $u = t_1 \cdots t_k \in \mathcal{L}(N)$ such that $t = t_i$ and $t' = t_{i+1}$ for some index i. Let $\mathcal{R} = (R, \ell)$ be the process of N associated with u as described in Lemma 2. Since $p \in P_{i,i+1}$ the place $\tilde{p}_{i,i+1}$ and the transitions $\tilde{t}_i$ and $\tilde{t}_{i+1}$ of the process satisfy $\ell(\tilde{t}_i) = t_i$, $\ell(\tilde{t}_{i+1}) = t_{i+1}$, and $p \in \ell(\tilde{p}_{i,i+1}) = P_{i,i+1}$. Since $\tilde{p}_{i,i+1}$ is the unique element of $\tilde{t}_i{}^\bullet \cap {}^\bullet\tilde{t}_{i+1}$ every input places of $\tilde{t}_{i+1}$ but $\tilde{p}_{i,i+1}$ are marked after firing the sequence $t_1 \ldots t_i$. Therefore place $\tilde{p}_{i,i+1}$ is not an implicit place. Conversely assume that for every pair of transitions $t \in {}^\bullet p$ and $t' \in p^\bullet$ there exists a process $\mathcal{P} = (R, \ell)$ of N and a non implicit place $p_R \in P_R$ in this process with $p_R \in t_R{}^\bullet \cap {}^\bullet t'_R$ such that $\ell(t_R) = t$, $\ell(t'_R) = t'$ and $p \in \ell(p_R)$. There exists a reachable marking where every input place of t'_R but

p_R are marked, then t'_R becomes enabled as soon (and only when) transition t_R fires and t'_R can fire immediately then. By Lemma 1 we can deduce the existence of a firing sequence of N where t is immediately followed by t' thus showing that p is a boundary place. □

Example 11.
The workflow net of Exple. 1 has two complete processes given next from which we deduce that all of its inner places are boundary places. Actually these two processes contains no implicit places and they allow to cover all possible triples (t, p, t') with $p \in t^\bullet \cap {}^\bullet t'$ from the workflow net.

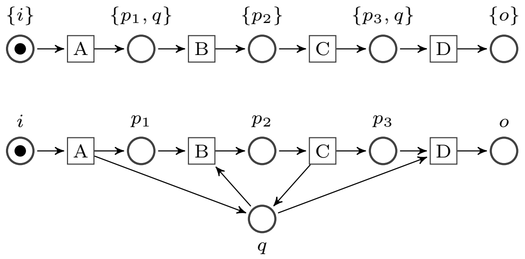


The workflow net of Exple. 5 has two complete processes given next. Place $\{p\}$ is an implicit place of the first process since it is marked in all reachable markings that contain place $\{q\}$. Similarly place $\{p'\}$ is an implicit place of the second process. We deduce that places p and p' in the original workflow net are not boundary places (even though they are not implicit places).



If $p \in t^\bullet \cap {}^\bullet t'$ is a boundary place of a workflow net N and $\tilde{p}$ is a non implicit place and $\tilde{t}$ and $\tilde{t'}$ are transitions of a process $\mathcal{R} = (R, \ell)$ of N such that $\ell(\tilde{t}) = t$, $\ell(t') = t'$, and $p \in \ell(\tilde{p})$, then $\ell(\tilde{p}) \subseteq t^\bullet \cup {}^\bullet t'$. The following example shows that this inclusion may be strict.

Example 12.
The workflow net shown next (below) has a unique process (the net system shown just above it). Place $q \in A^\bullet \cap {}^\bullet D$ is not a boundary place since there is no place $q_R \in A^\bullet \cap {}^\bullet D$ in the corresponding process.



We then conclude with the characterization of α-reconstructibility.

Theorem 3. *A workflow net is α-reconstructible if and only if it is an α-workflow net whose places are incomparable for $\sqsubseteq$.*

Proof. Theo. 2 shows that the given conditions are sufficient. It remains to show that the absence of short loops and the fact that inner places are boundary places are necessary conditions for α-reconstructibility. First, by definition of the construction α, an inner place p of the synthesized net is a boundary place. The justification is as follows. For every $t \in {}^\bullet p$ and $t' \in p^\bullet$ we have $t \cdot t' \in C_{\mathcal{L}(N)}$ (and also $t' \cdot t \notin C_{\mathcal{L}(N)}$), i.e. there exists some execution sequence of the form $\sigma = u \cdot t \cdot t' \cdot u' \in \mathcal{L}(N)$. The complete process R associated with this execution sequence has transitions $\tilde{t}$, $\tilde{t'}$ and places $\tilde{p}$ that lift respectively t, t', and p; i.e. such that $\ell(\tilde{t}) = t$, $\ell(\tilde{t'}) = t'$, ${}^\bullet\tilde{p} = \{\tilde{t}\}$, $\tilde{p}^\bullet = \{\tilde{t'}\}$, and $p \in \ell(\tilde{p})$. Since $\tilde{t'}$ can fired immediately after $\tilde{t}$ in this process we deduce that $\tilde{p}$ is a non-implicit place of R, hence place p is a boundary place. Now let us assume that N is a workflow net all of whose inner places are boundary places and containing a short loop given by two transitions t_1 and t_2 such that ${t_1}^\bullet \cap {}^\bullet t_2 \neq \emptyset$ and ${t_2}^\bullet \cap {}^\bullet t_1 \neq \emptyset$. Since the places in ${t_1}^\bullet \cap {}^\bullet t_2$ and ${t_2}^\bullet \cap {}^\bullet t_1$ are boundary places we deduce that $t_1 \cdot t_2$ and $t_2 \cdot t_1$ both belong to $C_{\mathcal{L}(N)}$ and thus $t_1 \|_{\mathcal{L}(N)} t_2$. It follows that the sets ${t_1}^\bullet \cap {}^\bullet t_2$ and ${t_2}^\bullet\ cap^\bullet t_1$ are both empty when these flow relations are taken in the synthesized net $\alpha(N)$ and therefore N is not isomorphic to $\alpha(\mathcal{L}(N))$ hence it is not α-reconstructible. □

6 Conclusion

The starting point of this work was the desire to achieve a characterization of the class of α-reconstructible workflow nets: even though the original presentation by van der Aalst *et al* was restricted to structured workflow nets it was clear that their method could be applied to a much wider class of nets. In particular none of the two properties shared by structured workflow nets, namely the absence of implicit places and a variant of free-choice property, are necessary to guarantee α-reconstructibility. The main condition for α-reconstructibility states that all (inner) places of the workflow net are boundary places. Boundary places and non implicit places coincide in the context of structured workflow nets but these notions diverge in general. By making this distinction explicit we hope to have gained a better understanding of α-reconstructibility. From a practical point of view however this characterization has probably a limited interest. The fact that the obtained conditions are not structural properties, in contrast with structured workflow nets, makes their verification more involved. More specifically we have to associate a workflow net with a finite set of "patterns" from which all of its processes can be generated.

The main goal of α is the exact reconstruction of processes from sets of execution sequences, and α is particularly good at achieving this goal from complete logs (that need not be full logs) of structured workflow nets. Without these assumptions, α may behave in an unexpected way. For instance, the synthesized net may fail to reproduce some of the execution sequences from the input log.

When using region-based net synthesis algorithms, one does not make any assumption on W. One just tries to construct for all W the simplest net model that contains all sequences in this set. Sobriety, which is crucial to process reconstruction algorithms, has minor importance in this different perspective. The α and ω algorithms are the two extreme of a range of process mining algorithms. By resorting only to local information α is more efficient but less expressive than ω. There is potentially room for the design of intermediate mining algorithms.

References

1. Eric Badouel and Philippe Darondeau. Theory of regions. In Reisig and Rozenberg [9], pages 529–586.
2. Eric Badouel and Philippe Darondeau. *Petri Net Synthesis.* (Book in preparation), 2013.
3. Nadia Busi and G. Michele Pinna. Characterizing workflow nets using regions. In *SYNASC*, pages 399–406. IEEE Computer Society, 2006.
4. Nadia Busi and G. Michele Pinna. Process discovery and petri nets. *Mathematical Structures in Computer Science*, 19(6):1091–1124, 2009.
5. Jörg Desel and Javier Esparza. *Free Choice Petri Nets.* Cambridge Tracts in Theoretical Computer Science, 40. Cambridge University Press, 1995.
6. Jörg Desel and Wolfgang Reisig. The synthesis problem of petri nets. In Patrice Enjalbert, Alain Finkel, and Klaus W. Wagner, editors, *STACS*, volume 665 of *Lecture Notes in Computer Science*, pages 120–129. Springer, 1993.
7. Andrzej Ehrenfeucht and Grzegorz Rozenberg. Partial (set) 2-structures. part i: Basic notions and the representation problem. *Acta Inf.*, 27(4):315–342, 1989.
8. Andrzej Ehrenfeucht and Grzegorz Rozenberg. Partial (set) 2-structures. part ii: State spaces of concurrent systems. *Acta Inf.*, 27(4):343–368, 1989.
9. Wolfgang Reisig and Grzegorz Rozenberg, editors. *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, volume 1491 of *Lecture Notes in Computer Science.* Springer, 1998.
10. Grzegorz Rozenberg and Joost Engelfriet. Elementary net systems. In Reisig and Rozenberg [9], pages 12–121.
11. Wil M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes.* Springer, 2011.
12. Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Which processes can be rediscovered?, 2002. BETA Working Paper Series, WP 74, Eindhoven University of Technology, Eindhoven.
13. Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.
14. Glynn Winskel. Event structures. In Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Advances in Petri Nets*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer, 1986.
15. Glynn Winskel. An introduction to event structures. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *REX Workshop*, volume 354 of *Lecture Notes in Computer Science*, pages 364–397. Springer, 1988.