Acta Informatica (2011) 48:213–242
DOI 10.1007/s00236-011-0137-8

Connectivity of Workflow Nets: The Foundations of Stepwise Verification

Artem Polyvyanyy · Matthias Weidlich · Mathias Weske

Received: 1 October 2010 / Accepted: 26 May 2011 / Published online: 29 June 2011
Author's version

Abstract Behavioral models capture operational principles of real-world or designed systems. Formally, each behavioral model defines the state space of a system, i.e., its states and the principles of state transitions. Such a model is the basis for analysis of the system's properties. In practice, state spaces of systems are immense, which results in huge computational complexity for their analysis. Behavioral models are typically described as executable graphs, whose execution semantics encodes a state space. The structure theory of behavioral models studies the relations between the structure of a model and the properties of its state space.

In this article, we use the connectivity property of graphs to achieve an efficient and extensive discovery of the compositional structure of behavioral models; behavioral models get stepwise decomposed into components with clear structural characteristics and inter-component relations. At each decomposition step, the discovered compositional structure of a model is used for reasoning on properties of the whole state space of the system. The approach is exemplified by means of a concrete behavioral model and verification criterion. That is, we analyze workflow nets, a well-established tool for modeling behavior of distributed systems, with respect to the soundness property, a basic correctness property of workflow nets. Stepwise verification allows the detection of violations of the soundness property by inspecting small portions of a model, thereby considerably reducing the amount of work to be done to perform soundness checks. Besides formal results, we also report on findings from applying our approach to an industry model collection.

Keywords Behavioral models, structure theory, connectivity, workflow nets, verification, soundness

Business Process Technology Group
Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Str. 2–3, D-14482 Potsdam, Germany
(Artem.Polyvyanyy,Matthias.Weidlich,Mathias.Weske)@hpi.uni-potsdam.de

1 Introduction

Behavioral models are conceptual models that capture operational principles of real-world or designed systems. The behavior of a system is defined by all possible states of the system, together with principles of the state transitions that the system is capable of. Behavioral models are typically expressed as directed graphs where nodes encode states and directed edges specify principles of state transitions. These graphs expose the compositional structure of behavioral models. The structure theory of behavioral models studies relations between structure of a model and properties of the state space that it specifies. Business process models capture the essence of how work is performed in organizations and, as such, can be addressed as behavioral models. Business process models are important for understanding the implications of business activities, for suggesting improvements, and for providing a blueprint for the implementation of business procedures in a company [1]. While the Business Process Model and Notation (BPMN) [2] is regarded as the de facto standard for process modeling, other approaches like EPC [3] and UML activity diagrams [4] are still in use.

Since the execution semantics of business process modeling languages is described rather informally, formal investigations rely on a mapping to formal languages. Petri nets, cf., [5], are a modeling language for describing concurrency in distributed systems with precise execution semantics and established analysis techniques. Common practice is to define the execution semantics of business process modeling languages by mapping their modeling constructs to Petri nets, e.g., [6,7,8,9]. Workflow nets are a structural subclass of Petri nets with a dedicated source place and a dedicated sink place, marking the start and end of the process, respectively. *Soundness* is a desirable correctness property of workflow nets [10], since a sound workflow net always terminates properly and each task can contribute to the completion of a process. Consequently, a business process is considered to be correct if the corresponding workflow net is sound.

Soundness verification is a well-studied problem. A basic technique for soundness verification is state space exploration. However, state space exploration suffers from the state space explosion problem, as the number of reachable states can be exponential in the size of the model. In this article, we use structural analysis of workflow nets to investigate soundness, providing insight into an alternative – and in many cases, preferable – way to check soundness. In particular, we employ the connectivity property of workflow nets. A workflow net gets decomposed into components based on its separating sets, i.e., sets of nodes of the net that when removed yield the net disconnected. We proceed stepwise, i.e., by gradually increasing the size of investigated separating sets. Based thereon, we point out how the soundness verification can be organized from the derived components of a workflow net. Where applicable, we draw conclusions on soundness for the general class of workflow nets; otherwise, the results are restricted to safe or free-choice nets. At each step of the decomposition, we argue about the algorithmic complexity of reaching the step and suggest diagnostic information that can be presented to process analysts as feedback on flaws in the behavior of a model. Though recent works show impressive results in efficiency of the soundness verification of industrial models, cf., [11], the results were achieved under the assumption of free-choiceness of models. Efficient soundness verification in the case of general nets will, however, allow one to verify models which contain advanced workflow patterns. Despite the variety of existing soundness verification techniques, the efficiency and the structural diagnostic information for the general class of workflow nets are unique characteristics of our approach, coming at the expense of verification completeness. We see a great potential

in combining our approach with existing techniques on soundness verification to achieve more mature and complete process model verification.

This article is based on previous results. In [12], we initially discussed the idea of using the connectivity property of behavioral models to guide their analysis. First results on applying this idea for soundness verification of workflow nets have been presented in [13]. Still, we elaborated solely on the decomposition which relies on separating sets composed of a single node, viz., the biconnected decomposition of workflow nets. This article goes beyond these initial results by investigating decompositions which are based on separating sets of size two, i.e., the step of the triconnected decomposition, and three, i.e., the step of the 4-connected decomposition. While the triconnected step incorporates two results that have been shown for the triconnected components of free-choice workflow nets in [14], we present novel results with respect to pruning and compositionality of the biconnected workflow nets. All results for the 4-connected step have not been discussed before. Besides the formal results on soundness verification for each of the decomposition steps, our contribution is an comprehensive approach that features diagnostic information on identified behavioral issues along with a verification algorithm incorporating the separate verification results. We also report on findings of applying our approach to a collection of real-world industrial process models.

The remainder of this article is structured as follows: The next section gives preliminaries for our work in terms of basic definitions for Petri nets, workflow nets, and soundness of workflow nets. Section 3 presents our main results. In this section, we employ connectivity property of workflow nets to perform soundness verification, i.e., subnets and their compositional relations obtained during connectivity-based decomposition of nets are employed for verifying the correctness of the whole system. Section 4 elaborates on the application and on the evaluation of our results. We present a verification algorithm and report on results of the application of our approach for an industry model collection. Section 5 generalizes the principles of the connectivity-based verification of workflow nets into a comprehensive framework for connectivity-based decomposition of behavioral models; the framework should be understood as general guidelines for performing structural analysis of behavioral models. Section 6 reviews related work. Finally, Section 7 concludes the article.

2 Preliminaries

This section introduces Petri nets – a well-known formalism for modeling distributed systems, cf., [5]. A Petri net has a structure given by a net, a marking that represents a state of the net, and the execution semantics that describes the principles of state transitions. Afterwards, we present WF-nets – the subclass of nets which in this article is in the focus of investigations for structural characteristics of behavioral correctness.

Definition 1 (Petri net)
A *Petri net*, or a *net*, $N = (P, T, F)$ has finite disjoint sets P of *places* and T of *transitions*, and the *flow* relation $F \subseteq (P \times T) \cup (T \times P)$.

We identify F with its characteristic function on the set $(P \times T) \cup (T \times P)$. We write $X = (P \cup T)$ for all nodes of a net. For a node $x \in X$, $\bullet x = \{y \in X \mid F(y, x) = 1\}$ and $x\bullet = \{y \in X \mid F(x, y) = 1\}$. A node $x \in X$ is an *input* (*output*) node of a node $y \in X$, iff $x \in \bullet y$ ($x \in y\bullet$). For $Y \subseteq X$, $\bullet Y = \bigcup_{y \in Y} \bullet y$ and $Y\bullet = \bigcup_{y \in Y} y\bullet$. For a node $x \in X$, $in(x) = \{(n, x) \mid n \in \bullet x\}$ are its *incoming* arcs and $out(x) = \{(x, n) \mid n \in x\bullet\}$ are its *outgoing* arcs. We denote by F^+ and F^* irreflexive and reflexive transitive closures of the flow relation, respectively.

Petri nets have precise execution semantics defined in terms of a token game.

Definition 2 (Net semantics)
Let $N = (P, T, F)$ be a net.

- $M : P \to \mathbb{N}_0$ is a *marking* of N, where $M(p)$, $p \in P$, returns the number of *tokens* in place p. $[p]$ denotes the marking when place p contains just one token and all other places contain no tokens. We identify M with the multiset containing $M(p)$ copies of p for every $p \in P$.
- For any transition $t \in T$ and for any marking M of N, t is *enabled* in M, denoted by $(N, M)[t\rangle$, iff $\forall\ p \in {\bullet}t : M(p) \geq 1$.
- If $t \in T$ is enabled in M, then it can *fire*, which leads to a new marking M', denoted by $(N, M)[t\rangle(N, M')$. The new marking M' is defined by $M'(p) = M(p) - F(p, t) + F(t, p)$, for each place $p \in P$.
- Let M_0 be a marking of N. If $(N, M_0)[t_1\rangle(N, M_1) \ldots (N, M_{n-1})[t_n\rangle(N, M_n)$ are transition firings, then a sequence of transitions $\sigma = t_1 \ldots t_n$, $n \in \mathbb{N}_0$, is a *firing sequence* leading from M_0 to M_n.
- For any two markings M and M' of N, M' is *reachable* from M in N, denoted by $M' \in [N, M\rangle$, iff there exists a firing sequence σ leading from M to M'. Note that σ can be the empty sequence. We have $M \in [N, M\rangle$ for every M of N.
- A *net system*, or a *system*, is a pair (N, M_0), where N is a net and M_0 is a marking of N. M_0 is called the *initial marking* of N.

Workflow (WF-)nets form a subclass of Petri nets. WF-nets were proposed in [15] for modeling workflow definitions. A WF-net is a net with two special places: one to mark the initialization and the other to mark the completion of a workflow.

Definition 3 (WF-net, Short-circuit net, WF-system)
A Petri net $N = (P, T, F)$ is a *workflow net*, or a *WF-net*, iff N has a dedicated *source* place $i \in P$, with ${\bullet}i = \emptyset$, N has a dedicated *sink* place $o \in P$, with $o{\bullet} = \emptyset$, and the *short-circuit* net $N' = (P, T \cup \{t^\star\}, F \cup \{(o, t^\star), (t^\star, i)\})$, $t^\star \notin T$, of N is strongly connected. A *WF-system* is a pair (N, M_i), where $M_i = [i]$.

In the following we shall also refer to the class of free-choice Petri nets [16]. A net $N = (P, T, F)$ is a *free-choice* net, iff $\forall p \in P$ with $|p\bullet| > 1$ holds ${\bullet}(p\bullet) = \{p\}$.

This article deals with structural characterization of behavioral correctness of WF-systems. Soundness and safeness are basic correctness properties of WF-systems [10]. Soundness states that every execution of a WF-system ends with a token in the sink place, and once a token reaches the sink place, no other tokens remain in the net. Safeness refers to the fact that there is never more than one token in the same place.

Definition 4 (Liveness, Boundedness, Safeness, Soundness)

- A system (N, M_0) is *live*, iff for every reachable marking $M \in [N, M_0\rangle$ and $t \in T$, there exists a marking $M' \in [N, M\rangle$, such that $(N, M')[t\rangle$.
- A system (N, M_0) is *bounded*, iff the set $[N, M_0\rangle$ is finite.
- A system (N, M_0) is *safe*, iff $\forall\ M \in [N, M_0\rangle\ \forall\ p \in P : M(p) \leq 1$.
- A WF-system (N, M_i) with $N = (P, T, F)$ is *sound*, iff the short-circuit system (N', M_i) is live and bounded.

Observe that in a sound WF-system, the initial marking $M_i = [i]$ always can evolve to the final marking $M_o = [o]$, at which no transition is enabled. If a WF-system is not sound, we also refer to it as an *unsound* system. Note that we refer to a WF-net N as being sound or unsound, if the WF-system (N, M_i) is sound or, respectively, unsound.

In our subsequent discussions we shall extensively refer to parts of nets. Therefore, we formally define the notions of a subnet and path of a net. Let $N' = (P', T', F')$ and $N = (P, T, F)$ be two nets, and let $P' \subseteq P$, $T' \subseteq T$. N' is a *subnet* of N, denoted by $N' \subseteq N$, iff $F' = F \cap ((P' \times T') \cup (T' \times P'))$. N' is a *partial subnet* of N, denoted by $N' \leq N$, iff $F' \subseteq F \cap ((P' \times T') \cup (T' \times P'))$. A *path* of N is a non-empty sequence $\langle x_1, \ldots, x_k \rangle$ of nodes, $x_i \in P \cup T$, $1 \leq i \leq k$ and $k > 1$, denoted by $\pi(x_1, x_k)$, which satisfies $(x_1, x_2), \ldots, (x_{k-1}, x_k) \in F$. We write $x_i \in \pi$, if x_i is on the path π. A *subpath* π' of a path π is a subsequence of π.

3 Stepwise Connectivity-Based Verification of WF-nets

In this section, we study the relation between the connectivity of short-circuit nets and the correctness, i.e., soundness, of WF-nets. The notion of a WF-net is in a tight relation with the class of strongly connected Petri nets. We emphasize this relation in Section 3.1. Afterwards, in Section 3.2, we present basic connectivity-related properties of nets. Then, in Sections 3.3 to 3.5, we investigate how the connectivity of a short-circuit net can be used for reasoning about the soundness of the WF-net. We perform the stepwise connectivity-based decomposition of a short-circuit net and derive conclusions about the soundness of the corresponding WF-net. At the same time, we argue about the computational complexity of algorithms that support the conclusions. For each decomposition step we discuss techniques to perform decompositions, methods to verify soundness, and feedback that can be provided to process analysts in cases of unsoundness.

3.1 Strong Connectivity of WF-nets

The structure of a Petri net (P, T, F) is defined by the graph (X, F). There is an interesting observation that relates the behavioral characteristics of a net with the connectivity of the underlying graph. The *strong connectedness theorem* [16] states that a net N for which there exists a marking M_0, such that (N, M_0) is live and bounded, is strongly connected. In other words, strong connectedness of a short-circuit net is a mandatory condition for soundness of the corresponding WF-net, as the latter is traced back to liveness and boundedness of the respective short-circuit net. A net is *strongly connected*, if there exist a directed path from each node in the net to each other node. Though implied by soundness, the requirement of strong connectedness of short-circuit nets is explicitly stated in the definition of WF-nets. Note that strong connectedness of a net can be tested in the time linear to the size of the net, viz., $O(|X| + |F|)$, by using Tarjan's algorithm for discovery of strongly connected components of a graph, cf., [17].

In the following, we examine connectivity-related properties of short-circuit nets, those not required by the definition, which can be used to explain soundness of WF-nets.

3.2 Connectivity of WF-nets

In this section, we briefly discuss connectivity-related properties of nets. We narrow down our discussion to those properties which we shall check later on short-circuit nets to derive conclusions about the soundness of the WF-nets. Detailed discussions and extensive examples will be provided as we proceed with the presentation of main results.

Two nodes x and y of a net are *connected*, if there is a path between x and y. Note that here we refer to a path that ignores directions of flow arcs. A net is *connected*, if every pair of distinct nodes of the net is connected. *k-connectivity* is the generalization of the connectivity property of a net. A net is *k-connected* if there exists no set of

$k-1$ nodes, whose removal renders the net disconnected, i.e., there is no path between some pair of nodes in the net. The set of nodes whose removal disconnects the net is called a *separating* $(k-1)$*-set* of the net. Separating 1- and 2-sets are called *cutvertices* and *separation pairs*, while 1-, 2-, and 3-connected nets are referred to as *connected*, *biconnected*, and *triconnected*, respectively. The *connectivity* of a net is the largest k for which the net is k-connected.

Fig. 1 shows a short-circuit net. Short-circuit nets are strongly connected and, thus, are connected. However, a short-circuit net must not be k-connected, where k is larger than one. For instance, the net in Fig. 1 contains cutvertex p_1 (highlighted with grey background). Thus, the net in the figure is not biconnected. Separation sets of a net can be used to decompose the net into subnets, or *components*, of a higher connectivity. The removal of the only cutvertex in Fig. 1 causes two subnets, one induced by nodes $\{p_1, t_2\}$ and the other by nodes $\{i, t_1, p_1, t_3, o, t^\star\}$. Both these subnets contain no cutvertices and, hence, are biconnected. In the subsequent sections, we study how separating sets and subnets induced by these sets can be used to decide on soundness of WF-nets.

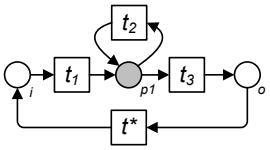


Fig. 1 A short-circuit net

3.3 The Biconnected Step

As already explained in the previous section, short-circuit nets are connected, but not necessarily biconnected. This section shows how the soundness verification of a WF-net can be broken down into checks of biconnected components of its short-circuit net.

3.3.1 Biconnected Decomposition of a WF-net

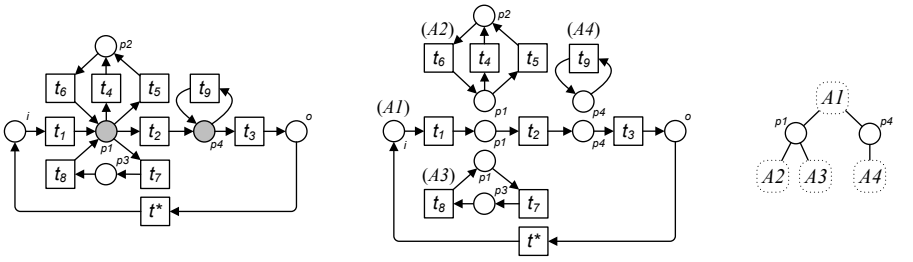
The classic sequential algorithm for computing biconnected components in a connected graph, proposed in [18], runs in linear time. Let (X, F) be a connected graph, then the algorithm requires time and space proportional to $max(|X|, |F|)$. Biconnected components can be arranged in a tree structure—the *tree of the biconnected components*. The tree has two types of nodes that refer either to cutvertices or to biconnected components. Edges of the tree connect cutvertices with associated biconnected components, i.e., there is an edge between a cutvertex and a biconnected component, if and only if the biconnected component contains the cutvertex. The number of nodes in the tree is $O(|X|)$ and, hence, the space required to store the tree and all the biconnected components is $O(max(|X|, |F|))$, i.e., linear to the size of the original graph. Such a tree is also known as BC-tree, cf., [19].

The results obtained for graphs can be transferred to WF-nets. A *biconnected subnet* is a biconnected component of a short-circuit net. The *tree of the biconnected subnets*, or the *2-WF-tree*, is the tree of the biconnected components of a short-circuit net.

Definition 5 (The tree of the biconnected subnets)
Let $N = (P, T, F)$ be a WF-net and let N' be its short-circuit net with the extra transition t^*. The *tree of the biconnected subnets*, or the *2-WF-tree*, of N is a tuple $\mathcal{T}_N^2 = (\mathcal{B}, \mathcal{C}, \rho, \eta, \triangle)$, where:

- $\mathcal{B}$ is a set of all biconnected subnets and $\mathcal{C}$ is a set of all cutvertices of N',
- $\rho = (P_\rho, T_\rho, F_\rho) \in \mathcal{B}$ is the root of $\mathcal{T}_N^2$, iff $t^* \in T_\rho$,
- $\eta : \mathcal{B} \to \mathcal{P}(\mathcal{B})$ assigns to each biconnected subnet its child biconnected subnets, for subnet $b_1 \in \mathcal{B}$, b_1 is a *parent* of $b_2 \in \mathcal{B}$ and b_2 is a *child* of b_1, iff $b_2 \in \eta(b_1)$,
- $\triangle \subseteq \mathcal{B} \times \mathcal{C} \times \mathcal{B}$, $(b_1, c, b_2) \in \triangle$, iff c is shared by b_1 and b_2, and $b_2 \in \eta(b_1)$.

Definition 5 captures the result of applying an algorithm for computing biconnected components on a short-circuit net. Note that we deliberately choose a biconnected subnet that contains the extra transition t^* as the root of the 2-WF-tree; there is always one such subnet. Function η must be defined iteratively starting from the root subnet, i.e., subnet ρ has no parent, each child subnet of the root subnet shares a cutvertex with it, each child subnet of the root is a parent of subnets that it shares a cutvertex with (except its parent), etc.



(a) (b) (c)

Fig. 2 (a) A short-circuit net, (b) biconnected subnets, and (c) the 2-WF-tree

Fig. 2 exemplifies the biconnected decomposition of a WF-net: Fig. 2(a) shows a short-circuit net. The net has two place cutvertices: $p1$ and $p4$; both are highlighted with grey background. The cutvertices induce four biconnected subnets $A1$–$A4$, cf., Fig. 2(b). Finally, Fig. 2(c) organizes the subnets in the 2-WF-tree with the root node that corresponds to the biconnected subnet $A1$.

Each biconnected subnet of a WF-net can be seen as a self-contained part of the whole net which itself can be formalized as a WF-net, referred to as a biconnected sub-WF-net of the original WF-net.

Definition 6 (Biconnected sub-WF-net)
Let $N = (P, T, F)$ be a WF-net, $\mathcal{T}_N^2 = (\mathcal{B}, \mathcal{C}, \rho, \eta, \triangle)$ its 2-WF-tree, and $b = (P_b, T_b, F_b) \in \mathcal{B}$ its biconnected subnet. A *biconnected sub-WF-net* of N, denoted by $b^\star$, $b^\star = (P_{b^\star}, T_{b^\star}, F_{b^\star})$, is a net, such that:

- If $b = \rho$, then $P_{b^\star} = P_b$, $T_{b^\star} = T_b \cap T$, and $F_{b^\star} = F_b \cap F$.
- If $b \neq \rho$ and $a \in \mathcal{B}$, $c \in \mathcal{C}$ are such that there exists $(a, c, b) \in \triangle$, then
  - if $c \in P$, then $P_{b^\star} = (P_b \setminus \{c\}) \cup \{i, o\}$, $T_{b^\star} = T_b$, and $F_{b^\star} = \{(x_1, x_2) \in F_b \mid x_1 \neq c \wedge x_2 \neq c\} \cup \{(i, x) \in \{i\} \times T_b \mid (c, x) \in F_b\} \cup \{(x, o) \in T_b \times \{o\} \mid (x, c) \in F_b\}$.
  - if $c \in T$, then $P_{b^\star} = P_b \cup \{i, o\}$, $T_{b^\star} = (T_b \setminus \{c\}) \cup \{t_i, t_o\}$, and $F_{b^\star} = \{(x_1, x_2) \in F_b \mid x_1 \neq c \wedge x_2 \neq c\} \cup \{(i, t_i), (t_o, o)\} \cup \{(t_i, x) \in \{t_i\} \times P_b \mid (c, x) \in F_b\} \cup \{(x, t_o) \in P_b \times \{t_o\} \mid (x, c) \in F_b\}$.

A WF-net that corresponds to a subtree of b, denoted by $b^\triangle$, can be obtained by merging (at shared cutvertices) sub-WF-net $b^\star$ with all subnets that are descendants of b in the 2-WF-tree. Biconnected sub-WF-nets are also referred to as *biconnected WF-nets*.

Fig. 3 presents sub-WF-nets of the short-circuit net that is given in Fig. 2(a). The sub-WF-nets correspond to the biconnected subnets in Fig. 2(b). Sub-WF-net $A1$ is obtained from

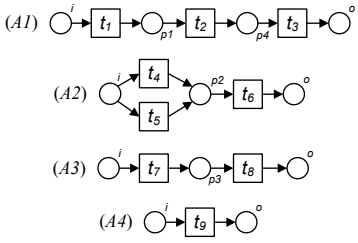


Fig. 3 Biconnected sub-WF-nets

the corresponding biconnected subnet by ignoring transition t^* and arcs that are incident to t^*. In the case when a biconnected subnet is not the root of the 2-WF-tree, cf., Fig. 2(c), the corresponding sub-WF-net can be obtained as follows: The cutvertex that corresponds to the parent node of the biconnected subnet in the 2-WF-tree is removed from the subnet and two fresh places are added; these are source place i and sink place o. If the removed cutvertex was a place, then all its outgoing arcs get rerouted to originate from i, whereas all its incoming arcs are rerouted to terminate at o. If the removed cutvertex was a transition, then two additional fresh transitions are added; these are transitions t_i and t_o. Afterwards, the flow relation is extended, such that t_i is put in the postset of i, while t_o is put in the preset of o. Finally, outgoing arcs of the removed cutvertex get rerouted to originate from t_i, whereas all the incoming arcs of the removed cutvertex are rerouted to terminate at t_o.

Construction of a WF-net that corresponds to a subtree in the 2-WF-tree is supported by two types of transformations, viz., refinements, of nets.

Definition 7 (Self-loop place refinement, Transition refinement)

- Let $N_1 = (P_1, T_1, F_1)$ be a net, $p \in P_1$ a place. A *self-loop place refinement* of p yields a net $N_2 = (P_1, T_1 \cup \{t_p\}, F_1 \cup \{(p, t_p), (t_p, p)\})$, denoted by $N_1[p]$.
- Let $N_1 = (P_1, T_1, F_1)$ be a net, $N_2 = (P_2, T_2, F_2)$ a WF-net with source i and sink o, $T_1 \cap T_2 = \emptyset$, $P_1 \cap P_2 = \emptyset$, and $t \in T_1$. A *transition refinement* of t by N_2 yields a net $N_3 = (P_3, T_3, F_3)$, denoted by $N_1[t/N_2]$, such that:
  - $P_3 = P_1 \cup (P_2 \setminus \{i, o\})$, $T_3 = (T_1 \setminus \{t\}) \cup T_2$, and
  - $F_3 = \{(x_1, x_2) \in F_1 | x_1 \neq t \wedge x_2 \neq t\} \cup \{(x_1, x_2) \in F_2 | \{x_1, x_2\} \cap \{i, o\}\} \cup \{(x_1, x_2) \in P_1 \times T_2 \mid (x_1, t) \in F_1 \wedge (i, x_2) \in F_2\} \cup \{(x_1, x_2) \in T_2 \times P_1 \mid (t, x_2) \in F_1 \wedge (x_1, o) \in F_2\}$.

A self-loop place refinement has been introduced in [20], while the concept of transition refinement is borrowed from [21,22]. Fig. 4(a) shows the result of the self-loop place refinement of place p_1 in WF-net $A1$ in Fig. 3, whereas Fig. 4(b) depicts the result of the transition refinement of t_{p_1} in the WF-net in Fig. 4(a) by WF-net $A2$ in Fig. 3.

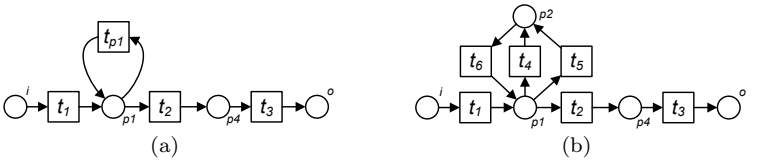


Fig. 4 (a) A self-loop place refinement of place p_1 in WF-net $A1$ in Fig. 3, i.e., $A1[p_1]$, and (b) a transition t_{p_1} refinement in (a) by WF-net $A2$ in Fig. 3, i.e., $(A1[p_1])[t_{p_1}/A2]$

3.3.2 Soundness Verification Based on Biconnected Decomposition

The biconnected decomposition of a WF-net can be related to the soundness verification in two ways: First, the type of the cutvertices can be exploited to detect unsoundness. Second, we show that the biconnected decomposition can be used to modularize the soundness verification problem. The following lemma shows that a WF-net can be sound only if all the cutvertices of the corresponding short-circuit net are places.

Lemma 1 *Let (N, M_i), $N = (P, T, F)$, be a WF-system and N' be the short-circuit net of N. If $t \in T$ is a cutvertex of N', then (N, M_i) is not sound.*

Proof Because t is a cutvertex of N', there exists $p' \in {}^\bullet t$, $p' \neq i$, such that t is on every path $\pi(i, p')$. We show now by induction that t is never enabled, i.e., for every marking $M \in [N, M_i\rangle$ holds $\neg(N, M)[t\rangle$.

base: $\neg(N, M_i)[t\rangle$ as $M_i(p') = 0$, i.e., t is not enabled by the initial marking.

step: Let M' be a marking reachable from M_i by a firing sequence σ that does not contain t, i.e., t was never enabled. Let $t' \in T$ be such that $(N, M')[t'\rangle$. Assume that $t' = t$, then $M'(p') \geq 1$. If $M'(p') \geq 1$, then σ contains all the transitions of some path $\pi(i, p')$ and, hence, contains t. We have reached the contradiction and, therefore, $t' \neq t$.

As t is never enabled, (N', M_i) is not live. Thus, (N, M_i) is not sound. □

According to Lemma 1, a transition cutvertex hints at unsoundness of the net. In case all cutvertices of a short-circuit net are places, the verification procedure can be broken down into checks of biconnected subnets of the short-circuit net. It is known that the self-loop place refinement preserves liveness, boundedness, and safeness of the net, cf., [20]. Therefore, soundness verification can be organized using the biconnected sub-WF-nets of a WF-net and the net transformations from Definition 7. That is, in the class of safe systems, the soundness of a system is closely related to the soundness of its biconnected sub-WF-nets.

Theorem 1 *Each biconnected sub-WF-net of a WF-net is safe and sound, iff the WF-net is safe and sound.*

Proof Let N be a WF-net and let $\mathcal{T}_N^2 = (\mathcal{B}, \mathcal{C}, \rho, \eta, \triangle)$ be the 2-WF-tree of N.

($\Rightarrow$) By structural induction on the tree of the biconnected subnets.

base: If $\mathcal{T}_N^2$ contains only one biconnected subnet, i.e., $|\mathcal{B}| = 1$, then N is a biconnected WF-net. Obviously, the claim holds.

step: Let $b \in \mathcal{B}$ be a biconnected subnet. Suppose that the claim holds for all $a^\triangle$ such that $a \in \eta(b)$. We show by induction that the claim is also true for $b^\triangle$.

$b^\star$ is a biconnected sub-WF-net of N and, hence, is safe and sound. Let $a \in \eta(b)$ and $c \in \mathcal{C}$ be such that $(b, c, a) \in \triangle$. A WF-net $b' = b^\star[c]$ with a self-loop transition t_c is safe and sound. A WF-net $b'[t_c/a^\triangle]$ is safe and sound, cf., statement 4 of Theorem 3 in [22]. Thus, after refining $b^\star$ with all the biconnected WF-nets that correspond to subnets from $\eta(b)$, one obtains a safe and sound WF-net that is equal to $b^\triangle$.

As $\rho^\triangle$ is equal to N, the claim holds.

($\Leftarrow$) The claim trivially holds by following ($\Rightarrow$) in the reverse direction. □

Therefore, it suffices to show that at least one biconnected sub-WF-net is not safe and sound in order to conclude that the WF-net is not safe and sound. As biconnected sub-WF-nets can be computed in time linear to the size of a net, the biconnected decomposition step does not add to the overall complexity of soundness verification.

3.3.3 Feedback on Unsoundness

We illustrate the feedback given by our verification approach by the exemplary model that is depicted in Fig. 5, along with its biconnected decomposition and the biconnected sub-WF-nets. Following on the results presented in the previous section, the first check to verify soundness refers to the type of the cutvertices. A transition cutvertex hints at unsoundness of the net and, therefore, constitutes valuable diagnostic information. For our example, we see that there is one transition cutvertex, viz., transition t_1. This transition is returned to a process analyst as a cause of unsoundness of the WF-net.

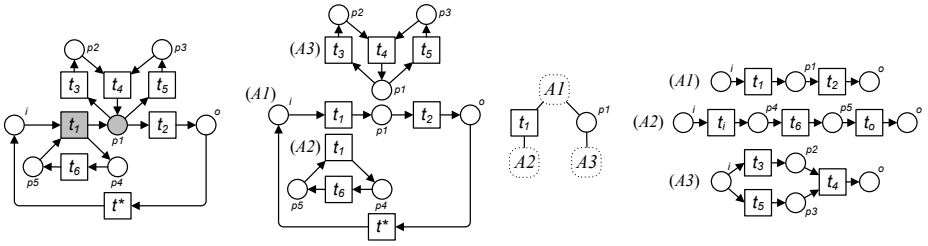


Fig. 5 (a) A short-circuit net of an unsound WF-net, (b) biconnected subnets, (c) the 2-WF-tree, (d) the biconnected sub-WF-nets

Apparently, this transition is never enabled, as the marking of one of the places in its preset depends on the firing of the very same transition. Hence, resolution of unsoundness has to consider the condition for enabling of transition t_1.

Second, we have shown that the soundness verification procedure can be broken down into checks of biconnected subnets of the short-circuit net. Once an unsound biconnected sub-WF-net is identified, it is presented to the process analyst. Again, this constitutes valuable diagnostic information. The cause of unsoundness is localized in a certain subnet and, thus, separated from the correct remaining parts of the WF-net. Referring to our example in Fig. 5, we see that the biconnected subnet ($A3$) induced by the cutvertex $p1$ is not sound. This subnet introduces an additional cause for unsoundness. The resolution of unsoundness also has to focus on this particular subnet.

3.4 The Triconnected Step

This section discusses connectivity specific aspects of the soundness verification of biconnected WF-nets. The biconnected WF-nets contain no cutvertices; they may, however, contain separation pairs that induce triconnected subnets.

3.4.1 Triconnected Decomposition of a Biconnected WF-net

The sequential algorithm for computing triconnected components in a biconnected graph proposed in [23] runs in linear time. In [24], the authors discuss implementation aspects of the algorithm. Let (X, F) be a biconnected graph, then the algorithm requires time and space proportional to $max(|X|, |F|)$. The triconnected components are used in [25] for analyzing the structure of directed graphs; the triconnected components form a hierarchy of single-entry-single-exit (SESE) subgraphs of a directed graph. In [26, 27], the authors propose a technique, viz., the Refined Process Structure Tree (RPST), that computes a hierarchy of SESE subgraphs in the general case when the graph contains vertices with multiple incoming and multiple outgoing edges. The subgraphs of the RPST are canonical, i.e., the RPST describes all the subgraphs that do not pairwise overlap on the sets of edges. In [28], the authors explain the relation between the triconnected components of the normalized version of the graph, where each vertex with multiple incoming and multiple outgoing edges is split into two, and its RPST. That is, the triconnected components of a normalized graph define its RPST.

In the following, we investigate the triconnected components of the normalized biconnected WF-nets. Every net can be normalized.

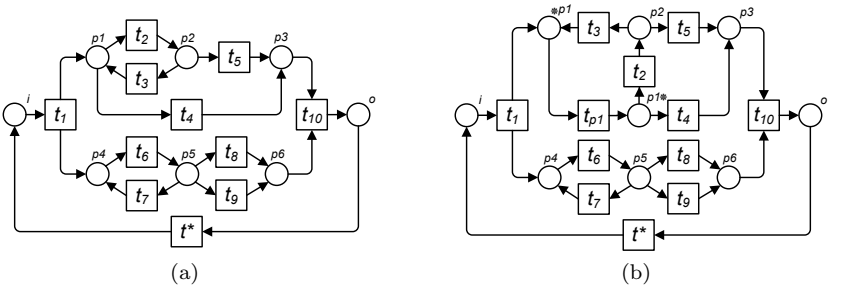


Fig. 6 (a) A short-circuit net, and (b) its normalized net

Definition 8 (Normalized net)
Let $N = (P, T, F)$ be a net.

- A *splitting* of $p \in P$ is *applicable*, iff $|\bullet p| > 1 \wedge |p \bullet| > 1$. The application results in a net $N' = (P', T', F')$, where $P' = ((P \setminus p) \cup \{*p, p*\}$, $T' = T \cup \{t_p\}$, and $F' = (F \setminus \{(x_1, x_2) \in F \mid x_1 = p \vee x_2 = p\}) \cup \{(t, *p) \in T \times \{*p\} \mid (t, p) \in F\} \cup \{(p*, t) \in \{p*\} \times T \mid (p, t) \in F\} \cup \{(*p, t_p), (t_p, p*)\})$.
- A *splitting* of $t \in T$ is *applicable*, iff $|\bullet t| > 1 \wedge |t \bullet| > 1$. The application results in a net $N' = (P', T', F')$, where $P' = P \cup \{p_t\}$, $T' = ((T \setminus t) \cup \{*t, t*\}$, and $F' = (F \setminus \{(x_1, x_2) \in F \mid x_1 = t \vee x_2 = t\}) \cup \{(p, *t) \in P \times \{*t\} \mid (p, t) \in F\} \cup \{(t*, p) \in \{t*\} \times P \mid (t, p) \in F\} \cup \{(*t, p_t), (p_t, t*)\})$.
- N is *normalized*, iff N has no applicable splitting.

Application of a splitting preserves liveness, safeness, and boundedness of a net, cf., [20]. The order of splittings has no effect on the final result. Fig. 6(a) shows the short-circuit net of a biconnected WF-net. The net has an applicable splitting of place p_1. Thus, it is not normalized. Fig. 6(b) shows the result of splitting p_1. As there are no further splittings applicable, the net in Fig. 6(b) is normalized.

Every separation pair of the triconnected component of the normalized short-circuit net induces a canonical triconnected subnet of the corresponding WF-net.

Definition 9 (Boundary, Entry, Exit, Triconnected subnet)
Let $N = (P, T, F)$ be a biconnected WF-net with source place i and sink place o. Let $\omega = (P_\omega, T_\omega, F_\omega)$ be a subnet of N, and let $x \in X_\omega$ be a node of ω.

- x is a *boundary* node of ω, iff $\exists\ f \in in(x) \cup out(x) : f \notin F_\omega$.
- x is an *entry* of ω, iff $in(x) \cap F_\omega = \emptyset$ or $out(x) \subseteq F_\omega$.
- x is an *exit* of ω, iff $out(x) \cap F_\omega = \emptyset$ or $in(x) \subseteq F_\omega$.
- ω is a *triconnected subnet* of N, iff ω has exactly two boundary nodes, one entry and one exit, denoted by $\omega_\triangleleft$ and $\omega_\triangleright$, respectively.
- ω is a *canonical* triconnected subnet in a set of all triconnected subnets Σ of N, iff $\forall\ \gamma = (P_\gamma, T_\gamma, F_\gamma) \in \Sigma : \omega \neq \gamma \Rightarrow (F_\omega \cap F_\gamma = \emptyset) \vee (F_\omega \subset F_\gamma) \vee (F_\gamma \subset F_\omega)$.

Note that we speak of *PP-*, *TT-*, *PT-*, *TP-bordered* triconnected subnets, depending on the type (place or transition) of the entry and exit of the respective subnet.

Fig. 7 shows two of many triconnected subnets of the net in Fig. 6(b). Subnet $B2$ is induced by separation pair $\{p_4, p_5\}$, whereas subnet $R1$ is induced by $\{*p_1, p_3\}$. p_4 is the entry and p_5 is the exit of $B2$. Similarly, $*p_1$ and p_3 are the entry and exit of $R1$.

There exist four structural classes of triconnected subnets that are adopted from classes of the triconnected components [25,26,27,28]: Each flow defines a subnet of a *trivial* class, e.g., (p_4, t_6) in $B2$. A subnet is a *polygon* if it decomposes into a sequence of subnets where the exit of a subnet is the entry of the next subnet in the sequence, e.g., two trivial subnets (p_4, t_6) and (t_6, p_5) form a polygon inside of $B2$. A subnet is a *bond* if it decomposes into a set of subnets that share boundary nodes, e.g., two polygons $\{(p_4, t_6), (t_6, p_5)\}$ and $\{(p_5, t_7), (t_7, p_4)\}$ share boundary nodes $\{p_4, p_5\}$ and form bond $B2$. If a subnet cannot be classified as trivial, polygon, or bond, it is said to be *rigid*, e.g., subnet $R1$. Note that names of subnets hint at their structural class.

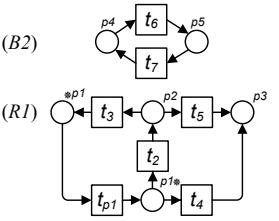


Fig. 7 Triconnected subnets

Definition 10 (The tree of the triconnected subnets)
Let $N = (P, T, F)$ be a normalized biconnected WF-net and let N' be its short-circuit net with the extra transition t^*. The *tree of the triconnected subnets*, or the *3-WF-tree*, of N is a tuple $\mathcal{T}_N^3 = (\Omega, \psi, \chi, \tau)$, where:

- Ω is a set of all canonical triconnected subnets of N',
- $\psi = (P_\psi, T_\psi, F_\psi) \in \Omega$ is the root of $\mathcal{T}_N^3$, iff $\nexists\, \omega = (P_\omega, T_\omega, F_\omega) \in \Omega : F_\psi \subset F_\omega$,
- $\chi : \Omega \to \mathcal{P}(\Omega)$ assigns to triconnected subnet its child triconnected subnets, for subnet $\omega_1 \in \Omega$, ω_1 is a *parent* of $\omega_2 \in \Omega$ and ω_2 is a *child* of ω_1, iff $\omega_2 \in \chi(\omega_1)$,
- $\tau : \Omega \to \{trivial, polygon, bond, rigid\}$ assigns a class to a subnet.

The tree of the triconnected subnets of a biconnected WF-net is defined by the RPST of its normalized short-circuit net and, thus, contains all canonical subnets of the net, cf., [28]. Trivial subnets are leaves in 3-WF-trees. Any polygon containing other polygons or bond containing other bonds are recognized as a single polygon or bond, respectively, cf., [25]. For our purposes, we further classify polygon and bond subnets: A subnet $\omega \in \Omega$ is a *simple* polygon, iff $\tau(\omega) = polygon$ and $\forall\, \alpha \in \chi(\omega) : \tau(\alpha) = trivial$. A subnet $\omega \in \Omega$ is a *sequence*, iff the subnet is a simple polygon or trivial. Let $h : \Omega \to \mathbb{N}_0$ assign heights to triconnected subnets, i.e., the length of the longest downward path from the corresponding node in the 3-WF-tree to a node that corresponds to a trivial subnet. A subnet $\omega \in \Omega$ is a *simple* bond, iff $\tau(\omega) = bond$ and $h(\omega) \leq 2$. Finally, a bond $\omega \in \Omega$ is a *loop*, iff $\exists\, \gamma \in \chi(\omega) : \gamma_\triangleright = \omega_\triangleleft$.

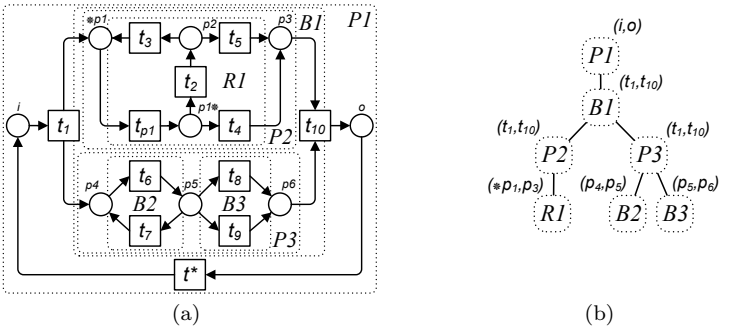


Fig. 8 (a) A normalized net and its triconnected subnets, and (b) the 3-WF-tree

Fig. 8 exemplifies the 3-WF-tree: Fig. 8(a) shows the net in Fig. 6(b) along with its canonical triconnected subnets. Each triconnected subnet is defined by the flow that is inside or intersects the corresponding box with the dotted line. Fig. 8(b) gives an alternative representation of the 3-WF-tree, i.e., a tree specified by the containment hierarchy of subnets. Each node in the tree is annotated with the pair of nodes of the net, where the first node is the entry and the second node is the exit of the corresponding subnet.

The 3-WF-tree specifies a compositional structure of the net. The net in Fig. 8(a) is composed of the top level polygon $P1$ with entry i and exit o. Polygon $P1$ contains bond $B1$ with entry t_1 and exit t_{10}. Observe that $P1$ also contains two trivial subnets: (i, t_1) and (t_{10}, o). Note that for simplicity reasons, we do not explicitly visualize trivial and sequence subnets. Bond $B1$ contains two polygons that share boundary nodes t_1 (an entry) and t_{10} (an exit) with the bond, where $P2$ corresponds to the upper branch and $P3$—to the lower one. Bonds $B2$ and $B3$ are simple bonds, both composed of two sequences. Moreover, bond $B2$ is also a loop. The net contains one rigid subnet—$R1$.

A triconnected subnet contains all the triconnected subnets, as well as the behavior that they specify, downwards in the hierarchy of the 3-WF-tree. However, the behavior induced by a triconnected subnet can be made explicit by abstracting the behavior of all its descendant subnets.

Definition 11 (Abstraction)
Let $N = (P, T, F)$ be a normalized biconnected WF-net. Let $\mathcal{T}_N^3 = (\Omega, \psi, \chi, \tau)$ be its 3-WF-tree, $\omega = (P_\omega, T_\omega, F_\omega) \in \Omega$, and $\gamma = (P_\gamma, T_\gamma, F_\gamma) \in \chi(\omega)$. An *abstraction* of γ in ω by transition t_γ results in a subnet $\omega_\gamma^\star$, such that:

- If $\gamma_\triangleleft \in P \wedge \gamma_\triangleright \in P$, then $\omega_\gamma^\star = (P_\omega \setminus (P_\gamma \setminus \{\gamma_\triangleleft, \gamma_\triangleright\}), (T_\omega \setminus T_\gamma) \cup \{t^\gamma\}, (F_\omega \setminus F_\gamma) \cup \{(\gamma_\triangleleft, t^\gamma), (t^\gamma, \gamma_\triangleright)\})$.
- If $\gamma_\triangleleft \in P \wedge \gamma_\triangleright \in T$, then $\omega_\gamma^\star = ((P_\omega \setminus (P_\gamma \setminus \{\gamma_\triangleleft\})) \cup \{p^\gamma\}, (T_\omega \setminus (T_\gamma \setminus \{\gamma_\triangleright\})) \cup \{t^\gamma\}, (F_\omega \setminus F_\gamma) \cup \{(\gamma_\triangleleft, t^\gamma), (t^\gamma, p^\gamma), (p^\gamma, \gamma_\triangleright)\})$.
- If $\gamma_\triangleleft \in T \wedge \gamma_\triangleright \in P$, then $\omega_\gamma^\star = ((P_\omega \setminus (P_\gamma \setminus \{\gamma_\triangleright\})) \cup \{p^\gamma\}, (T_\omega \setminus (T_\gamma \setminus \{\gamma_\triangleleft\})) \cup \{t^\gamma\}, (F_\omega \setminus F_\gamma) \cup \{(\gamma_\triangleleft, p^\gamma), (p^\gamma, t^\gamma), (t^\gamma, \gamma_\triangleright)\})$.
- If $\gamma_\triangleleft \in T \wedge \gamma_\triangleright \in T$, then $\omega_\gamma^\star = ((P_\omega \setminus P_\gamma) \cup \{p^{*\gamma}, p^{\gamma *}\}, (T_\omega \setminus (T_\gamma \setminus \{\gamma_\triangleleft, \gamma_\triangleright\})) \cup \{t^\gamma\}, (F_\omega \setminus F_\gamma) \cup \{(\gamma_\triangleleft, p^{*\gamma}), (p^{*\gamma}, t^\gamma), (t^\gamma, p^{\gamma *}), (p^{\gamma *}, \gamma_\triangleright)\})$.

An abstraction of a child subnet results in a net where the child subnet is replaced by a fresh transition. A *triconnected (sub-)WF-net* of N, denoted by $\omega^\downarrow$, can be obtained from ω by abstracting all its non-trivial child subnets, introducing a path from a fresh source place i leading to $\omega_\triangleleft$, possibly through a transition, and introducing a path from $\omega_\triangleright$ to a fresh sink place o, possibly through a transition.

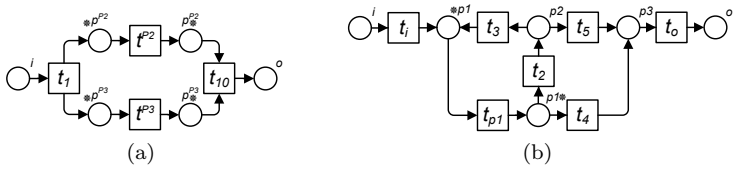


Fig. 9 Triconnected sub-WF-nets: (a) B1, and (b) R1

Fig. 9 shows two triconnected sub-WF-nets that correspond to subnets $B1$ and $R1$ of the net in Fig. 8(a). Note that the time to compute and space to store all the triconnected sub-WF-nets of a biconnected WF-net is linear to the size of the net, cf., [28].

3.4.2 Soundness Verification Based on Triconnected Decomposition

In the following, we discuss how the triconnected sub(-WF-)nets can be used to judge if a net behaves correctly. Similar to Section 3.3.2, we start by characterizing separating sets of a net. However, in this case the sets are separation pairs. First, we identify simple bonds that hint at a WF-net being not sound.

Lemma 2 (Pruning) *Let (N, M_i), $N = (P, T, F)$, be a biconnected WF-system. Let $\mathcal{T}_N^3 = (\Omega, \psi, \chi, \tau)$ be its 3-WF-tree, with a simple bond $\omega \in \Omega$. If ω is neither PP-, nor non-loop TT-, nor non-loop TP-bordered, then (N, M_i) is not sound.*

Proof Because ω is a simple bond, it holds that all child subnets of ω are sequences. There exist eight classes of simple bonds that are based on two criteria: is the bond a loop (ii) or not (i), and is the bond (a) PP-, (b) PT-, (c) TP-, or (d) TT-bordered. Fig. 10 visualizes all eight classes of simple bonds. In the figure, each directed arc stands for $n \in \mathbb{N}$ sequence subnets with entries and exits that correspond to the source and target of the arc. Note that we always assume the left node of a bond to be its entry node. Then, it trivially holds: If ω is of class (i.b), (ii.c), or (ii.d), then there is a dead transition in the WF-system. If ω is of class (ii.b), then there is a live-lock in the WF-system. In both cases, the WF-system is not sound. Moreover, if at least one child subnet of ω is trivial, then ω cannot be of class (i.a), (i.d), (ii.a), and (ii.d), due to the bipartite property of Petri nets. □

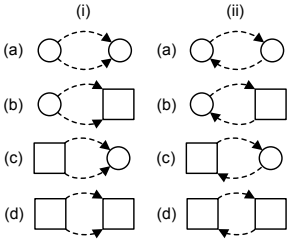


Fig. 10 Simple bond classes

A biconnected WF-system is said to be *pruned*, if it is not identified by Lemma 2 as unsound. Pruning is applicable for general biconnected WF-systems, but considers solely simple bonds. For the class of biconnected *free-choice* WF-nets, we can even provide a complete characterization of bonds. This is due to the fact that the free-choice property implies a tight coupling of the syntax and semantics for sound WF-systems, cf., [22, 29]. In order to present the results, we recall some definitions from [30].

Definition 12 (Circuit, Handle, Bridge)
Let $N = (P, T, F)$ be a net.

- A path $\pi(x_1, x_k)$ of N is a *circuit*, iff $(x_k, x_1) \in F$ and no node occurs more than once in the path.
- For a partial subnet $N' = (P', T', F')$ of N, a path $\pi(x_1, x_k)$ (where all x_i are distinct) of N is a *handle* of N', iff $\pi \cap (P' \cup T') = \{x_1, x_k\}$.
- For two partial subnets $N' = (P', T', F')$, $N'' = (P'', T'', F'')$ of N, a path $\pi(x_1, x_k)$ (where all x_i are distinct) of N is a *bridge* from N' to N'', iff $\pi \cap (P' \cup T') = \{x_1\}$ and $\pi \cap (P'' \cup T'') = \{x_k\}$.

Note that we speak of PP-, TT-, PT-, TP- handles and bridges, depending on the type (place or transition) of the start and the end node of the corresponding path. Finally, the following two lemmas show that a bond of a sound biconnected free-choice WF-system is either place or transition bordered, and that each loop is place bordered. While we introduced the two lemmas in previous work [14], we recall them for the sake of completeness of the presented approach.

Lemma 3 *Let (N, M_i), $N = (P, T, F)$, be a biconnected free-choice WF-system. Let $\mathcal{T}^3_N = (\Omega, \psi, \chi, \tau)$ be its 3-WF-tree, with a bond $\omega \in \Omega$. If ω is TP-bordered or PT-bordered, then (N, M_i) is not sound.*

Proof Assume ω is a bond with $\{p, t\}$ boundary nodes, $p \in P$ and $t \in T$. There exists a circuit Γ in N that contains $\{p, t\}$. Let Γ_ω be a subpath of Γ inside ω. There exists a child subnet γ of ω that contains Γ_ω. A bond has $k \geq 2$ child subnets, cf., [28,31]. Let ϑ be a child of ω, $\vartheta \neq \gamma$. We distinguish two cases:

- Let H be a path from p to t contained in ϑ. H is a PT-handle of Γ. In a live and bounded free-choice system, H is bridged to Γ_ω through a TP-bridge K, cf., Proposition 4.2 in [30]. This implies that $\vartheta = \gamma$; otherwise bond ω contains path K that is not inside of a single child subnet, cf., [31,28].
- Let H be a path from t to p contained in ϑ. H is a TP-handle of Γ. In a live and bounded free-choice system, no circuit has TP-handles, cf., Proposition 4.1 in [30]. Thus, (N, M_i) is not a sound free-choice WF-system.

Therefore, ω either has a single child subnet, in which case ω is not a bond, or (N, M_i) is not a sound free-choice WF-system. □

Lemma 4 *Let (N, M_i), $N = (P, T, F)$, be a biconnected free-choice WF-system. Let $\mathcal{T}^3_N = (\Omega, \psi, \chi, \tau)$ be its 3-WF-tree, with a loop $\omega \in \Omega$. If ω is TP-bordered, TT-bordered, or PT-bordered, then (N, M_i) is not sound.*

Proof Because of Lemma 3, ω is either place or transition bordered. Assume ω is transition bordered. There exists a place p such that $p \in {}^\bullet\omega_\triangleleft \cap P_\omega$, $M_i(p) = 0$. Transition $\omega_\triangleleft$ is enabled if there exists a marking $M \in [(N, M_i)\rangle$ with $M(p) > 0$. Since ω is a connected subnet, for all $t \in T_\omega \setminus \{\omega_\triangleleft, \omega_\triangleright\}$ all edges are in ω, i.e., $(in(t) \cup out(t)) \subseteq F_\omega$. Thus, every path from i to p visits $\omega_\triangleleft$. Thus, $M(p) > 0$ if $\omega_\triangleleft$ has fired, was enabled before. We reached a contradiction. Transition $\omega_\triangleleft$ is never enabled and N is not live, and hence, not sound. Since any loop is not transition bordered, it is place bordered (Lemma 3). □

Finally, similar as in Section 3.3.2, we propose to organize the soundness verification of biconnected WF-systems from individual checks of its sub-WF-nets, viz., triconnected sub-WF-nets.

Theorem 2 *Each triconnected sub-WF-net of a biconnected WF-net is safe and sound, iff the WF-net is safe and sound.*

Proof By structural induction on the tree of the triconnected subnets; analogous to the proof of Theorem 1. □

It suffices to show that at least one of the triconnected sub-WF-nets of a biconnected WF-net is not safe and sound in order to conclude that the net is not safe and sound. The analysis discussed in this section can be performed in the time linear to the size of a net and, hence, does not influence the overall complexity of soundness verification.

3.4.3 Feedback on Unsoundness

We illustrate the feedback given by our approach for the exemplary workflow net depicted together with its 3-WF-tree in Fig. 11. The workflow net is clearly not sound. A first cause of unsoundness can already be identified in the pruning step. Bond $B2$ induced by the separating pair (p_4, t_8) is a simple bond; all its child subnets are sequences. Furthermore, this simple bond is PT-bordered. This, in turn, contradicts with Lemma 2. Hence, the respective separating nodes, i.e., place p_4 and transition t_8, are returned to a process analyst as diagnostic information.

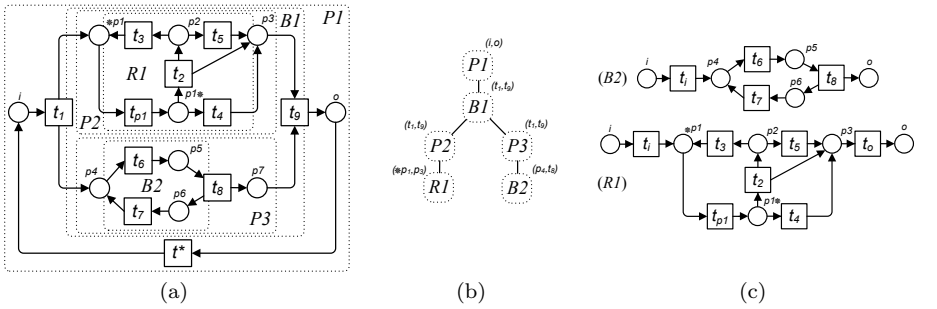


Fig. 11 (a) A short-circuit net of an unsound biconnected WF-net, (b) the 3-WF-tree, and (c) the triconnected sub-WF-nets

Further on, the example given in Fig. 11 also illustrates the kind of feedback given once an unsound triconnected subnet is identified. Here, the triconnected sub-WF-net $R1$ induced by the separation pair $(*p_1, p_3)$ is not sound, cf., Fig. 11(c), which according to Theorem 2 makes the whole net unsound. Observe that a net in Fig. 11(a) is free-choice and, thus, must also be safe. Therefore, the triconnected sub-WF-net $R1$ is an additional cause for unsoundness and has to be corrected in order to ensure soundness.

3.5 The 4-Connected Step

This section is devoted to soundness verification of rigid triconnected WF-nets. The short-circuit net of a rigid WF-net contains no separation pairs. Still, the net might contain separating sets composed of three nodes, which induce 4-connected components. Those might be leveraged for soundness verification.

3.5.1 4-Connected Decomposition of a Triconnected WF-net

This section explains an approach for the discovery of the 4-connected subnets in a triconnected WF-net. The authors are not aware of any existing dedicated algorithm that can be directly applied. However, the discovery can be organized by employing the technique from the triconnected step, cf., Section 3.4.1. The 4-connected subnets of a triconnected WF-net are detected as follows: First, a node is removed from the net. This step is referred to as *one-step connectivity reduction*. Afterwards, separation pairs of the modified net are discovered by constructing its 3-WF-tree. The removed node and each separation pair captured in the tree form a separating set composed of three nodes. The procedure should be repeated for each node in the net.

Definition 13 (One-step connectivity reduction)
Let $N = (P, T, F)$ be a rigid triconnected WF-net with source i and sink o.

- A node $y \in X$ is a *separation node*, iff $|\bullet y| + |y \bullet| > 2$.
- Given a separation node $y \in X$, a node $z \in X$ is in the set of *reduced nodes* $X^{(y)}$, iff $z = y$ or there is a path $\pi(y, z)$ or a path $\pi(z, y)$ that does not contain a separation node of N other than y.
- Given a separation node $y \in X$, the *one-step connectivity reduction* yields a subnet $N_y = (P_y, T_y, F_y)$, such that $P_y = P \setminus X^{(y)}$ and $T_y = T \setminus X^{(y)}$.
- A subnet N_y derived by one-step connectivity reduction of N by y is *well-structured*, iff its short-circuit net N'_y is strongly connected.

The one-step connectivity reduction is illustrated in Fig. 12. Fig. 12(a) depicts the short-circuit net of a rigid WF-net, while Fig. 12(b) shows the short-circuit of a subnet derived by one-step connectivity reduction by separation node p_2. As the resulting net is well-structured, its 3-WF-tree is computed, cf., Fig. 12(c).

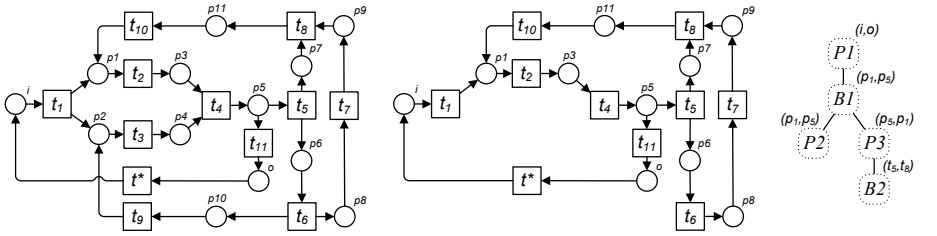


Fig. 12 (a) A short-circuit net of a rigid triconnected WF-net, (b) an exemplary one-step connectivity reduction of (a), and (c) the 3-WF-tree of (b)

The well-structured subnets derived via one-step connectivity reduction can be related to the structure of the original net.

Proposition 1 *Let N be a rigid WF-net, N' its short-circuit net, and N_x a well-structured subnet of N' derived via one-step connectivity reduction. Each bond subnet of N_x contains a handle for a circuit in N'.*

Proof Let $\omega = (P_\omega, T_\omega, F_\omega)$ be a bond subnet of N_x. As N_x is well-structured, its short-circuit net N'_x is strongly connected. Hence, N'_x has a circuit that, in turn, contains $\omega_\triangleleft$ and $\omega_\triangleright$. As ω is of type bond, it has at least two node-disjoint paths between $\omega_\triangleleft$ and $\omega_\triangleright$, where *node-disjoint* paths are paths with only end nodes in common. One of these paths is a subpath for the circuit in N'_x, whereas the second one is its handle. It trivially holds that the circuit is also the circuit in N'. □

3.5.2 Soundness Verification Based on 4-Connected Decomposition

For the class of free-choice nets, the 4-connected decomposition can be used to check mandatory conditions for the soundness of a rigid triconnected WF-net. Soundness of a free-choice WF-net can be decided using the *Rank Theorem* [16], which relates the well-formedness property (there exists a live and bounded marking for the net) to the incidence matrix of the net. According to [32], well-formedness and, thus, soundness of a

free-choice net can be decided in $O(|P|^2 \cdot |T|)$ time. Still, the 4-connected decomposition allows for more efficient checks of mandatory conditions for soundness. Consequently, these checks might be applied before soundness is assessed based on the Rank Theorem. The following lemma provides such a check by partial structural characterizations of sound nets.

Lemma 5 *Let (N, M_i) be a rigid free-choice WF-system and N_x a well-structured subnet of N derived via one-step connectivity reduction. If N_x contains a bond that is non-loop TP-bordered or loop PT-bordered, then (N, M_i) is not sound.*

Proof Let $\omega = (P_\omega, T_\omega, F_\omega)$ be a bond subnet of N_x. According to Proposition 1, ω induces a handle for a circuit in N. We distinguish two cases:

- ω is non-loop TP-bordered. Then, the handle induced by ω is a TP-handle.
- ω is loop PT-bordered. For $\omega_\triangleleft$ we know that either $in(\omega_\triangleleft) \cap F_\omega = \emptyset$ or $out(\omega_\triangleleft) \subseteq F_\omega$. The former is not possible as ω is a loop. From the latter, it follows that there is only one path $\pi(\omega_\triangleleft, \omega_\triangleright)$ in ω. Thus, the handle induced by ω for a circuit in N (Proposition 1) is a path $\pi(\omega_\triangleright, \omega_\triangleleft)$, i.e., a path from the exit to the entry of ω. As ω is PT-bordered, this path is a TP-handle.

In both cases, ω induces a TP-handle for a circuit in N. According to Proposition 4.1 in [30], no circuit of a live and bounded free-choice net has TP-handles. □

In the same vein, an acyclic rigid subnet has to meet a structural requirement in order to be sound. The following lemma applies to all free-choice biconnected WF-nets and not only to those derived via one-step connectivity reduction from a triconnected WF-net.

Lemma 6 *Let (N, M_i) be a biconnected free-choice WF-system. If N contains an acyclic rigid subnet that is TP-bordered, then (N, M_i) is not sound.*

Proof Let $\omega = (P_\omega, T_\omega, F_\omega)$ be a rigid subnet of N that is TP-bordered. Then, there is a circuit in the short-circuit net N' that contains both boundary nodes, $\omega_\triangleleft \in T$ and $\omega_\triangleright \in P$. According to Menger's theorem, cf., [33,34], there are two node-disjoint undirected paths from $\omega_\triangleleft$ to $\omega_\triangleright$. As ω is acyclic, all edges on these paths are directed equally, yielding two paths $\pi(\omega_\triangleleft, \omega_\triangleright)$. Thus, one of these paths is a TP-handle of the subnet containing the circuit. According to Proposition 4.1 in [30], no circuit of a live and bounded free-choice net has TP-handles. □

The application of the 4-connected decomposition allows for checking the mandatory conditions for soundness of a rigid free-choice WF-net in $O(|X|^2)$ time, where X are nodes of the net. For each separation node in a rigid free-choice WF-net, we apply a one-step connectivity reduction and derive the respective 4-connected components in linear time. For these components, Lemmas 5 and 6 can be checked in linear time as well. Consequently, this technique can be seen as a preliminary step that is applied before the more costly assessment of soundness based on the Rank Theorem.

3.5.3 Feedback on Unsoundness

We illustrate the feedback given in case unsoundness is detected by 4-connected decomposition with an example. Fig. 13 depicts the short-circuit net of a rigid triconnected WF-net similar to the one introduced above in Fig. 12. However, the net depicted in Fig. 13(a) is not sound. The cause for unsoundness is detected as follows: Applying the one-step connectivity reduction by separation node p_2 leads to the well-structured WF-net depicted in Fig. 13(b). For this subnet, in turn, the triconnected decomposition

is applied, which results in the 3-WF-tree illustrated in 13(c). The 3-WF-tree reveals that bond $B2$ is non-loop TP-bordered. According to Lemma 5, such a bond violates a mandatory condition for soundness of the WF-net. As diagnostic information, the respective handle induced by this bond is presented to the process analyst. That is, the two paths $\pi(t_5, p_7)$, one directly between the two nodes and one via the nodes p_6, t_6, p_8, and t_7, identify the cause of the unsoundness.

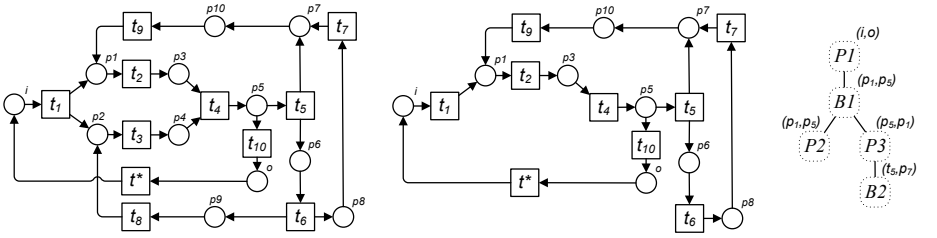


Fig. 13 (a) An unsound short-circuit net of a rigid triconnected WF-net, (b) an exemplary one-step connectivity reduction of (a), and (c) the 3-WF-tree of (b)

Feedback on unsoundness detected based on Lemma 6 can be provided in a similar way. The entry and exit of an acyclic TP-bordered rigid subnet are cause of unsoundness which must be corrected in order to ensure soundness. In fact, the WF-net in Fig. 13(a) contains a single TP-bordered rigid subnet with entry t_1 and exit p_5. However, the subnet is cyclic and, thus, in this case does not hint at unsoundness of the WF-net.

4 Application

This section shows how the results presented in Section 3 can be applied for the purpose of soundness verification. Algorithm 1 formalizes the verification procedure that is obtained by integrating the individual results proposed in Sections 3.3 to 3.5. The algorithm specifies a predicate which takes a WF-net as input and returns `true`, if the net is sound, and `false`, if the net is unsound. Note that Algorithm 1 does not formalize the way feedback is given in case of unsoundness; this was suppressed in order to keep the formalization concise. As discussed above, feedback can directly be given in terms of the elements that violate one of the necessary conditions for soundness.
First and foremost, Algorithm 1 comprises the biconnected decomposition of the given WF-net along with the verification of the types of the respective cutvertices. Afterwards, each biconnected sub-WF-net is decomposed into its triconnected subnets. For those, the algorithm checks the necessary conditions for soundness in the general case (types of boundary nodes of simple bonds). If these conditions are met, further necessary conditions are checked if the respective net is free-choice. This comprises checks for free-choice bonds, acyclic rigid subnets, and bonds derived from rigid subnets via one-step connectivity reduction. Note that if all these conditions are met but there are sub-WF-nets which cannot be handled by the proposed theory, we cannot conclude on soundness directly. As our structural approach is not exhaustive, we have to incorporate additional techniques which are capable of delivering results in the general case. For the class of free-choice nets we rely on the soundness verification based on the Rank Theorem, cf., Section 3.5.2, whereas in the general case we can get as exhaustive as performing reachability analysis of a net, cf., Algorithm 1. However, we see that for

Algorithm 1: Connectivity-based Soundness Verification of a WF-net

Input: N—a WF-net
Output: *true* if N is sound; *false* otherwise

$N' = (P', T', F')$:= get short-circuit net of N
$\mathcal{T}^2_{N'} = (\mathcal{B}, \mathcal{C}, \rho, \eta, \triangle)$:= get the tree of the biconnected subnets of N'
foreach $c \in \mathcal{C}$ **do if** $c \in T'$ **then return** *false*;
;
$areFC := areSound := true$
foreach $b \in \mathcal{B}$ **do**
  $b^\star$:= get normalized biconnected sub-WF-net of b
  $isFC :=$ *true* if $b^\star$ is free-choice; *false* otherwise
  $areFC := areFC \wedge isFC$
  $\mathcal{T}^3_{b^\star} = (\Omega, \psi, \chi, \tau)$:= get the tree of the triconnected subnets of $b^\star$
  foreach $\omega \in \Omega$ **do**
    if *(ω is simple bond) $\wedge$ (ω is PT-, loop TP-, or loop TT-bordered)* **then**
      return *false*
    if *areFC $\wedge$ (ω is bond) $\wedge$ (ω is PT-, TP-, or loop TP-bordered)* **then**
      return *false*
    if *areFC $\wedge$ (ω is rigid)* **then**
      if *(ω is acyclic) $\wedge$ (ω is TP-bordered)* **then return** *false*
      ;
      foreach ω' *derived from ω via one-step connectivity reduction* **do**
        if ω' *has bond that is non-loop TP-, or loop PT-bordered* **then**
        return *false*;
      end
      ω^λ := get triconnected sub-WF-net of ω
      $isSound :=$ *true* if ω^λ is sound by *Rank Theorem*; *false* otherwise
      $areSound := areSound \wedge isSound$
    end
  end
end
if $areFC \wedge areSound$ **then return** *true* ;
else perform reachability graph analysis;

certain classes of nets, i.e., nets that are free-choice and free of triconnected rigid subnets, our checks are even sufficient.

For validation purposes, we have implemented the verification approach that is proposed in Algorithm 1 and tested it against a collection of industry process models; we have used a model collection that was first used for the soundness verification in [11]. The model collection comprises 732 WF-nets, 375 of which are sound and 357 are unsound. The authors of [11] proposed soundness checking based on heuristics and state space exploration for the triconnected decomposition of (free-choice) process graphs and also report on findings regarding the application of other verification techniques, such as model checking and coverability analysis. While the authors of [11] show impressive results in terms of efficiency, the results have been achieved under optimizations for free-choice models. Here, our focus is different, as we aim at the verification of a general class of process models at the expense of completeness of the verification.

When testing the aforementioned collection of WF-nets for soundness, we observed the following results: 732 WF-nets contain 82 108 trivial, 26 818 polygon, 9 216 bond, and 553 rigid subnets. 256 WF-nets are classified as sound. 138 bond subnets are the cause of unsoundness (cf., Lemmas 2, 3, and 4). 56 rigid subnets are identified as the cause of unsoundness via one-step connectivity reduction (cf., Lemma 5), whereas 15 rigid subnets are classified as acyclic TP-bordered (cf., Lemma 6). Moreover, we have implemented heuristics for soundness verification that are discussed in [35]. This allowed us to additionally classify 29 rigid subnets as the cause of unsoundness and 78 rigid subnets as ones that can be present in sound WF-nets. These results show that a large share of models can be indeed verified by employing structural analysis only, among others the analysis proposed in this article, avoiding costly state space explorations. All the checks are performed within few milliseconds, which is comparable with the results reported in [11].

To illustrate the application of Algorithm 1, we refer to a fragment of a WF-net from the model collection; the fragment is shown in Fig. 14. The algorithm classifies the net as unsound. According to the theory of this article, there are several causes of unsoundness that can be discovered in the fragment. We highlight them in the figure. The net contains two TP-bordered bonds $B1$ and $B2$ (cf., Lemma 3), which are highlighted with black and, respectively, grey background in the top of the figure. Moreover, the net contains TP-bordered acyclic rigid $R1$ (cf., Lemma 6), see highlighted with black background in the middle of Fig. 14. Finally, rigid $R2$ is also the cause of unsoundness of the net, see highlighted with grey background in the lower part of the fragment. The one-step connectivity reduction of $R2$ reveals a TP-bordered bond (cf., Lemma 5); the entry and exit nodes of the bond are depicted with a thick borderline in the lower part of Fig. 14. Subnets, entries and exits of subnets, bonds derived via one-step connectivity reduction, these are examples of structural information which can be reported to a process analysts as a feedback on unsoundness of the WF-net.

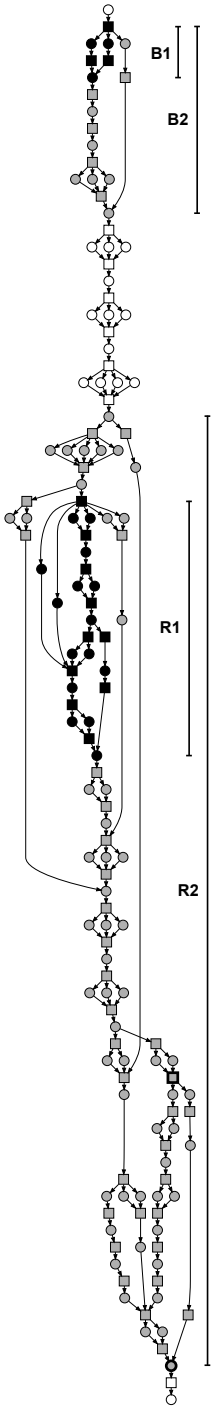


Fig. 14 A WF-net

5 Connectivity-Based Decomposition Framework

Based on the experience which was gained in Section 3, this section generalizes the ideas of connectivity-based decomposition of WF-nets into a comprehensive stepwise decomposition framework, i.e., a stepwise strategy for learning the compositional structure of behavioral models which use graphs as an underlying formalization mechanism. In Section 3, we focused on vertex connectivity of a concrete behavioral model, i.e., WF-nets. The framework presented in this section, comprises more generic notions of connectivity that are independent of a behavioral model. We believe that the proposed ideas can be of use in the

context of further developments in research on structural analysis of distributed systems.

First, in Section 5.1, we give the definition of a graph connectivity in the most general sense. Afterwards, principles of the connectivity-based decomposition of graphs are explained in Section 5.2. Finally, in Section 5.3, these principles are generalized and organized into the guidelines for decomposing behavioral models.

5.1 Connectivity of graphs

Connectivity is one of the basic concepts in graph theory. The investigations in Section 3 were performed using the connectivity property of workflow nets which is based on removals of places and/or transitions. In general, connectivity of a graph can also be checked based on removals of its edges. A graph G is a pair (V, E), where V is a set of vertices (or nodes) and $E \subseteq V \times V$ is a set of edges. A graph is undirected if the edges have no orientation, i.e., edges are treated as unordered pairs of vertices. A multi graph is a graph that allows multiple edges between a pair of vertices.

Connectivity is a property of a graph. In an undirected (multi) graph G, two vertices u and v are *connected* if G contains a path between u and v; otherwise the vertices are considered to be *disconnected*. A graph G is *connected* if every pair of distinct vertices in G is connected; otherwise G is *disconnected*. Please note that though connectivity is defined for undirected (multi) graphs it can be used in the context of directed graphs in a straight forward manner, i.e., by ignoring edge directions.

k-connectivity is the generalization of the connectivity property of a graph. A graph G is *k-connected* if there exists no set of $k-1$ elements, each a vertex *or* an edge, whose removal renders the graph disconnected, i.e., there is no path between some pair of elements in the graph. The set is called a *separating* $(k-1)$*-set* of G. Note that removal of a vertex implies removal of all its incident edges. Separating 1- and 2-sets of a graph that are composed solely of vertices are called *cutvertices* and *separation pairs*, while 1-, 2-, and 3-connected graphs are referred to as *connected*, *biconnected*, and *triconnected*, respectively. Finally, the *connectivity* of a graph is the largest k for which the graph is k-connected. Note that a graph composed of a single vertex is accepted to be connected, while a complete graph that is composed of n vertices, $n \geq 2$, is $(n-1)$-connected.

Fig. 15 shows two graphs. The graph in Fig. 15(a) is biconnected. Observe that removal of any element of the graph, either a vertex or an edge, keeps the graph connected. However, the graph in Fig. 15(a) is not triconnected. The removal of a separation pair $\{v_1, v_4\}$ renders the graph composed of two disconnected vertices v_2 and v_3. Hence, the largest k for which the graph is k-connected is 2. The graph in Fig. 15(a) can be modified to become "better" connected. The graph in Fig. 15(b) is obtained from the graph in Fig. 15(a) by adding a single edge that connects vertices v_2 and v_3. The modified graph is a complete graph composed of four vertices and, thus, it is triconnected. Note that removal of any pair of elements of the graph in Fig. 15(b) renders a connected graph.

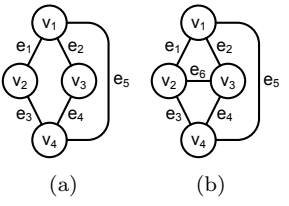


Fig. 15 (a) A biconnected graph, (b) a complete graph

5.2 Connectivity-Based Decomposition of Graphs

A k-connected graph contains no separating $(k-1)$-sets, but can contain separating k-sets. After removing a separating set from a graph, the graph gets decomposed

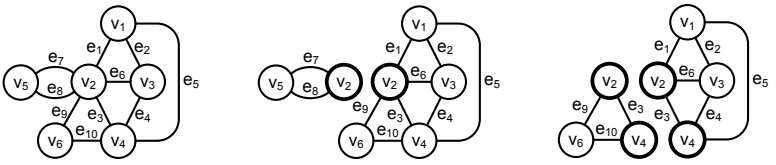


Fig. 16 (a) A graph, (b) the biconnected decomposition, and (c) the triconnected decomposition

into disconnected subgraphs (or *components*). Subsequently, obtained subgraphs of higher connectivity can be decomposed by using larger separating sets. By gradually increasing the size of separating sets used to decompose a graph, one performs a stepwise connectivity-based decomposition of the graph. In Fig. 16(b) and Fig. 16(c), we exemplify two steps of the connectivity-based decomposition of the graph in Fig. 16(a).

Decomposition starts with a connected graph. If a graph is disconnected, then it must be broken into connected subgraphs and the decomposition should proceed independently on each of the connected subgraphs. The graph in Fig. 16(a) is connected. In a connected graph, there exists a path between every pair of vertices. However, the existence of a path is not guaranteed if one element gets removed from the graph, either a vertex or an edge. Vertex v_2 is the only cutvertex of the graph in Fig. 16(a). If vertex v_2 gets removed, vertex v_5 gets disconnected from the rest of the graph. Therefore, the graph in Fig. 16(a) is not biconnected. A connected graph can be decomposed into biconnected subgraphs by means of the *biconnected decomposition*, which computes the biconnected subgraphs induced by the removals of cutvertices of the graph. For instance, Fig. 16(b) shows two subgraphs of the graph in Fig. 16(a) which are induced by the removal of its only cutvertex v_2.

Each of the subgraphs in Fig. 16(b) has no separating set that is composed of a single vertex or single edge and, hence, subgraphs are biconnected. One can proceed with the decomposition of these subgraphs into triconnected subgraphs. This can be accomplished by means of the *triconnected decomposition*, i.e., by removing separation pairs from the biconnected subgraphs. Because subgraph $(\{v_2, v_5\}, \{e_7, e_8\})$ in Fig. 16(b) is complete, the decomposition should proceed only on one subgraph. Finally, Fig. 16(c) shows two subgraphs induced by the removal of separation pair $\{v_2, v_4\}$. Both subgraphs in Fig. 16(c) are complete and, thus, decomposition terminates.

The connectivity-based decomposition, as exemplified above, gives information on the compositional structure of a graph, i.e., subgraphs that the graph is composed of and the principles of the subgraphs composition in the graph. In the example above, we have first decided to decompose the graph based on its cutvertices and afterwards decomposed induced subgraphs based on separation pairs. Note that usually the decision on how to decompose the given (sub)graph cannot be determined uniquely. For instance, decompositions of the subgraph in Fig. 16(b) can also be induced by removals of elements $\{e_9, e_{10}\}$ or $\{v_2, e_{10}\}$. Each sequence of decisions along decomposition of a graph results in a unique graph decomposition strategy which allows one to observe unique structural characteristics of the graph. In the next section, we propose a framework which organizes and suggests graph decomposition strategies.

5.3 Decomposition Framework

In this section, we describe a framework for performing structural decompositions of behavioral models that are formalized in the form of executable graphs. The framework

is founded on the principles of the graph decomposition which were discussed and exemplified in Section 5.2. We understand the framework as the research agenda which should provide the direction when searching for new results on the verification of workflow nets or when discovering new or improving existing techniques which are founded on structural decompositions of behavioral models.

The classical concept of graph connectivity is based on the notion of graph elements, both vertices and edges. In the following, we make a clear distinction! The *vertex (edge) connectivity* of a graph is the size of the smallest separating set of the graph that is composed only of vertices (edges). For an arbitrary graph, it always holds that its vertex connectivity is less than or equal to its edge connectivity [36]. Intuitively, the removal of an edge, when testing the edge connectivity, can be substituted by the removal of an incident vertex, which in turn implies the removal of all its incident edges. Finally, in the most general sense, one can speak about (n, e)-*connectivity* of a graph. A graph is (n, e)-*connected* if there exists no set of n nodes and there exists no set of e edges whose removal makes the graph disconnected.

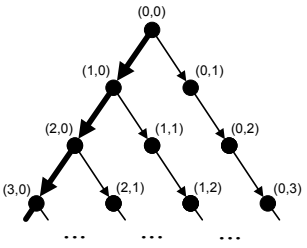


Fig. 17 The visualization of the connectivity-based decomposition framework

An (n, e)-connected graph, is not necessarily $(n + 1, e)$- or $(n, e + 1)$-connected. Therefore, it can be decomposed into the subgraphs of a higher connectivity. A graph decomposition strategy is a sequence of decomposition decisions, i.e., decisions on the decomposition of the subgraphs obtained on the previous decomposition step. Our decomposition framework organizes and suggests possible graph decomposition strategies based on the (n, e)-connectivity notion of graphs.

Fig. 17 visualizes the principles of the connectivity-based decomposition framework. In the figure, each point represents a connectivity property of a graph that is subject to decomposition. For example, point $(0, 0)$ means that the graph is connected if no nodes and no edges are removed. Arcs in the figure suggest which decompositions can be performed for a graph with a certain connectivity level. For instance, one can decompose a $(0, 0)$-connected graph by looking for a single node or a single edge whose removal renders the graph disconnected. Taking one of these decomposition steps will result in the discovery of either $(1, 0)$- or, respectively, $(0, 1)$-connected subgraphs of the original graph. Afterwards, $(1, 0)$- or $(0, 1)$-connected subgraphs can be treated as subjects for $(2, 0)$-, $(1, 1)$-, or $(0, 2)$-decomposition. By proceeding in this way, highly connected graphs get gradually decomposed.

Fig. 17 does not suggest all possibilities for performing decompositions. For instance, $(0, 2)$-connected graphs can be subjects for $(2, 0)$-decomposition. Within all possibilities for organizing a graph decomposition strategy, we propose to give the preference to the vertex-based decomposition over the edge-based one (refer to the left-most path in Fig. 17). Such a strategy stimulates the subgraphs to be discovered "faster", i.e., by performing less decomposition steps, and "finer", i.e., by discovering more subgraphs. Recall that the vertex connectivity of a graph is less than or equal to its edge connectivity. However, in the situations when it is absolutely important to achieve the granularity on the level of edges, we suggest to deviate from the main strategy only once by switching from the vertex-based to the edge-based decomposition strategy.

So far we have looked into decompositions of undirected graphs. Behavioral models, like WF-nets, are usually formalized as directed graphs, where directions of edges route

the flow of control through graphs. Edge directions are irrelevant for the purposes of the connectivity-based decomposition. However, edge directions become useful when classifying boundary nodes, i.e., nodes that connect subgraphs to the rest of the graph, of the derived subgraphs as entries or exits. A lot of research was carried out by the compiler theory community to gain value from the triconnected decomposition of process specifications, i.e., the discovery of the triconnected subgraphs in process graphs. The decompositions which proved useful are $(2,0)$-decomposition, or the tree of the triconnected components, cf., [25], and $(0,2)$-decomposition, cf., [37]. The triconnected subgraphs of a directed process graph form hierarchies of SESE fragments and are widely used for process analysis, process comparison, process comprehension, etc.

An (n,e)-decomposition, where $n+e \geq 3$, allows one to decompose triconnected graphs (or graphs with a higher connectivity level) into multi-entry-multi-exit (MEME) subgraphs, with $n+e$ entries and exits. For reasonable (n,e) combinations, i.e., when $n+e$ is sufficiently small, it is possible to perform decompositions in low polynomial-time complexity. For example, the $(3,0)$-decomposition of a $(2,0)$-connected graph can be accomplished by removing a vertex from the graph and afterwards running the triconnected decomposition, for which the linear time complexity algorithm exists, cf., [24]. Each discovered separation pair, together with the removed vertex, form a separating triple of the original graph. The procedure should be repeated for each vertex in the graph. Hence, a square-time complexity decomposition procedure can be obtained. Following the described rationale, one can accomplish $(k,0)$-decomposition of a graph in $O(n^{k-1})$ time, where n is proportional to the size of the graph.

6 Related Work

Our approach relates to other works on the verification of behavioral models. In [35], the authors propose to organize the verification of workflow graphs from fragments that have a single-entry edge and a single-exit edge, i.e., $(0,2)$-connected subgraphs in our classification. Albeit related, this work is based on the property of edge-connectivity, whereas our work leverages node-connectivity, yielding a more fine granular decomposition. Soundness checking based on heuristics and state space exploration for a triconnected decomposition, or $(2,0)$-connected subgraphs, of a (free-choice) process graph has been proposed in [11]. Our approach goes beyond this work by embedding the idea of the triconnected soundness checking into a decomposition approach that allows for stepwise verification. In addition, we base our findings on Petri nets as a generic behavioral model. Thus, our approach is able to cope with the whole spectrum of constructs of common modeling languages that can be mapped to Petri nets (e.g., exception handling in BPEL processes), whereas the existing approaches rely on the specific notion of a process graph. Note that [11] also reports findings on the application of other verification approaches, such as LoLA and Woflan. LoLA is a tool that is capable of checking various properties of a net by inspecting its state space [38]. To increase efficiency, LoLA incorporates several techniques for state space reduction. For the investigations in [11], the authors employed CTL model checking and partial order reduction of LoLA. Woflan is a tool for verifying the soundness of workflow nets [39]. Woflan combines structural Petri net reductions, S-coverability analysis, and state space exploration based on coverability trees into the unique verification approach. Our approach can be used in combination with these, or any other verification techniques, to deliver the *divide and conquer* strategy for verification of behavioral models.

Verification of Petri nets can benefit from structural reductions. Reduction rules are designed to transform a net into a smaller net while preserving essential properties of the net. Reduction rules can be applied before verification to decrease effects of state space explosion. Berthelot proposed a set of rules which reduce live and bounded marked graphs to a single transition [40,41]. Desel and Esparza, in [16], proposed a complete kit of reduction rules for free-choice Petri nets. In [20], Murata presented reduction rules which preserve the liveness, safeness, and boundedness properties of ordinary nets. Reduction rules constitute a line of active research not only for Petri nets, but also for models of concurrency which extend Petri nets. The extensions aim at support of features included in process definition languages such as BPMN, EPC, and UML activity diagrams. In [42], Wynn et al. proposed soundness-preserving reduction rules for reset WF-nets. Reset nets extend Petri nets with the concept of a reset flow. The semantics of a reset arc that connects a place and a transition is to remove all tokens from the place when the transition fires. Reset flow can be used to model cancellation. The proposed reduction rules are based on the rules for general nets and include additional restrictions with respect to reset arcs. By employing formal mappings of YAWL nets to reset nets, the same authors presented soundness-preserving reduction rules for YAWL workflow nets with cancellation regions and OR-joins [43]. Based on these rules, the authors devised an approach to check structural properties of YAWL nets, such as the weak soundness property, the soundness property, reducible cancellation regions, and convertible OR-joins. Finally, in [44], the authors defined a set of reduction rules which preserve the liveness and boundedness of reset/inhibitor nets. Reset/inhibitor nets are reset nets with the concept of an inhibitor flow. An inhibitor arc from a place to a transition can prevent the transition from being enabled if the place contains a token. Inhibitor arcs are useful when modeling blocking.

A convenient property of a set of reduction rules is that of completeness. Completeness guarantees that a net can always be reduced with the help of the reduction rules to another smaller net which hints at certain interesting property of the net, e.g., liveness, boundedness, or soundness. The completeness of a set of reduction rules is a well-known problem, e.g., all of the above mentioned sets of rules are incomplete when applied to nets of an arbitrary structure. Moreover given a set of rules, it is a challenge to decide in which order these rules must be applied to obtain the smallest reduced net, i.e., a net in which effects of the state space explosion problem, when checking a property of interest, are decreased the most. We advocate the usage of a conceptually different approach. Instead of contributing to the interminable search for a complete set of *specific* reduction rules, we operate with a set of *generic* rules which are defined using the connectivity property of graphs. We always operate with complete sets of rules (regardless of the net topology) which have common structural characteristics. Additionally, the connectivity property allows us to learn compositional structure of a net which suggests potential orders in which rules can be applied. Therefore, instead of searching for structural transformations which preserve an interesting property of a net, we check if this property can be deduced from a set of all subnets of the net; the subnets have common connectivity characteristics and collectively build up the net.

Further related work comprises inheritance preserving transformation rules for WF-systems defined by Wil van der Aalst and Twan Basten [45]. The rules are designed to avoid problems when migrating old workflow systems to new ones. When employed, the rules restrict changes in such a way that new workflow systems inherit certain properties of the old workflow systems. The rules can be applied in two directions, i.e., to reduce or to specify a workflow system. The transformations guarantee the

preservation of the soundness property. Albeit similar, these transformation rules are different from the approach proposed in this article. As our focus is solely on the verification of workflow systems, our structural implications on the soundness property are more general. For instance, our approach is not restricted by the class of safe workflow systems but is applicable for general workflow nets. Moreover, we additionally describe generic structural patterns which hint at unsoundness of workflow systems, e.g., a transition cutvertex, cf., Lemma 1. These patterns allow us to formulate feedbacks in cases of unsoundness. Finally, the particular feature of our approach is that it includes instructions for identification of *all* needful structural patterns within a workflow system, i.e., patterns which can confirm or reject soundness.

In [22], Wil van der Aalst exploited the hierarchical concept of transition refinement for checking soundness of workflow systems. Similarly, the approach reported in this article deals with the modular analysis of the soundness property. In particular, we have investigated an efficient way for modularization of workflow nets for the purpose of their further verification. During our investigations we reused some of the results on compositionality of workflow nets which were reported in [22].

7 Conclusion

In this work, we have investigated the relation between the connectivity property of a workflow net and its behavioral correctness in terms of the soundness property. We showed that soundness verification can be conducted following on a stepwise decomposition of a workflow net. For all the decomposition steps, we presented the necessary structural conditions for detecting unsoundness. We showed how our results are applied as a part of a verification procedure and tested these results against a collection of industry process models. During our tests, we observed the application of all formal results on the detection of unsoundness introduced in this article.

We introduced the connectivity-based decomposition framework – a systematic approach for learning the compositional structure of behavioral models. Our main contribution is the instantiation of this framework for soundness verification of workflow nets. However, we see a great potential for instantiating this framework for other modeling languages and use cases. Essentially, the connectivity-based decomposition framework defines a research agenda for improving methods which are founded on the structural decompositions of behavioral models.

Despite the large body of related work on the formal verification of process models, we are not aware of any work that employs the connectivity property as an angle to their structural analysis in a systematic way. As shown in this article, stepwise decomposition based on connectivity is an effective and efficient way to address verification. Even though recent work showed impressive results in terms of soundness checking efficiency [11], the results have been achieved under optimizations for free-choice models. Our approach of stepwise decomposition, in turn, provides the foundations to tackle general classes of behavioral models. By employing the decomposition, we realize a *divide and conquer* verification strategy that can be combined with existing verification techniques to achieve a higher level of maturity in solutions to the problem of behavioral verification. For instance, our decomposition strategy can be combined with the works on reduction and inheritance rules, cf., Section 6, to guide the application of these rules. While the decomposition is created using low-complexity algorithms, diagnostic information in case of unsoundness is provided as well.

The results reported in this article have already shown the usefulness of the connectivity property when checking soundness of WF-nets; the results range from the observation on cutvertices for the general class of nets to unique feedback on unsoundness for free-choice nets. We foresee the generalization of these results and introduction of new results for k-connected subnets as future steps in our research.

Acknowledgments. The authors want to thank two anonymous reviewers for their valuable comments that led to the major restructuring of content in the article.

References

1. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer-Verlag (2007)
2. Object Management Group: Business process model and notation (BPMN) version 2.0 beta 2 (June 2010)
3. Keller, G., Nüttgens, M., Scheer, A.W.: Semantische Prozeßmodellierung auf der Grundlage 'Ereignisgesteuerter Prozeßketten (EPK)'. Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Universität des Saarlandes (January 1992) In German.
4. Object Management Group: Unified modeling language: Superstructure (UML) version 2.1.1 (February 2007)
5. Reisig, W.: Petri Nets: An Introduction. Volume 4 of Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag (1985)
6. Lohmann, N., Verbeek, E., Dijkman, R.M.: Petri net transformations for business processes – a survey. Transactions on Petri Nets and Other Models of Concurrency (TOPNOC) **2** (2009) 46–63
7. Lohmann, N.: A feature-complete Petri net semantics for WS-BPEL 2.0. In: WS-FM. Volume 4937 of Lecture Notes in Computer Science., Springer (2008) 77–91
8. Kindler, E.: On the semantics of EPCs: A framework for resolving the vicious circle. In: BPM. Volume 3080 of Lecture Notes in Computer Science., Springer (2004) 82–97
9. Eshuis, R., Wieringa, R.: Tool support for verifying UML activity diagrams. IEEE Transactions on Software Engineering (TSE) **30**(7) (2004) 437–447
10. van der Aalst, W.M.P.: Verification of workflow nets. In: ICATPN. Volume 1248 of Lecture Notes in Computer Science., Springer (1997) 407–426
11. Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Instantaneous soundness checking of industrial business process models. In: BPM. Volume 5701 of Lecture Notes in Computer Science., Springer (2009) 278–293
12. Polyvyanyy, A.: Structural abstraction of process specifications. In: ZEUS. Volume 563 of CEUR Workshop Proceedings., CEUR-WS.org (2010) 73–79
13. Polyvyanyy, A., Weidlich, M., Weske, M.: The biconnected verification of workflow nets. In: OTM Conferences (1). Volume 6426 of Lecture Notes in Computer Science., Springer (2010) 410–418
14. Weidlich, M., Polyvyanyy, A., Mendling, J., Weske, M.: Efficient computation of causal behavioural profiles using structural decomposition. In: Petri Nets. Volume 6128 of Lecture Notes in Computer Science., Springer-Verlag (2010) 63–83
15. van der Aalst, W.M.P.: The application of Petri nets to workflow management. Journal of Circuits, Systems, and Computers (JCSC) **8**(1) (1998) 21–66
16. Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge University Press (1995)
17. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM Journal on Computing (SIAMCOMP) **1**(2) (1972) 146–160
18. Hopcroft, J., Tarjan, R.E.: Algorithm 447: Efficient algorithms for graph manipulation. Communications of the ACM (CACM) **16**(6) (1973) 372–378
19. Battista, G.D., Tamassia, R.: On-line graph algorithms with SPQR-trees. In: ICALP. Volume 443 of Lecture Notes in Computer Science., Springer (1990) 598–611
20. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE (PIEEE) **77**(4) (1989) 541–580
21. Valette, R.: Analysis of Petri nets by stepwise refinements. Journal of Computer and System Sciences (JCSS) **18**(1) (1979) 35–46
22. van der Aalst, W.M.P.: Workflow verification: Finding control-flow errors using Petri-net-based techniques. In: BPM. Volume 1806 of Lecture Notes in Computer Science., Springer (2000) 161–183

23. Hopcroft, J., Tarjan, R.E.: Dividing a graph into triconnected components. SIAM Journal on Computing (SIAMCOMP) **2**(3) (1973) 135–158
24. Gutwenger, C., Mutzel, P.: A linear time implementation of SPQR-trees. In: Graph Drawing. Volume 1984 of Lecture Notes in Computer Science., Springer (2001) 77–90
25. Tarjan, R.E., Valdes, J.: Prime subprogram parsing of a program. In: POPL. (1980) 95–105
26. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. In: BPM. Volume 5240 of Lecture Notes in Computer Science., Springer (2008) 100–115
27. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. Data and Knowledge Engineering (DKE) **68**(9) (2009) 793–818
28. Polyvyanyy, A., Vanhatalo, J., Völzer, H.: Simplified computation and generalization of the refined process structure tree. In: WS-FM. Volume 6551 of Lecture Notes in Computer Science., Springer (2011) 25–41
29. Kiepuszewski, B., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Fundamentals of control flow in workflows. Acta Informatica **39**(3) (2003) 143–209
30. Esparza, J., Silva, M.: Circuits, handles, bridges and nets. In: ATPN. Volume 483 of Lecture Notes in Computer Science., Springer (1991) 210–242
31. Polyvyanyy, A., Smirnov, S., Weske, M.: The triconnected abstraction of process models. In: BPM. Volume 5701 of Lecture Notes in Computer Science., Springer (2009) 229–244
32. Kemper, P.: Linear time algorithm to find a minimal deadlock in a strongly connected free-choice net. In: ATPN. Volume 691 of Lecture Notes in Computer Science., Springer (1993) 319–338
33. Menger, K.: Zur allgemeinen Kurventheorie. Fund. Math. **10** (1927) 96—115
34. Skiena, S.S.: Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica. Reading, MA: Addison-Wesley (1990)
35. Vanhatalo, J., Völzer, H., Leymann, F.: Faster and more focused control-flow analysis for business process models through SESE decomposition. In: ICSOC. Volume 4749 of Lecture Notes in Computer Science., Springer (2007) 43–55
36. Diestel, R.: Graph Theory. Springer-Verlag (2005)
37. Johnson, R.C.: Efficient Program Analysis Using Dependence Flow Graphs. PhD thesis, Cornell University, Ithaca, NY, USA (1995)
38. Wolf, K.: Generating Petri net state spaces. In: ICATPN. Volume 4546 of Lecture Notes in Computer Science., Springer (2007) 29–42
39. Verbeek, E., Basten, T., van der Aalst, W.M.P.: Diagnosing workflow processes using woflan. The Computer Journal (CJ) **44**(4) (2001) 246–279
40. Berthelot, G.: Checking properties of nets using transformation. In: ATPN. Volume 222 of Lecture Notes in Computer Science., Springer (1986) 19–40
41. Berthelot, G.: Transformations and decompositions of nets. In: Advances in Petri Nets. Volume 254 of Lecture Notes in Computer Science., Springer (1987) 359–376
42. Wynn, M.T., Verbeek, E., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Soundness-preserving reduction rules for reset workflow nets. Information Sciences (ISCI) **179**(6) (2009) 769–790
43. Wynn, M.T., Verbeek, E., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Reduction rules for YAWL workflows with cancellation regions and OR-joins. Information & Software Technology (INFSOF) **51**(6) (2009) 1010–1020
44. Verbeek, E., Wynn, M.T., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Reduction rules for reset/inhibitor nets. Journal of Computer and System Sciences (JCSS) **76**(2) (2010) 125–143
45. van der Aalst, W.M.P., Basten, T.: Inheritance of workflows: An approach to tackling problems related to change. Theoretical Computer Science (TCS) **270**(1–2) (2002) 125–203