DOI 10.1007/s00165-010-0161-4
The Author(s) © 2010. This article is published with open access at Springerlink.com
Formal Aspects of Computing (2011) 23: 333–363

Formal Aspects of Computing

Soundness of workflow nets: classification, decidability, and analysis

W. M. P. van der Aalst[1,2], K. M. van Hee[1], A. H. M. ter Hofstede[1,2], N. Sidorova[1], H. M. W. Verbeek[1], M. Voorhoeve[1] and M. T. Wynn[2]

[1] Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
E-mail: w.m.p.v.d.aalst@tue.nl
[2] Queensland University of Technology, Brisbane, Australia

Abstract. *Workflow nets*, a particular class of Petri nets, have become one of the standard ways to model and analyze workflows. Typically, they are used as an abstraction of the workflow that is used to check the so-called *soundness property*. This property guarantees the absence of livelocks, deadlocks, and other anomalies that can be detected without domain knowledge. Several authors have proposed alternative notions of soundness and have suggested to use more expressive languages, e.g., models with cancellations or priorities. This paper provides an *overview of the different notions of soundness and investigates these in the presence of different extensions of workflow nets*. We will show that the eight soundness notions described in the literature are decidable for workflow nets. However, most extensions will make all of these notions undecidable. These new results show the theoretical limits of workflow verification. Moreover, we discuss some of the analysis approaches described in the literature.

Keywords: Petri nets, Decidability, Workflow nets, Reset nets, Soundness, Verification

1. Introduction

Before we outline the content of the paper, we first motivate the need for workflow verification techniques. Then we justify the setting chosen, followed by a discussion on the tradeoff between expressiveness and decidability.

1.1. Motivation

In the last 15 years, we have witnessed a shift from "data-aware" information systems to "process-aware" information systems [DAH05]. To support business processes an enterprise information system needs to be aware of these processes and their organizational context. Early examples of process-aware information systems were called WorkFlow Management (WFM) systems [AH04, GHS95, JB96, LR99, Mue04, Wes07]. In more recent years, vendors prefer the term Business Process Management (BPM) systems. BPM systems have a wider scope than the classical WFM systems and are not just focusing on process automation. BPM systems tend to provide more support for various forms of analysis and management support. Both WFM and BPM aim to support operational processes that we refer to as "workflow processes" or simply "workflows".

Correspondence and offprint requests to: W. M. P. van der Aalst, E-mail: w.m.p.v.d.aalst@tue.nl

WFM and BPM systems are driven by process models. Therefore, it is important to ensure the correctness of such models. Unfortunately, existing commercial systems do not support the verification of workflows. Moreover, as shown in various case studies, process designers tend to make many errors. Consider for example the study reported in [MNA07] which is based on more than 2000 process models and showed that more than 10 percent of these models is flawed. Even established sets of models such as the SAP reference model (consisting of 604 EPC models) turn out to have many errors (more than 20 percent is flawed). Typical errors are deadlocks (a case gets stuck), livelocks (a case cannot progress), and other anomalies. Repairing such errors can be time consuming and costly. Therefore, workflow verification is highly relevant. However, as workflow languages get more expressive, it becomes more difficult to analyze them. This paper explores the theoretical limits of workflow verification.

1.2. Soundness of WF-nets

The flow-oriented nature of workflow processes and mainstream workflow notations makes the Petri net formalism a natural candidate for the modeling and analysis of workflows. Most workflow management systems provide a graphical language which is close to Petri nets. Although the routing elements are different from Petri nets, the informal semantics of the languages used are typically token-based and hence a (partial) mapping is relatively straightforward. A characteristic of workflow processes is that they are typically case-oriented, i.e., processes can be instantiated for multiple cases but the life-cycles of different cases do not get intertwined.

As an example, let us consider Fig. 1a which models a workflow in terms of BPMN (Business Process Modeling Notation) [Whi09]. BPMN is becoming the de facto standard for business process modeling. However, in the past other notations (EPC, BPEL, etc.) have been equally successful and it is to be expected that in the future other notations will come and go. Moreover, languages such as BPMN have no formal semantics. Fortunately, it is easy to translate the basic constructs of such languages (XOR/AND-splits/joins) onto Petri nets given their flow-oriented nature and intended token semantics. Figure 1b illustrates this. Both models describe a complaints handling workflow that starts with a registration step after which two parallel branches are started. The top branch is concerned with the handling of a questionnaire. After sending the questionnaire to the customer that submitted the complaints, there are two possibilities. The customer may return the questionnaire in time and subsequently it is processed. Otherwise, a timeout occurs and this step is skipped. In the lower branch of the workflow, the complaint is first processed. After this the result is evaluated. Based on this evaluation, the complaint is checked or not. If it is checked, the result may be OK or not. If it is not OK, the complaint is processed again. This is repeated until no check is needed or the check is OK. Finally, after completing both parallel branches, the complaint is archived. Note that the BPMN notation has special symbols for XOR/AND-splits/joins. These are called gateways. For example, the "+" gateway following the registration step is a so-called AND-split. Both parallel branches are synchronized by the "+" gateway just before archive. Similarly, "×" gateways model XOR-splits and XOR-joins. For example, the evaluate step is followed by an "×" gateway modeling the corresponding choice.

The corresponding Petri net shown in Fig. 1b has a particular structure starting with a source place *start* and ending with a sink place *end*. The modeled workflow can be instantiated by putting tokens on the input place *start*. Each of these tokens represents the creation of a particular case. The goal is that after a while there will be a token with a sink place in output place *end* for each initiated case.

This paper focuses on processes having the structure shown in Fig. 1b. These are so-called *workflow nets* (WF-nets). WF-nets were introduced in [Aal97, Aal98] and are currently the most widely used model to *formally* describe workflow processes.

In the context of WF-nets a correctness criterion called *soundness* has been defined [Aal97, Aal98]. A WF-net such as the one sketched in Fig. 1b is sound if and only if the following three requirements are satisfied: (1) *option to complete*: for each case it is always still possible to reach the state which just marks place *end*, (2) *proper completion*: if place *end* is marked all other places are empty for a given case, and (3) *no dead transitions*: it should be possible to execute an arbitrary activity by following the appropriate route through the WF-net. In [Aal97, Aal98] it was shown that soundness is decidable and that it can be translated into a liveness and boundedness problem, i.e., a WF-net is sound if and only if the corresponding short-circuited net (i.e., the net where place *end* is connected to place *start*) is live and bounded.

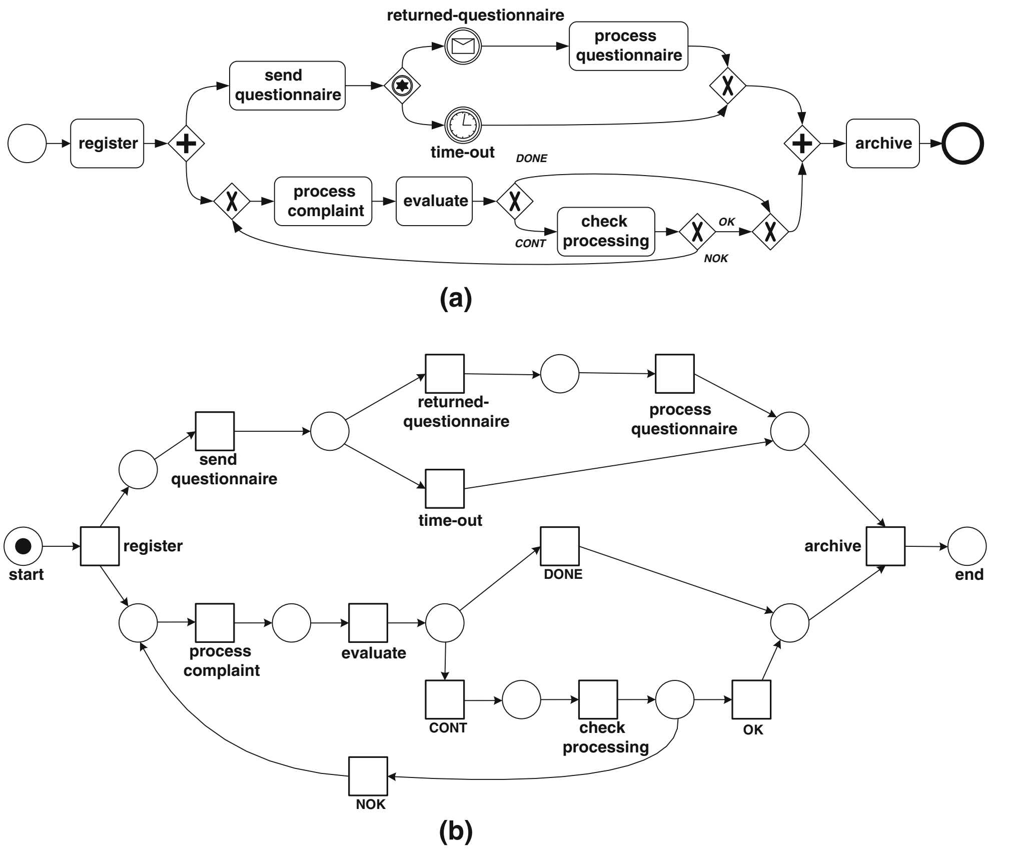

Fig. 1. A BPMN model and its corresponding WF-net. A WF-net is a Petri net with a start and an end place. The goal is that a case initiated via place *start* successfully completes by putting a token in place *end*. **a** Example process expressed in terms of BPMN. **b** Same process mapped onto a WF-net

1.3. On the tradeoff between expressiveness and decidability

Since the mid-nineties many people have been looking at the verification of workflows. These papers all assume some underlying model (e.g., WF-nets) and some correctness criterion (e.g., soundness). Hence there are two dimensions when considering workflow verification:

- *Expressiveness of the model*. Some authors assume a model that is less expressive than WF-nets, e.g., there are many variants of the so-called workflow graph model [SO00] which is essentially a free-choice net and thus easier to analyze than WF-nets without restrictions. Other authors propose more expressive models, e.g., models that allow for cancellation, priorities, data dependencies, recursion, and complex joins such as the inclusive OR-join [Kin06, WAHE06].
- *Correctness criterion*. The notion of soundness defined in [Aal97, Aal98] is intuitively appealing. However, some authors suggest weakening the correctness notion [DA04, DR01, Mar03, Mar05a, PW06b] while others propose to strengthen the correctness notion [HSV04, HSSV05, Too04].

In this paper, we are interested in the verification of different variants of WF-nets using different soundness notions. We will systematically consider *four classes of WF-nets and eight notions of soundness* and focus on the decidability of the corresponding 4*8=32 verification problems.

The four classes of WF-nets are based on two possible extensions of WF-nets: *reset arcs* and *inhibitor arcs*. We will show that other extensions do not influence the expressive power of WF-nets and therefore are less relevant. Reset arcs are closely linked to notions of cancellation present in some of the more advanced languages. Inhibitor arcs allow for the modeling of advanced constructs such as priorities, preemption, OR-joins, etc.

The eight notions of soundness used in this paper have been identified after a thorough analysis of the literature. Some of the notions weaken one or more of the requirements in the original definition [Aal97, Aal98], e.g., the requirement that the net should not have dead transitions or the requirement that the net should always terminate properly. Other notions strengthen some of the conditions stated in [Aal97, Aal98], e.g., the requirement that soundness still holds even if the net is instantiated multiple times in parallel.

Thus far the decidability of soundness has not been investigated systematically. In fact, as far as we know, *this is the first paper to investigate decidability of soundness for WF-nets with reset and/or inhibitor arcs*.

The main motivation for this research is that researchers continue to come up with new workflow models and verification techniques. By providing a systematic overview, we hope to reveal the fundamental limits of workflow verification. This way we hope to avoid that authors continue to come up with verification problems and approaches that turn out to be special cases of already known results, i.e., we want to help the researchers with positioning their research problems as special cases of already known results, thus avoiding the re-invention of the wheel. Moreover, we show that most correctness notions are in fact undecidable when combined with certain extensions that correspond to reset or inhibitor arcs. This clearly shows the theoretical limits of workflow verification.

1.4. Outline

The remainder of this paper is organized as follows. First, we briefly present an overview of related work (Sect. 2). A more detailed review of related work is given in later parts of the paper, e.g., when introducing the various soundness notions. Then, Sect. 3 presents some of the preliminaries (mathematical notations and Petri net basics). Section 4 presents the basic notion of a WF-net and introduces the four classes of WF-nets investigated in this paper. In Sect. 5 the classical notion of soundness is introduced followed by definitions of seven other notions of soundness considered in the literature. Section 6 presents the main results. It systematically investigates the four classes of WF-nets and eight notions of soundness and focuses on the decidability of the corresponding 32 verification questions. Section 7 provides pointers to different analysis approaches. The goal is not to present new methods but to provide a high-level overview of existing approaches. Here we also emphasize that despite the fact that many verification questions are undecidable, a more pragmatic approach can help in finding numerous errors. Section 8 concludes the paper by summarizing the results and reflecting on the state-of-the-art in workflow verification.

2. Related work

Since the mid nineties, many researchers have been working on workflow verification techniques [Aal97, Aal98, Aal00, AHV02, ALM+08, BP98, BB00, BK02, BZ04, DR01, DAV05, FBS02, FBS04, HSV04, HSS05, KGMW00, KMR00, LZLC02, LMSW06, Mar05a, Mar05b, MRS05a, MRS05b, MMN+06, MNA07, SO97, SO00, SW01, VVL07, VA05, VAH07, VBA01, Wom06, WAHE06, WEAH05]. It is impossible to give a complete overview here. Moreover, most of the papers on workflow verification focus on rather simple languages, e.g., AND-XOR-graphs which are even less expressive than classical Petri nets. Therefore, we only mention the work directly relevant to this paper.

This paper extends the results given in [AHH+09] where it is shown that soundness is undecidable for WF-nets with reset arcs. In this paper, these results are extended in two directions: (1) various notions of soundness are investigated (not only classical soundness); (2) also WF-nets with inhibitor arcs are considered. Moreover, unlike [AHH+09], this paper aims to provide a survey of workflow correctness notions and discuss potential analysis techniques.

The use of Petri nets in workflow verification has been studied extensively. In [Aal97, Aal98] the foundational notions of WF-nets and soundness are introduced. In [HSV03, HSV04] two alternative notions of soundness are introduced: k-soundness and generalized soundness. These notions allow for dead parts in the workflow but address problems related to multiple instantiation. In [Mar03, Mar05a] the notion of weak soundness is proposed. This notion allows for dead transitions. The notion of relaxed soundness is introduced in [DA04, DR01]. This notion allows for potential deadlocks and livelocks, however, for each transition there should be at least one proper execution. Lazy soundness [PW06b, PW06a] is another variant that only focuses on the end place and allows for excess tokens in the rest of the net. Finally, the notions of up-to-k-soundness and easy soundness are introduced in [Too04]. More details on these notions proposed in the literature are given in Sect. 5.

Most soundness notions (except generalized soundness [HSV03, HSV04]) can be investigated using classical model checking techniques that explore the state space. However, such approaches can be intractable or even impossible because the state-space may be infinite. Therefore, alternative approaches that avoid constructing the (full) state space have been proposed. [Aal00] describes how structural properties of a workflow net can be used to detect the soundness property. [VA05, VAH07] presents an alternative approach for deciding relaxed soundness in the presence of OR-joins using invariants. The approach taken results in the approximation of OR-join semantics and transformation of YAWL nets [AH05] into Petri nets with inhibitor arcs. In the general area of reset nets, Dufourd et al.'s work has provided valuable insights into the decidability status of various properties of reset nets including reachability, boundedness and coverability [DFS98, DJS99, FS01]. For decidability results for ordinary Petri nets we refer to [CW99, Esp98a, Esp98b, EN94].

A number of authors have investigated reduction rules for Petri nets and for various subclasses of Petri nets. In Murata's paper [Mur89], six reduction rules are presented for Petri nets and this set of rules can be used as a starting point for workflow-net reduction rules. In [DE95], a set of reduction rules is proposed for free-choice Petri nets while preserving well-formedness. Berthelot presents a set of reduction rules for general Petri nets [Ber87]. Reduction rules have been suggested to be used together with Petri nets for the verification of workflows (cf. Chapter 4 in [AH04]). Similar approaches have been applied to other languages such as EPCs [KNS92, Kin06], BPMN [Whi09], etc. Six reduction rules that preserve correctness for EPCs including reduction rules for trivial constructs, simple splits and joins, similar splits and joins, XOR loops and optional OR-loops are proposed in [DAV05]. In [SO00] a set of reductions rules for AND-XOR graphs (i.e., a special case of free-choice nets) is presented. The authors claim that these rules are complete (i.e., any correct workflow can be reduced completely). However, as shown in [AHV02, LZLC02] this is not the case. This can be easily corrected by using the reduction rules presented in [DE95] or by using a more direct method (e.g., based on the Rank Theorem as shown in [AHV02]). None of the reduction rules mentioned above takes cancellation into account. This case is handled in [WVA+09]. In [VWAH10] the soundness preserving reduction rules are extended to nets with inhibitor arcs.

In this paper, we focus on WF-nets without resources, interaction, or data. These can be added by introducing resource places (cf. resource-constrained WF-nets [BP98, HSSV05, JKJ10]), communication places (cf. open WF-nets [AMSW09, LMSW06, LW10, MRS05a, MRS05b, Wol09]), or data places ([TAS09]) and analyzed using dedicated techniques.

The focus of this paper is on the verification of Petri-net-based workflow models. Although Petri nets are the most widely used formal method for modeling and analyzing workflows, also process algebraic techniques have been proposed. CSP, CCS, and the pi-calculus have been used to model the so-called workflow patterns and provide semantics for languages such as BPMN and BPEL. For example, [WG07] and [PW09] show how the well-known workflow patterns [AHKB03] can be represented in CSP and pi-calculus respectively. Languages such as BPMN have been mapped onto Petri nets [ODA+09], but also process algebras such as CSP can be used to provide semantics [WG08]. In [WG09] Dwyer's property specification patterns [DAC99] are applied in the context of BPMN, i.e., for a given process model it is checked whether particular temporal properties hold. Model checkers such as FDR can be used to check such properties [WG08, WG09].

We would also like to refer to some empirical work on workflow verification. A detailed analysis of the SAP reference model is presented in [MMN+06, MVD+08]. Here 604 EPC models [KNS92, Kin06] are automatically translated to YAWL [AH05] and analyzed using Petri-net invariants. This study showed, using a simple technique such as invariants, that at least 5.6 percent of SAP's EPC models have obvious flaws (deadlocks, etc.). Many errors resulted from the fact that splits of one type were joined by splits of another type. Another typical problem was that fragments of the same model were modeling different entities. For example, a job application was mixed up with the need to fill a position. As a result, part of the model was about a position while other parts of the model talked about the handling of possibly many applications related to a single position.

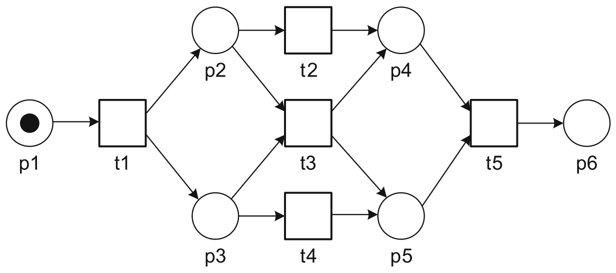

Fig. 2. A basic Petri net with places $\{p1, p2, p3, p4, p5, p6\}$ and transitions $\{t1, t2, t3, t4, t5\}$

Because the SAP model assumes a 1-to-1 correspondence between a position and an application, it deadlocks when there are multiple applications for the same position. Later this study was extended to a larger set of models (more than 2000 EPC models from various sources) using more precise analysis techniques [MNA07]. Of these more than 2000 models at least 10 percent has errors. This more refined analysis showed that 20.9 percent of the SAP models are actually flawed. In [VVL07] a set of 340 real business processes modeled with the IBM WebSphere Business Modeler is analyzed. We would also like to point to the comparative analysis of various workflow verification approaches described in [FFJ+09]. Here large collections of real-life models are used to compare the performance of various tools and techniques.

Empirical research clearly shows that modelers are likely to make errors if they are not supported by analysis tools. This illustrates the practical relevance of the research on workflow verification.

3. Preliminaries

This section introduces basic Petri-net related concepts [DE95, Jen97, Mur89] used in the remainder of this paper.

3.1. Basic Petri nets

First, we informally introduce the classical Petri net. In the next subsection, this model is extended and further formalized.

Definition 3.1 (*Basic Petri net*) A *basic Petri net* is a triple (P, T, F). P is a finite set of *places*, T is a finite set of *transitions* $(P \cap T = \emptyset)$, and $F \subseteq (P \times T) \cup (T \times P)$ is a set of *arcs* (flow relation).

Figure 2 shows a basic Petri net. Places are represented by circles and transitions are represented by squares.

For any relation/directed graph $G \subseteq A \times A$ (including the graph defined by the flow relation of a Petri net F), we define the preset $\bullet a = \{a_1 \mid (a_1, a) \in G\}$ and postset $a\bullet = \{a_2 \mid (a, a_2) \in G\}$ for any node $a \in A$. We use $\overset{G}{\bullet} a$ or $a \overset{G}{\bullet}$ to explicitly indicate the context G if needed. Based on the flow relation F we use this notation as follows. $\bullet t$ denotes the set of input places for a transition t. The notations $t\bullet$, $\bullet p$ and $p\bullet$ have similar meanings, e.g., $p\bullet$ is the set of transitions sharing p as an input place. In the Petri net shown in Fig. 2: $\bullet p5 = \{t3, t4\}$, $p5\bullet = \{t5\}$, $\bullet t3 = \{p2, p3\}$, $t3\bullet = \{p4, p5\}$, etc.

At any time a place contains zero or more *tokens*, drawn as black dots. The state of the Petri net, often referred to as its *marking*, is the distribution of tokens over its places, i.e., $M \in \mathbb{B}(P)$. $\mathbb{B}(P) = P \to \mathbb{N}$ denotes all multi-sets (bags) over P. $M(p)$ denotes the number of times p is included in the multi-set, i.e., how many tokens place p contains in marking M. We assume that the standard operators are defined for bags and hence also for markings, e.g., union $(X + Y)$, difference $(X - Y)$, presence of an element in a multi-set $(x \in X)$, sub-multi-set $(X \leq Y)$, and size $(|X| = \sum_{a \in A} X(a))$. Concrete multi-sets are denoted using square brackets, e.g., $[p1^3, p2^2, p3] = [p1, p1, p1, p2, p2, p3]$ denotes the marking with three tokens in place $p1$, two tokens in place $p2$, and one token in place $p3$.

In the Petri net shown in Fig. 2 only one place is initially marked ($p1$), i.e., $M = [p1]$. Note that more places could be marked in the initial state and that places can be marked with multiple tokens.

For a basic Petri net, we assume the standard *firing rule*, i.e., a transition t is said to be *enabled* with respect to some marking M if and only if each input place p of t contains at least one token. An enabled transition may *fire*, and if transition t fires, then t *consumes* one token from each input place p of t and *produces* one token for each output place p of t. For example, in Fig. 2, $t1$ is enabled and the firing of $t1$ will result in the state that marks places $p2$ and $p3$. In this state $t2$, $t3$, and $t4$ are enabled. If $t2$ fires, $t3$ becomes disabled, but $t4$ remains enabled. Similarly, if $t4$ fires, $t3$ becomes disabled, but $t2$ remains enabled, etc.

In the next subsection, we will formalize the firing rule for an extended class of Petri nets. Before doing so, we introduce some well-known subclasses of Petri nets.

Definition 3.2 (*Net classes*) Let $N = (P, T, F)$ be a basic Petri net.

- N is a *state machine net* if and only if $\forall_{t \in T} \; |\bullet t| = |t \bullet| = 1$.
- N is a *marked graph* if and only if $\forall_{p \in T} \; |\bullet p| = |p \bullet| = 1$.
- N is a *free-choice net* if and only if $\forall_{t_1, t_2 \in T} \; (\bullet t_1 \cap \bullet t_2 \neq \emptyset) \Rightarrow (\bullet t_1 = \bullet t_2)$.

The Petri net shown in Fig. 2 does not fit into any of the classes defined above. If we remove transition $t3$, the resulting net is free-choice. These different net classes are relevant from the viewpoint of analysis. For example, liveness and boundedness (two behavioral properties) can be decided in polynomial time for free-choice nets while this is not the case for non-free-choice nets [DE95].

A Petri net is connected if there is a path from any node (place or transition) to any other node in the graph while ignoring the direction of the arcs, i.e., the Petri net cannot be partitioned in two disconnected parts. In the remainder we assume any Petri net to be connected and having at least two nodes, i.e., at least a place and a transition.

3.2. Extended Petri nets

The basic Petri net model (Definition 3.1) is very simple and is not able to express all routing constructs one may encounter in real-life workflows. However, there are several obvious extensions of the basic model. Some of these extensions enhance the expressiveness (e.g., reset and inhibitor arcs) while other extensions only provide convenient shorthands (e.g., arc weights).

When modeling workflows in term of Petri nets, transitions correspond to activities. Let $\mathcal{A}$ be a universe of activity labels, i.e., $a \in \mathcal{A}$ refers to some activity. Multiple transitions can refer to the same activity, i.e., have the same activity labels. The special label τ refers to a *silent step* [GW96]. We also say that transitions bearing the τ label are "invisible", i.e., transitions not corresponding to any activity and only added for routing purposes. Note that $\tau \notin \mathcal{A}$.

Definition 3.3 (*Extended Petri net*) An *extended Petri net* is a tuple $(P, T, F, \; W, A, L, R, H)$, where:

- (P, T, F) is a basic Petri net,
- $W \in F \to \mathbb{N}\backslash\{0\}$ is an (arc) weight function,
- $A \subseteq \mathcal{A}$ is a set of (activity) labels,
- $L \in T \to A \cup \{\tau\}$ is a labeling function,
- $R \in T \to 2^P$ is a function defining reset arcs, and
- $H \in T \to 2^P$ is a function defining inhibitor arcs.

Figure 3 illustrates the four extensions mentioned in the above definition. Figure 3a shows the extension with arc weights. The arc from place $p1$ to transition $t1$ denotes an ordinary arc, i.e., $W(p1, t1) = 1$ indicating that $t1$ consumes one token from $p1$ when firing. The arc from transition $t1$ to place $p2$ has weight 10, i.e., $W(t1, p2) = 10$ indicating that $t1$ produces ten tokens for $p2$ when firing. We extend the weight function for the situation that there is not an arc connecting two nodes, i.e., $W(x, y) = 0$ if $(x, y) \notin F$. Moreover, for extended nets we redefine the preset and postset operator to return bags rather than sets: $\bullet a = [x^{W(x,y)} \mid (x, y) \in F \; \wedge \; a = y]$ and $a \bullet = [y^{W(x,y)} \mid (x, y) \in F \; \wedge \; a = x]$.

Figure 3b illustrates the notion of transition labels, i.e., each transition has a label. Note that multiple transitions may have the same label, e.g., $L(t1) = L(t3) = X$ in Fig. 3b. The label defines the "observable effect". In workflow terms: the label refers to the activity being executed while firing the corresponding transition. As a convention we will not show labels graphically if the labels coincide with transition identifiers, i.e., if $L(t) = t$, the label is omitted and just the transition identifier is shown (cf. Fig. 3a).

The notion of reset arcs is illustrated in Fig. 3c. Here the four double-headed arcs are reset arcs. Note that $R(tr) = \{p2, p3, p4, p5\}$ and $R(t) = \emptyset$ for all other transitions t. Transition tr is enabled if and only if there is a token in place pr, i.e., reset arcs do not influence enabling. However, after the firing of tr all tokens are removed from the four places $p2$, $p3$, $p4$, and $p5$.

Figure 3d has one so-called inhibitor arc. The arc connecting $p2$ and $t4$ specifies that $p2$ has to be empty when $t4$ fires. Note that $t4$ is enabled if and only if $p3$ contains at least one token and $p2$ contains *no* tokens. Note that in this example this implies that $t2$ has priority over $t4$, i.e., $t4$ can only occur after $t2$ has removed the token from $p2$. Note that $H(t4) = \{p2\}$ and $H(t) = \emptyset$ for all other transitions t.

After this informal introduction of the firing rule and the various extensions, we formalize this notion.

Definition 3.4 (*Firing rule*) Let $N = (P, T, F, W, A, L, R, H)$ be an extended Petri net and $M \in \mathbb{B}(P)$ be a marking.

- A transition $t \in T$ is enabled, notation $(N, M)[t\rangle$, if and only if, $M \geq \bullet t$ and $M(H(t)) = 0$.[1]
- An enabled transition t can fire while changing the state to M', notation $(N, M)[t\rangle(N, M')$, if and only if, $M' = \pi_{P\setminus R(t)}(M - \bullet t) + t\bullet$.[2]

The additional requirement $M(H(t)) = 0$ (i.e., $\sum_{p\in H(t)} M(p) = 0$) states that all places in $H(t)$ need to be empty for t to be enabled. The resulting marking $M' = \pi_{P\setminus R(t)}(M - \bullet t) + t\bullet$ is obtained by first removing the tokens required for enabling: $M - \bullet t$. Then all tokens are removed from the reset places of t using projection. Applying function $\pi_{P\setminus R(t)}$ removes all tokens except the ones in the non-reset places $P\setminus R(t)$. Finally, the specified numbers of tokens are added to the output places. Note that $t\bullet$ is a *bag* of places.

$(N, M)[t\rangle(N, M')$ defines how a Petri net can move from one marking to another by firing a transition. We can extend this notion to firing sequences. Suppose $\sigma = \langle t_1, t_2, \ldots, t_n\rangle$ is a sequence of transitions present in some Petri net N with initial marking M. $(N, M)[\sigma\rangle(N, M')$ means that there is also a sequence of markings $\langle M_0, M_1, \ldots, M_n\rangle$ where $M_0 = M$, $M_n = M'$, and for any $0 \leq i < n$: $(N, M_i)[t_{i+1}\rangle(N, M_{i+1})$. Using this notation we define the set of reachable markings $R(N, M)$ as follows: $R(N, M) = \{M' \in \mathbb{B}(P) \mid \exists_\sigma (N, M)[\sigma\rangle(N, M')\}$. Note that by definition $M \in R(N, M)$ because the initial marking M is trivially reachable via the empty sequence $(n = 0)$.

The notions from Definition 3.4 can easily be lifted to the level of labels. $(N, M)[(a)\rangle$ means that starting in state M it is possible to reach a marking through a (possibly empty) sequence of silent transitions such that a transition t with a visible label a $(L(t) = a \in A)$ becomes enabled. $(N, M)[(\sigma)\rangle(N, M')$ means that there exists some sequence σ' such that $(N, M)[\sigma'\rangle(N, M')$ and the projection of σ' onto its visible labels yields σ.

For a marked Petri net we also define classical behavioral properties such as liveness and boundedness.

Definition 3.5 (*Liveness, boundedness*) Let $N = (P, T, F, W, A, L, R, H)$ be an extended Petri net and $M \in \mathbb{B}(P)$ be a marking.

- (N, M) is live if and only if $\forall_{M'\in R(N,M)}\ \forall_{t\in T}\ \exists_{M''\in R(N,M')}\ (N, M'')[t\rangle$.
- (N, M) is bounded if and only if $R(N, M)$ is finite.
- (N, M) is safe if and only if $\forall_{M'\in R(N,M)}\ \forall_{p\in P}\ M'(p) \leq 1$.

A marked Petri net is live iff from any reachable marking it is possible to (again) enable any transition. A place p is bounded iff there is a $k \in \mathbb{N}$ such that $M'(p) \leq k$ for any reachable marking M'. A marked Petri net is bounded iff all of its places are bounded. This is the case if and only if the number of reachable markings is finite. A net is safe iff the number of tokens per place is bounded by 1, i.e., safeness is a special case of boundedness.

[1] $M(P') = \sum_{p\in P'} M(p)$ denotes the number of tokens in the set of places $P' \subseteq P$, i.e., $M(H(t)) = 0$ if and only if the places in $H(t)$ are all unmarked.

[2] $\pi_A(X)$ denotes the projection of multi-set X onto A, i.e., $(\pi_A(X))(a) = X(a)$ if $a \in A$ and $(\pi_A(X))(a) = 0$ if $a \notin A$.

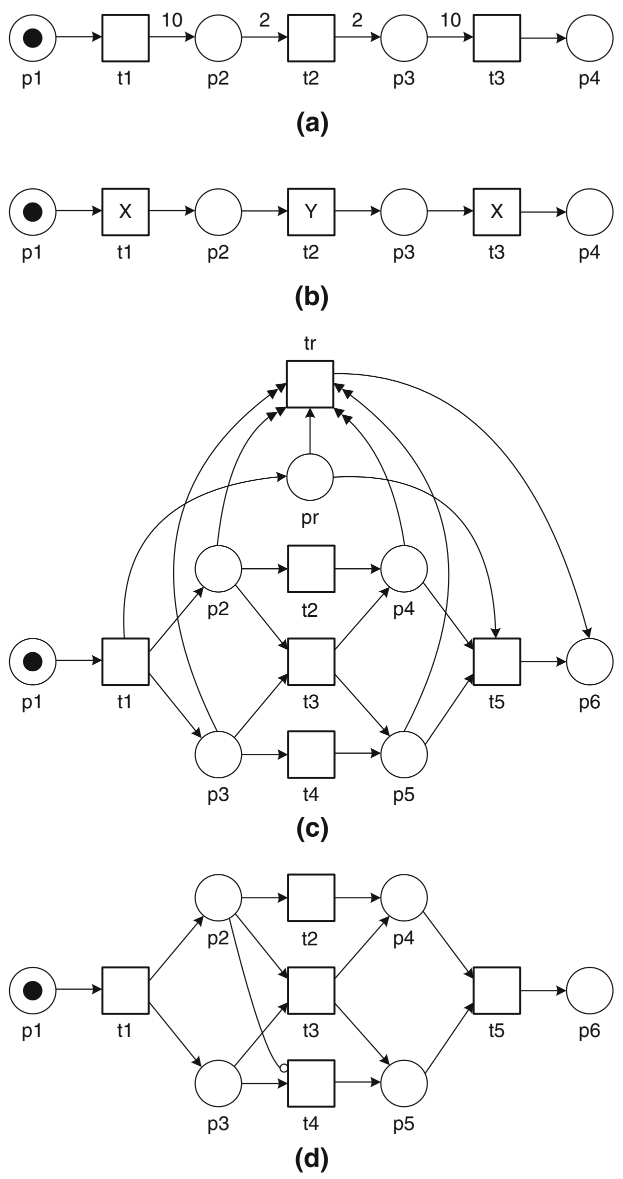

Fig. 3. Four extended Petri nets illustrating the different extensions. **a** Extended Petri net with arc weights. **b** Extended Petri net with transition labels. **c** Extended Petri net with reset arcs. **d** Extended Petri net with inhibitor arcs

4. Workflow nets

In the previous section, we considered arbitrary Petri nets without having an application in mind. However, when looking at workflows, we can make some assumptions about the structure of the Petri net. The idea of a workflow process is that many *cases* (also called *process instances*) are handled in a uniform manner. The workflow definition describes the ordering of *activities* to be executed for each case. For example, the workflow for the handling of insurance claims describes how an individual claim is processed. Although at any point in time there may be many claims in the pipeline, the workflow definition only looks at one case in isolation. Each of the cases will have a well-defined starting point and ending point. There is a point in time where the instance starts, i.e., the process is instantiated for a particular case, and, hopefully, there is a point in time where the instance is completed. Even if one considers multiple start activities and multiple end activities, from a conceptual viewpoint there is still a unique initial state and a unique final state.

These basic assumptions lead to the notion of a *WorkFlow net* (WF-net) [Aal97, Aal98]. Using Fig. 1b we already informally introduced this notion and now it is time to formalize it.

Definition 4.1 (*WF-net*) An extended Petri net $N = (P, T, F, W, A, L, R, H)$ is a WorkFlow net (WF-net) if and only if

- There is a single source place i, i.e., $\{p \in P \mid \bullet p = \emptyset\} = \{i\}$.
- There is a single sink place o, i.e., $\{p \in P \mid p\bullet = \emptyset\} = \{o\}$.
- Every node is on a path from i to o, i.e., for any $n \in P \cup T$: $(i, n) \in F^*$ and $(n, o) \in F^*$ where F^* is the reflexive transitive closure of relation F.
- There is no reset arc connected to the sink place, i.e., $\forall_{t \in T} \; o \notin R(t)$.

Figures 2 and 3 show five WF-nets. In each of these nets $i = p1$ is the source place and either $p4$ or $p6$ is the sink place o. Every node in each of these Petri nets is on a path from i to o. The requirement that $\forall_{t \in T} \; o \notin R(t)$ has been added to emphasize that termination should be irreversible, i.e., it is not allowed to complete (put a token in o) and then undo this completion (remove the token from o).

Transitions in a WF-net correspond to activities. Note that Definition 4.1 allows for multiple start ($i\bullet$) and end ($\bullet o$) activities. It is also easy to generalize this definition to multiple start and end places. However, to simplify notation we assume a single start and end place. Definition 4.1 also does not explicitly address notions such as AND/XOR-splits/joins. By default a transition corresponds to an AND-join/AND-split activity. To model an XOR-join or XOR-split activity, the desired behavior can be added using silent transitions (transitions with label τ) or duplicate transitions (multiple transitions having identical labels). Such transformations are trivial as shown in [Aal98].

Definition 4.1 focuses on the control-flow and abstracts from resources, interaction, and data. While the definition can be extended with additional perspectives, we deliberately abstract from such things for several reasons. First of all, WF-nets are *not* proposed as an end-user language. We envision that people use languages such as (extended) EPCs [Kin06], BPMN [Whi09], etc. or some proprietary workflow language. The control-flow aspects of such languages can be mapped onto WF-nets. We aim to investigate the foundations of workflow modeling and analysis and do not want to focus on a particular language. Resources, interaction, and data can be added by introducing resource places (cf. resource-constrained WF-nets [BP98, HSSV05]), communication places (cf. open WF-nets [LMSW06, MRS05a, MRS05b]), or data places ([TAS09]). When cases compete for resources and resources may be locked by individual cases, these interactions may cause deadlocks. Similarly, dataflow problems may cause livelocks. However, for a more realistic modeling of these perspectives one needs to resort to colored Petri nets [Jen97] or other models allowing for the modeling of more involved resource allocation strategies and dataflow. As a result, analysis (other than simulation) tends to get intractable. Moreover, it may be impossible to accurately model these perspectives. For example, a decision may be based on some complex calculation, external data, or human judgment. Hence, many decisions need to be modeled as non-deterministic choices anyway. Therefore, it seems natural to abstract from these aspects and first analyze the control-flow in isolation as further motivated in [Aal98] and many other papers on workflow verification.

5. Soundness

Based on the notion of WF-nets we now investigate the fundamental question: "Is the workflow correct?". If one has domain knowledge, this question can be answered in many different ways. For example, one can express particular properties in temporal logics such as LTL, CTL, etc. As shown in [WG09] Dwyer's property specification patterns [DAC99] can be applied in this context. In this paper, we focus on generic questions not requiring any domain knowledge Hence, one needs to resort to generic questions such as: "Does the workflow terminate?", "Are there any deadlocks?", "Is it possible to execute activity A?", etc. Such kinds of generic questions triggered the definition of *soundness* [Aal97, Aal98]. In this paper, we consider different soundness notions. However, we first start with the original definition given in [Aal97].

Definition 5.1 (*Classical soundness* [Aal97, Aal98]) Let $N = (P, T, F, W, A, L, R, H)$ be a WF-net. N is sound if and only if the following three requirements are satisfied:

- Option to complete: $\forall_{M \in R(N,[i])}\ [o] \in R(N, M)$.
- Proper completion: $\forall_{M \in R(N,[i])}\ (M \geq [o]) \Rightarrow (M = [o])$.
- No dead transitions: $\forall_{t \in T}\ \exists_{M \in R(N,[i])}\ (N, M)[t\rangle$.

Figure 1b was used to informally introduce the notion of soundness. Note that here $i = start$ and $o = end$. Each of the five WF-nets depicted in Figs. 2 and 3 is sound.

The first requirement in Definition 5.1 states that starting from the initial state (just a token in place i), it is always possible to reach the state with one token in place o (state $[o]$). If we assume a strong notion of fairness, then the first requirement implies that eventually state $[o]$ is reached. Strong fairness, sometimes also referred to as "impartial" or "recurrent" [KA99], means that in every infinite firing sequence, each transition fires infinitely often. Note that weaker notions of fairness are not sufficient, see Figure 2 in [KA99]. However, such a fairness assumption is reasonable in the context of workflow management since all choices are made (implicitly or explicitly) by applications, humans or external actors. If we required termination without this assumption, all nets allowing loops in their execution sequences would be called unsound, which is clearly not desirable. The second requirement states that the moment a token is put in place o, all the other places should be empty. The last requirement states that there are no dead transitions (tasks) in the initial state $[i]$. It is easy to see that the second requirement is implied by the first one.

As pointed out in [Aal97, Aal98], classical soundness of a WF-net without reset and/or inhibitor corresponds to liveness and boundedness of the so-called short-circuited net. The short-circuited net is the Petri net obtained by connecting o to i, thus making the net cyclic.

Lemma 5.1 (Soundness, liveness and boundedness) Let N be a WF-net and $\overline{N}$ the extended Petri net obtained by connecting o to i through a new transition t^*.

- If N is sound, then $(\overline{N}, [i])$ is live.
- If N has no reset and inhibitor arcs, then N is sound if and only if $(\overline{N}, [i])$ is live and bounded.

Proof. If N is sound, then the moment a token is put into o all other places are empty. Hence, t^* can only bring the net back to the initial state. Moreover, the set of reachable states does not change by adding t^* and all transitions remain non-dead. Since $(\overline{N}, [i])$ can return to the initial state again and again, the net is live.

If N has no reset and inhibitor arcs, then the results presented in [Aal98] apply. Hence soundness coincides with liveness and boundedness of the short-circuited net. □

Note that if N is sound, the net does not need to be bounded. For example, there could be a reset arc removing an arbitrary (i.e., unbounded) number of tokens before producing a token for o.

After the initial paper on soundness of WF-nets [Aal97, Aal98] many other papers followed. Some extend the results while others explore alternative notions of soundness. In the remainder of this section we define seven alternative notions described in the literature. These notions strengthen or weaken some of the requirements mentioned in Definition 5.1.

The first notion of soundness focuses on the "option to complete", i.e., the first requirement in Definition 5.1. Moreover, this notion is parameterized with a variable k which indicates the initial number of tokens in the source place.

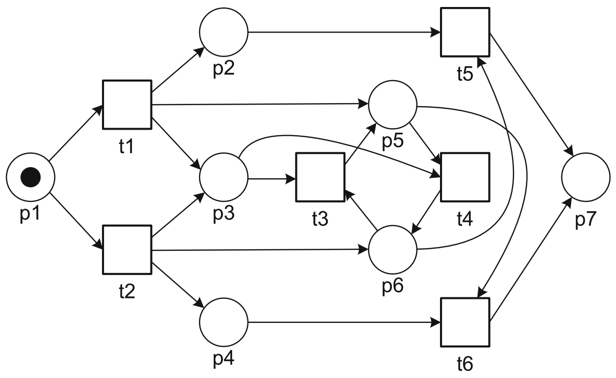

Fig. 4. A WF-net that is 1-sound but not 2-sound

Definition 5.2 (*k-soundness* [HSV03, HSV04]) Let N be a WF-net. N is k-sound if and only if $\forall_{M \in R(N,[i^k])}\, [o^k] \in R(N, M)$.

Note that any WF-net is 0-sound and that 1-soundness corresponds to the first requirement in Definition 5.1. In Lemma 11 in [HSV04] it is shown that also for k-soundness the "option to complete" implies "proper completion" (like for classical soundness). We introduce k-soundness mainly to be able to define useful notions such as weak soundness and generalized soundness.

Consider the WF-net shown in Fig. 4. This WF-net is classical sound and therefore also 1-sound. From the marking $[p1]$ there are two firing sequences both leading to $[o]$: $\langle t1, t4, t5\rangle$ and $\langle t2, t3, t6\rangle$. However, the net is not 2-sound, e.g., the sequence $\langle t1, t1, t4, t3\rangle$ results in a deadlock not being the desired final state. In fact it is not sound for any $k \geq 2$.

The notion of 1-soundness is also known as weak soundness [Mar03, Mar05a].

Definition 5.3 (*Weak soundness* [Mar03, Mar05a]) Let N be a WF-net. N is weak sound if and only if N is 1-sound.

Figure 5 shows three WF-nets that are all weak sound. The WF-net shown in Fig. 5a is not classical sound, because transition $t3$ is dead. It is also not 2-sound because it is possible to reach state $[p3]$ when starting in $[p1^2]$ (i.e., a token is missing in the final state). Figure 5b is not 2-sound because of a similar problem (state $[p3]$ is reachable from $[p1^2]$). Figure 5c is not 2-sound because the workflow may deadlock in state $[p1, p3]$. It is easy to see that similar problems occur when $k > 2$.

For a given k, it is easy to construct a WF-net that is k-sound but not $k + 1$-sound. Therefore, we define the notion of up-to-k-soundness.

Definition 5.4 (*up-to-k-soundness* [Too04]) Let N be a WF-net. N is up-to-k-sound if and only if N is l-sound for all $0 \leq l \leq k$

Generalized soundness [HSV03, HSV04] intuitively corresponds to up-to-∞-soundness, i.e., k-sound for any $k \in \mathbb{N}$.

Definition 5.5 (*Generalized soundness* [HSV03, HSV04]) Let N be a WF-net. N is generalized sound if and only if for all $k \in \mathbb{N}$: N is k-sound.

The soundness notions discussed so far consider all possible execution paths and if for one path the desired end state is not reachable, the net is not sound. In a way this implies that the workflow is "lunacy proof", e.g., the user cannot select a path that will deadlock. The notion of relaxed soundness assumes a responsible user or environment, i.e., the net does not have to be "lunacy proof" as long as there exist "good" execution paths.

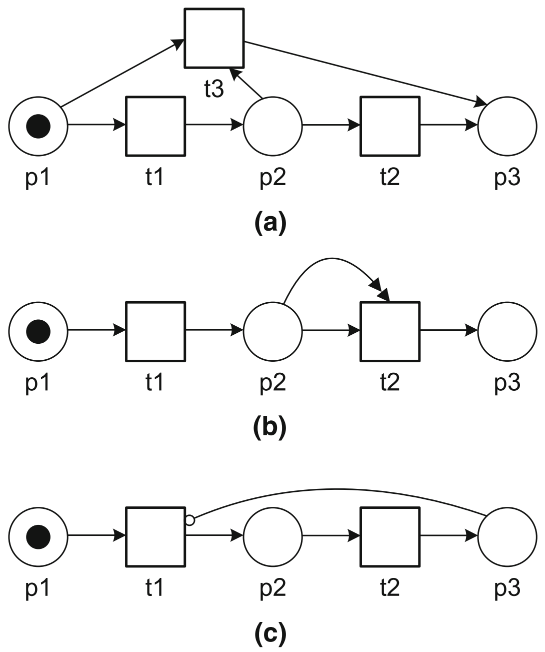

Fig. 5. Some more WF-nets that are 1-sound but not k-sound for any $k \geq 2$. **a** A 1-sound net, **b** a classical sound net, **c** a classical sound net

Definition 5.6 (*Relaxed soundness* [DA04, DR01]) Let N be a WF-net. N is relaxed sound if and only if for each transition $t \in T$: $\exists_{M,M' \in R(N,[i])}\ (N, M)[t\rangle(N, M') \wedge [o] \in R(N, M')$.

Figure 6a shows a net that is not weak sound but that is relaxed sound. Note that firing sequences $\langle t2, t4, t6\rangle$ and $\langle t1, t3, t4, t5\rangle$ represent "good" executions ending in $[p6]$ and covering all transitions. It is assumed that the incorrect firing sequence $\langle t1, t3, t4, t6\rangle$ is avoided. As shown in [DA04] it is possible to automatically convert a relaxed sound WF-net into a sound WF-net by blocking the undesired paths, i.e., the workflow engine can select the "good" behavior. The relaxed sound WF-net in Fig. 6a can be converted into a sound WF-net by adding a place connecting $t2$ and $t6$.

The soundness notions discussed so far focus on ending in a state with no tokens in any place other than the sink place. Lazy soundness weakens this requirement, i.e., tokens may be left behind as long as the sink place is marked precisely once.

Definition 5.7 (*Lazy soundness* [PW06b, PW06a]) Let N be a WF-net. N is lazy sound if and only if the following two requirements are satisfied:

- Option to complete: $\forall_{M \in R(N,[i])}\ \exists_{M' \in R(N,M)}\ M'(o) = 1$.
- Proper completion: $\forall_{M \in R(N,[i])}\ M(o) \leq 1$.

The net in Fig. 6b is lazy sound. Note that $t3$ can even fire repeatedly after putting a token in $p5$. The last notion of soundness, named easy soundness, only considers the possibility of the option to complete.

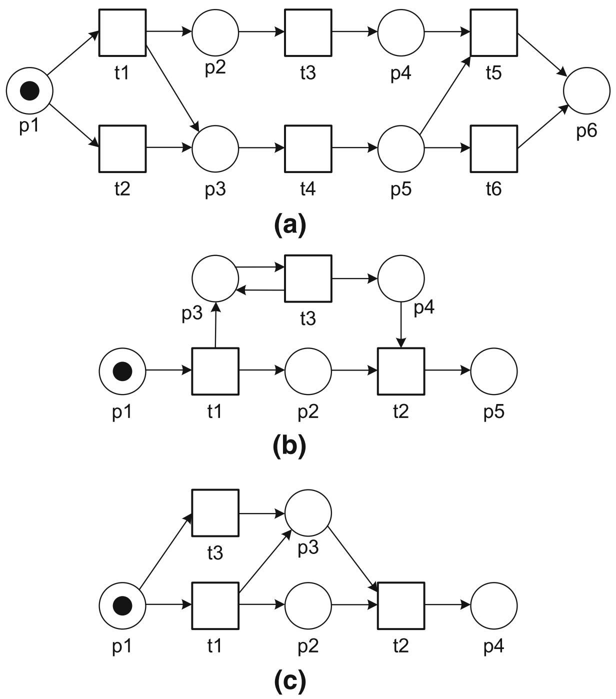

Fig. 6. Some more WF-nets illustrating relaxed, lazy, and easy soundness. **a** A relaxed sound WF-net, **b** a lazy sound net, **c** an easy sound net

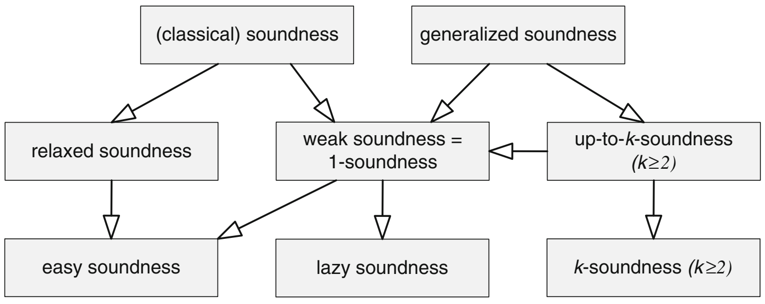

Fig. 7. Relationships between different various kinds of soundness (→ = "implies")

Definition 5.8 (*Easy soundness* [Too04]) Let N be a WF-net. N is easy sound if and only $[o] \in R(N, [i])$.[3]

Figure 6c is easy sound but does not satisfy any of the other 7 soundness notions. It is easy sound because the firing sequence $\langle t1, t2\rangle$ indeed leads from $[i]$ to $[o]$.

While introducing the various soundness notions, we already indicated that some soundness notions are stronger than other soundness notions. Figure 7 shows the different implications. Classical soundness implies relaxed soundness and weak soundness. Weak soundness corresponds to the first requirement of classical soundness. Relaxed soundness corresponds to a weakening of this first requirement. It is trivial to see that generalized soundness implies weak soundness and up-to-k soundness. Up-to-k soundness of course also implies k soundness. Weak soundness implies lazy soundness because it weakens the "option to complete" and "proper completion" requirements by just considering sink place o. Easy soundness is implied by relaxed soundness and by weak soundness since both imply that there is at least one path from $[i]$ to $[o]$. Note that Fig. 7 only shows the transitive reduction of all implications, e.g., because of transitivity generalized soundness also implies lazy soundness.

The eight soundness notions mentioned in Fig. 7 will be considered in the remainder of this paper. Several other soundness notions have been defined in the literature. For example, in the context of open WF-nets [LMSW06, MRS05a, MRS05b], cross-organizational WF-nets [KMR00], and interacting BPMN/pi-calculus processes [PW06a] additional notions of soundness have been defined. However, these notions do not look at a single WF-net in isolation and therefore are outside the scope of this paper. Note that when considering interaction and/or resources different notions are needed.

6. Decidability

In this section we explore the different notions of soundness and the various classes of WF-nets and their decidability. Here we will focus on extended WF-nets and the eight notions of soundness defined earlier. First, we show that arc weights and transition labels are not relevant for decidability. Then, we show that WF-nets with inhibitor arcs are more expressive than nets with reset arcs. Based on these initial insights we present our decidability results.

6.1. Removing arc weights and labels

This subsection shows that arc weights and transition labels are not relevant for decidability. It is easy to see that transition labels do not influence soundness. Moreover, as Fig. 8 shows, weighted arcs can also be removed. In Fig. 8 it is assumed that k is the maximal arc weight on the input side and output side of p. Note that there happens to be an input transition producing k tokens and an output transition consuming k tokens. In the general case there may be arbitrarily many input and output transitions as long as they consume/produce not more than k tokens from/for place p. Suppose there is an output transition which consumes $l \leq k$ tokens from p in Fig. 8a. In Fig. 8b this transition will have l out of the k newly added places as input. Any subset of these places will do, e.g., $\{p_1, \ldots, p_l\}$.

Note that the construction does not introduce any new type of arcs, e.g., inhibitor arcs are only needed if they were already present in the original net. Moreover, if the initial net has a WF-net structure, the resulting net also has this structure.

Since we can abstract from arc weights and transition labels, we restrict ourselves to *core WF-nets* in the remainder.

Definition 6.1 (*Core WF-nets*) A *core WF-net* is a WF-net (P, T, F, W, A, L, R, H) where $A = T$, for all $t \in T$: $L(t) = t$, and for all $f \in F$: $W(f) = 1$. A core WF-net can be represented by (P, T, F, R, H).

Since arc weights and transition labels are just "syntactical sugaring", we do not need to consider them when investigating soundness. For example, if generalized soundness is decidable for WF-nets without arc weights, then it will also be decidable for WF-nets with arc weights.

[3] In [Too04] this notion was named weak soundness but interpreted differently from [Mar03, Mar05a]. Hence, in this paper this notion is referred to as "easy" soundness to avoid confusion.

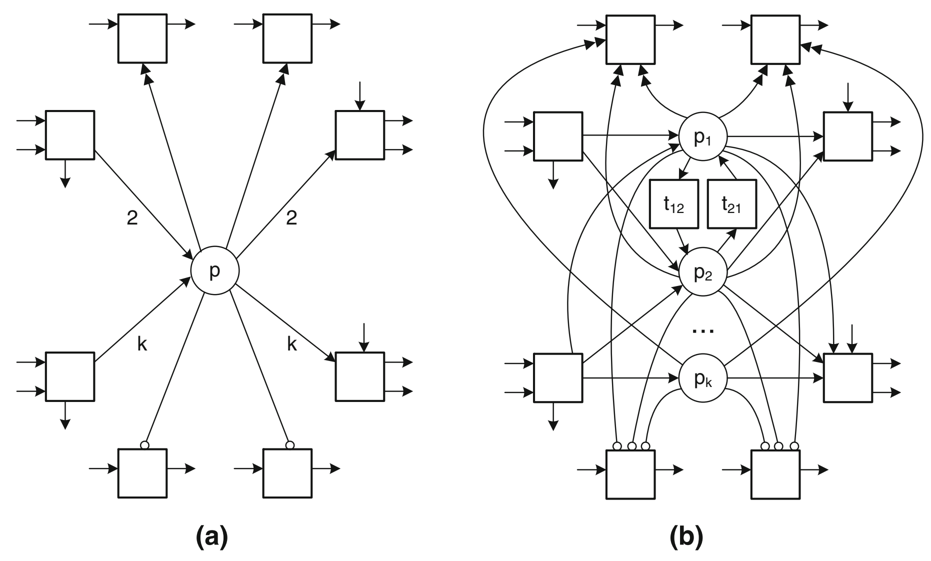

Fig. 8. Removing arc weights: place p is split up into k places and transitions are added such that tokens can freely move among these places. The arc weights 2 and k can be replaced by any weight $1 \leq l \leq k$ and the construction also works for more/less input/output transitions. **a** Net with arc weights. **b** Net without arc weights

6.2. Inhibitor arcs can emulate reset arcs

Reset and inhibitor arcs are clearly more than "syntactical sugaring" and really add to the expressiveness, i.e., one can construct WF-nets with reset or inhibitor arcs that do not have an equivalent WF-net without such arcs.[4] However, as shown below, WF-nets with reset arcs can be translated into equivalent WF-nets with inhibitor arcs. First, we show this for arbitrary extended Petri nets.

Proposition 6.1 (Inhibitor arcs can emulate reset arcs) *Let N be an extended Petri net. All reset arcs can be replaced by inhibitor arcs without changing the behavior* (*modulo branching bisimulation* [GW96]).

Proof. It is easy to see that the construction shown in Fig. 9 can be used to remove any reset arc. Transition t is replaced by a start transition t_s and an end transition t_e. The start transition t_s has the same label as transition t and the end transition t_e has a τ label. Each transition with a reset arc is replaced by a small network as shown in Fig. 9. Place x is added to guarantee mutual exclusion, i.e., there is one such place and all original transitions in the new net consume a token from x and return a token to x while all new start transitions consume a token from x and all new end transitions produce a token for x. As a result, any firing of t_s is followed by zero of more firings of t_c, followed by one firing of t_e. The effect of firing in one sequence $t_s(t_c)^n t_e$ is equivalent to firing t in the original. It is easy to establish a branching bisimulation relation between both nets by associating the label of t to t_s and giving transitions t_c and t_e a τ label. □

Proposition 6.1 is also applicable to (core) WF-nets, i.e., by adding a new global start and end transition, place x can be marked and unmarked and the "WF-net structure" remains intact.

In the remainder of this section we investigate decidability for the eight notions of soundness and WF-nets with reset and/or inhibitor arcs. Here, we often use the following corollary.

[4] Note that the term "equivalent" is ill defined in this context. Although this statement is not very sensitive to the notion of equivalence considered, one can think of standard equivalence notions such as branching bisimulation [GW96].

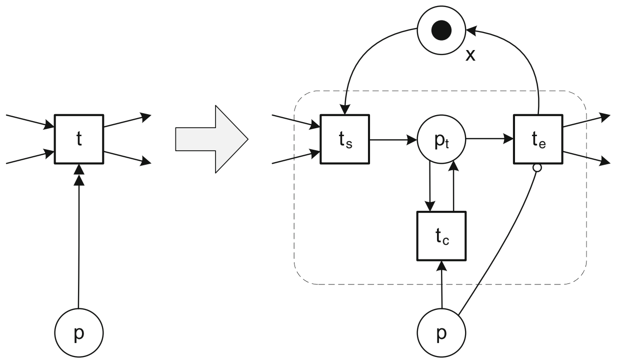

Fig. 9. Construction showing that reset arcs can be translated into inhibitor arcs

Corollary 6.1 (Undecidability of soundness for WF-nets with inhibitor arcs) *If a particular soundness property (classical soundness, k-soundness, weak soundness, etc.) is undecidable for WF-nets with reset arcs it will also be undecidable for WF-nets with inhibitor arcs.*

This corollary follows directly from Proposition 6.1. Any WF-net with reset arcs can be converted into a Petri net without reset arcs and just inhibitor arcs. Note that the resulting net is not a WF-net because of the initially marked x place. However, by adding a new "start transition" and a new "end transition", the net can be transformed into an equivalent WF-net.

6.3. Classical soundness

In this subsection, we explore the decidability of soundness for WF-nets. If a WF-net has no reset and no inhibitor arcs, we can use Lemma 5.1 to show that soundness is decidable. Such a WF-net N is sound if and only if $(\overline{N}, [i])$ is live and bounded. Since liveness and boundedness are both decidable, soundness is also decidable. For some subclasses (e.g., free-choice nets), this is even decidable in polynomial time [Aal97, Aal98].

Unfortunately, soundness is not decidable for WF-nets with reset and/or inhibitor arcs. First, we show that reset arcs make the verification problem undecidable.

Theorem 6.1 (Undecidability of soundness) *Soundness is undecidable for WF-nets with reset arcs.*

Proof. Reachability is known to be undecidable for reset nets while coverability is decidable. In [AHH$^+$08, AHH$^+$09] a construction is given to relate reachability to soundness. The main idea is to provide a construction such that the reachability of a particular marking M of an arbitrary marked net (N, M_I) corresponds to the non-soundness of a larger WF-net N' which embeds N, i.e., M is reachable from (N, M_I) if and only if N' is *not* sound. The construction is rather tricky and involved, therefore, it is not shown here. □

Theorem 6.1 shows that the ability of cancellation combined with unbounded places makes soundness undecidable. This is a relevant result because many workflow languages have such features. Since inhibitor arcs can emulate reset arcs (cf. Proposition 6.1), the undecidability result also applies to WF-nets with inhibitor arcs.

6.4. Weak soundness

Next, we investigate the decidability of weak soundness, also known as 1-soundness. Weak soundness corresponds to the first requirement of classical soundness. Since the second requirement is implied by the first one, the only difference is the third requirement, i.e., for weak soundness it is not required that there are no dead transitions. From the viewpoint of decidability this is less relevant because dead transitions can be removed from a WF-net with reset arcs.

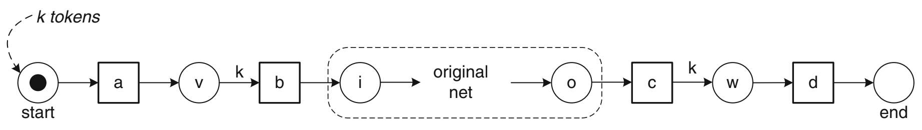

Fig. 10. The original WF-net is weak sound if and only if the constructed WF-net is k-sound. Because k tokens are put in place *start*, transition b fires once to initialize the original net. If the original net puts one token in o, then c can fire once followed by k executions of b, thus producing k tokens for place *end*

Corollary 6.2 (Undecidability of weak soundness) *Weak soundness is undecidable for WF-nets with reset arcs.*

Proof. Let N be an arbitrary WF-net with reset arcs but no inhibitor arcs. Remove all dead transitions in $(N, [i])$ and let N' be the resulting WF-net. This is possible because coverability is decidable for reset nets [DFS98, FS01] and therefore it is possible to check for each transition t whether $\bullet t$ is coverable from the initial marking (see also the proof of Theorem 6.1). Now N is weak sound if and only if N' is sound. Since Theorem 6.1 shows that soundness is undecidable, weak soundness is also undecidable. □

If a WF-net has no reset and no inhibitor arcs, weak soundness is decidable. Note that we can first remove all dead transitions from N, then soundness corresponds again to liveness and boundedness of the short-circuited net, which is decidable [Aal97, Aal98].[5] It is also obvious that, in this case, weak soundness can be checked by simply inspecting the well-known coverability graph [Mur89].

Because of Corollary 6.1, soundness is also undecidable for WF-nets with inhibitor arcs.

6.5. *k*-Soundness

Now we consider the situation of k-soundness with $k \geq 2$. Just like 1-soundness, the decidability results are equal to classical soundness. The question whether a net is weak sound can be translated into a k-soundness question, and vice versa, for any k as shown in Fig. 10.

Corollary 6.3 (Undecidability of k-soundness) *For any $k \geq 2$: k-soundness is undecidable for WF-nets with reset arcs.*

Proof. We will show that decidability of k-soundness for WF-nets with reset arcs would imply decidability of weak soundness for WF-nets with reset arcs, which would contradict Theorem 6.1. Let N be an arbitrary WF-net with reset arcs. From N we construct the WF-net N_k which embeds N and has parameter $k \geq 2$ as shown in Fig. 10. Clearly, N_k is a WF-net. Again the arc weights can be removed as shown in Fig. 8; this is the reason for adding the additional transitions at the beginning and end. Moreover, it is easy to verify that N is weak sound if and only if N_k is k-sound. Therefore, we can apply Theorem 6.1 to show undecidability. □

Note that k-soundness is also undecidable for WF-nets with inhibitor arcs (cf. Corollary 6.1), but is decidable for WF-nets without reset and/or inhibitor arcs.

6.6. Up-to-*k*-soundness

Up-to-k-soundness can be translated into k "l-soundness" problems where $1 \leq l \leq k$. Therefore, intuitively it is no surprise that the results are identical.

[5] The removal of dead transitions could potentially transform a WF-net into a non-WF-net. However, this can be repaired easily by adding a new source place i_s, start transition t_s, sink place o_e, end transition t_e, and self-loop place x. t_s consumes a token from the new source place i_s and produces a token for the old source place i and the self-loop place x. t_e consumes a token from the old sink place o and x, and produces a token for the new sink place o_e. All original transitions have x as a self loop, i.e., they consume a token from and produce a token for x.

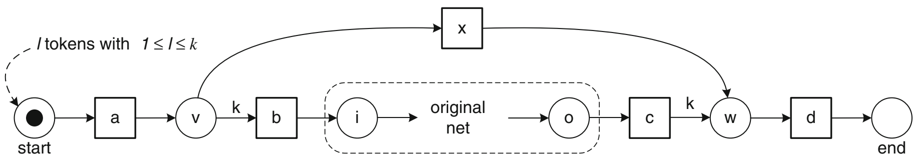

Fig. 11. The original WF-net is weak sound if and only if the constructed WF-net is up-to-k-sound

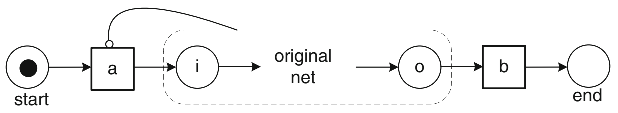

Fig. 12. The original WF-net is weak sound if and only if the constructed WF-net is generalized sound. Note that a short-hand notation is used; transition a has an inhibitor arc for each place in the original net

Corollary 6.4 (Undecidability of up-to-k-soundness) *For any $k \geq 2$: up-to-k-soundness is undecidable for WF-nets with reset arcs.*

Proof. We will show that decidability of up-to-k-soundness for WF-nets with reset arcs would imply decidability of weak soundness for WF-nets with reset arcs, which would contradict Theorem 6.1. Let N be an arbitrary WF-net with reset arcs. From N we construct the WF-net N_k which embeds N and has parameter $k \geq 2$ as shown in Fig. 11. This net is identical to the one in Fig. 10 apart from the "bypass" transition x. N is weak sound if and only if N_k is up-to-k-sound and therefore the latter is also undecidable. Note that if less than k tokens are put into the source place, the original net N is not activated and the tokens bypass N via x. Therefore, k-soundness of the net in Fig. 10 is the same as up-to-k-soundness of the net in Fig. 11. □

Using Corollary 6.1 it can be shown that up-to-k-soundness is also undecidable for WF-nets with inhibitor arcs.

6.7. Generalized soundness

While weak-soundness and k-soundness are closely related to classical soundness, generalized soundness is quite different because it marks the source place with an arbitrary (i.e., non-predefined) number of tokens. Let us first consider the situation without reset and/or inhibitor arcs. Even in this simple setting it is not possible to use the coverability graph [Mur89] to decide soundness. The reason is that the problem corresponds to inspecting infinitely many coverability graphs. Fortunately, in [HSV04] it was shown that generalized soundness is decidable for WF-nets without reset and/or inhibitor arcs.

Theorem 6.2 (Decidability of generalized soundness [HSV04]) *Generalized soundness is decidable for WF-nets without reset and/or inhibitor arcs.*

Proof. See [HSV04]. □

Let us now consider a WF-net with inhibitor arcs and no reset arcs. It is well-known that the reachability problem is undecidable for Petri nets with inhibitor arcs [CW99]. This can be used to prove that generalized soundness is undecidable.

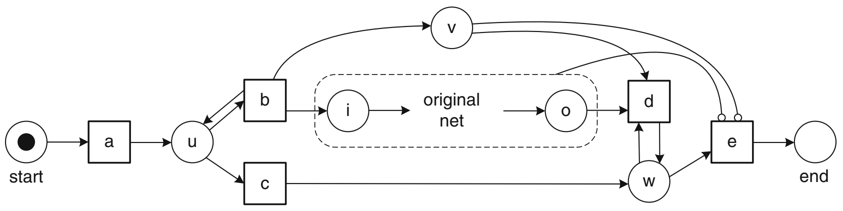

Fig. 13. Construction showing that generalized soundness can be expressed in terms of weak soundness for WF-nets with inhibitor arcs

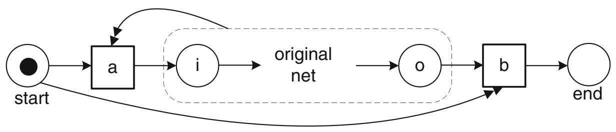

Fig. 14. Failed attempt to show that generalized soundness is undecidable for WF-nets with reset arcs

Proposition 6.2 (Undecidability of generalized soundness) *Generalized soundness is undecidable for WF-nets with inhibitor arcs.*

Proof. Let N be an arbitrary inhibitor WF-net. Let N' be the net obtained using the construction shown in Fig. 12.

For any k and any reachable state of $(N', [start^k])$, a blocks if there is still a token in the original net and will continue to block until b removes the last token from N.

Assume N is weak sound. It is easy to see that any token put into i, can always evolve to a state with a token in o and where the rest of the original net is empty and a continues to block. In this state b can fire and this can be repeated for all k tokens initially put in place $start$. Hence N' is generalized sound.

Assume N' is generalized sound. Hence the net N' is also weak sound which implies that N is weak sound.

Therefore, N' is generalized sound if and only if N is weak sound. Since weak soundness is undecidable for WF-nets with inhibitor arcs, generalized soundness is also undecidable. □

In Fig. 12, the original WF-net N is activated only once, i.e., a blocks any new activations until b removes the last token from N. This is sufficient for proving undecidability. However, using Fig. 13, we also explore the reverse situation. Let N be the original net in Fig. 13 and let N' be the extended WF-net as shown in the same figure. N is generalized sound if and only if N' is weak sound. This construction provides additional insight in the relation between weak and generalized soundness.

Generalized soundness is decidable for WF-nets without reset/inhibitor arcs and undecidable for WF-nets with inhibitor arcs. Therefore, the remaining question is: "Is generalized soundness decidable for WF-nets with reset arcs?". The answer to this question is unknown, i.e., it is still an open problem.

One may think that a construction similar to Fig. 12 is possible for reset arcs. Such an attempt is made in Fig. 14. Let N be the original net in Fig. 14 and let N' be the extended WF-net as shown in the same figure. The goal would be to show that N is generalized sound if and only if N' is weak sound. However, this construction does not work because the N' produces a single token for place *end*, independent of the number of tokens initially put in *start*. The thing that is missing is a "counter" like place v in Fig. 13. However, using reset arcs it is impossible to make such a construction. It is also not possible to use the construction of Theorem 6.1, because if multiple cases enter the construction, the original net fragment is able to reach states not reachable from M_I. It is not possible to temporarily block the entry of additional cases, because the WF-net starts empty (i.e., no "mutex place" is possible) and no inhibitor arcs are allowed. Therefore, decidability of generalized soundness for WF-nets with reset arcs remains an open problem.

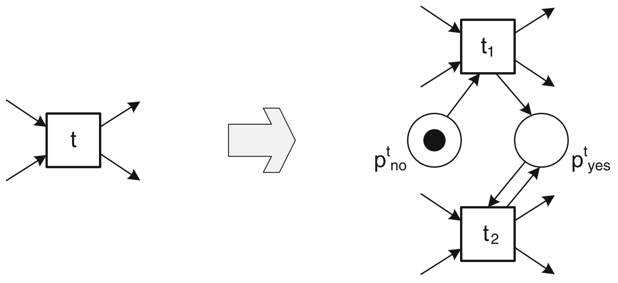

Fig. 15. Construction showing that relaxed soundness is decidable for WF-nets without reset/inhibitor arcs, i.e., t is involved in a "good run" in the original model (*left*) if and only if $[o, p^t_{yes}]$ is reachable in the adapted model (*right*)

6.8. Relaxed soundness

Relaxed soundness differs fundamentally from notions such as classical, weak, and generalized soundness, because it allows for deadlocks, etc. as long as there is a "good execution" possible for each transition.

Theorem 6.3 (Undecidability of relaxed soundness) *Relaxed soundness is undecidable for WF-nets with reset arcs.*

Proof. Unlike the proof of Theorem 6.1, relaxed soundness is directly related to reachability. The main idea is to provide a construction such that the reachability of a particular marking M of an arbitrary marked net (N, M_I) corresponds to the relaxed soundness of a larger WF-net N' which embeds N, i.e., M is reachable from (N, M_I) if and only if N' is relaxed sound. The construction of such a WF-net is non-trivial. Hence, we refer to the detailed proof provided in [AHH+08, AHH+09]. □

Combining Theorem 6.3 and Proposition 6.1 shows that relaxed soundness is also undecidable for WF-nets with inhibitor arcs.

Relaxed soundness is decidable for WF-nets without reset/inhibitor arcs. In most cases, a direct inspection of the coverability graph will be sufficient to conclude this. Often one can also use a partially constructed reachability graph to show relaxed soundness. If the net is unbounded, there may be cases where such simple inspections are inconclusive. A conclusive strategy to check for relaxed soundness would be to add two places (e.g., p^t_{yes} and p^t_{no}) for transition t that record whether t has been fired or not. Note that transition t needs to be duplicated into t_1 and $t_{\geq 2}$ as shown in Fig. 15. Transition t_1 corresponds to the first execution of t while $t_{\geq 2}$ corresponds to later executions. A "good execution" in the original model leads from $[i]$ to $[o]$. Now the reachability of state $[o, p^t_{yes}]$ corresponds to a "good" execution sequence where t occurred at least once. This can be repeated for all transitions, and, since reachability is decidable for classical Petri nets, this implies that relaxed soundness is decidable for WF-nets without reset/inhibitor arcs.

6.9. Lazy soundness

Lazy soundness focuses on the marking of place o and does not require the net to be empty after putting a token in o. Nevertheless, we can use the construction of Theorem 6.1 to show that the property is undecidable for WF-nets with reset arcs

Theorem 6.4 (Undecidability of lazy soundness) *Lazy soundness is undecidable for WF-nets with reset arcs.*

As a result, lazy soundness is also undecidable for WF-nets with inhibitor arcs. The property is, however, decidable for WF-nets without reset and/or inhibitor arcs, e.g., by inspecting the coverability graph.

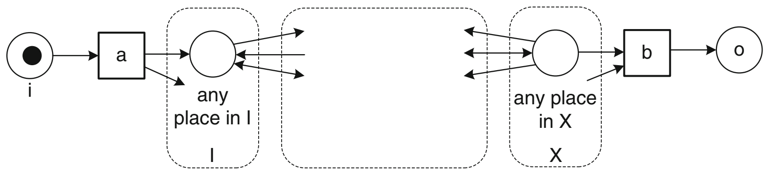

Fig. 16. Construction showing that easy soundness is undecidable for WF-nets with reset arcs

6.10. Easy soundness

The last soundness notion we consider is easy soundness. Recall that this notion simply checks whether there is an execution path from $[i]$ to $[o]$.

Theorem 6.5 (Undecidability of easy soundness) *Easy soundness is undecidable for WF-nets with reset arcs.*

Proof. Let (N, M_I) be an arbitrarily marked Petri net with reset arcs but no inhibitors. Without loss of generality we again assume that N is connected and that every transition has input and output places.

To show that easy soundness is undecidable, we construct a new net $(N', [i])$ which embeds (N, M_I) such that N' is easy sound if and only if some marking M_X is reachable from (N, M_I). By doing so, we show that reachability in an arbitrary reset net can be analyzed through soundness, making easy soundness also undecidable.

Figure 16 shows N and N' using the notation used before.

- Assume marking M_X is reachable from (N, M_I). This implies that M_X is also reachable in $(N', [i])$ and that b can fire without leaving any tokens behind in the set of places of the original net. Hence, firing b results in $[o]$ and clearly the net is easy sound.
- Assume N' is easy sound, i.e., there is a firing sequence leading from $[i]$ to $[o]$. Hence there was a moment when b fired. If at this point in time the marking was not exactly M_X, then tokens are left behind in the original net. Since all transitions have input and output places, the remaining tokens cannot be removed completely and hence all subsequent markings are larger than $[o]$. Hence, just before b fired, the marking was exactly M_X showing that M_X is indeed reachable from (N, M_I). □

Easy soundness is therefore also undecidable for WF-nets with inhibitors. The property is decidable for WF-nets without reset/inhibitor arcs because reachability is decidable for classical Petri nets.

6.11. Summary

As shown, all eight soundness properties are decidable for WF-nets without reset and/or inhibitor arcs. For WF-nets with reset arcs the decidability of generalized soundness is still unknown. For all other cases, soundness is undecidable. Nevertheless, there are pragmatic approaches that allow for the discovery of design errors even if such features are present. This will be explained further in the next section.

7. Analysis

In this section, we focus more on the pragmatic side of workflow verification. While for WF-nets without reset and/or inhibitor arcs all soundness properties are decidable, for extended WF-nets (e.g., nets with reset and/or inhibitor arcs) these notions are typically undecidable. Note that most workflow and process modeling languages (e.g., BPMN, EPCs, Staffware, BPEL, FileNet, etc.) have a control-flow language where the basic elements correspond to WF-nets without reset and/or inhibitor arcs, while the more advanced constructs require reset or inhibitor arcs. Therefore, one can argue that, as a rule-of-thumb, for simple models (independent of the language used) any form of soundness is decidable while for models using notions such as priorities, cancellation, etc. no form of soundness is decidable. However, even if more advanced constructs are used, workflow verification is still possible! Note that even if soundness is undecidable for a particular class of WF-nets, for many representatives of such a class, it may still be possible to conclude soundness or non-soundness. There may be rules of the form "If WF-net N has property X, then N is sound" or "If WF-net N has property Y, then N is not sound". As shown in [MNA07, MMN+06, MVD+08] it is possible to find many errors using such an approach. In [MMN+06, MVD+08] it was shown that at least 5.6 percent

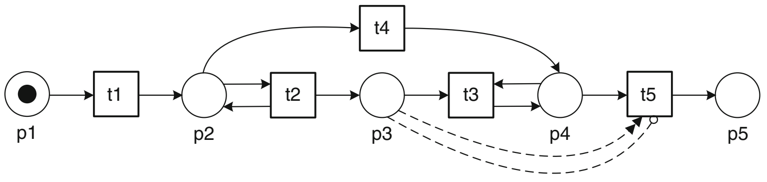

Fig. 17. N is the WF-net without the reset/inhibitor arc, N_R is the net with the reset arc, N_I is the net with the inhibitor arc, and N_{RI} is the net with both the reset arc and inhibitor arc

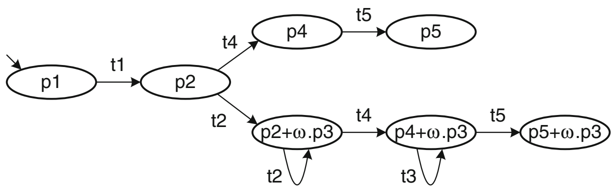

Fig. 18. The coverability graph of N

of the process models in SAP's reference model are not sound. However, this 5.6 percent is merely a lower bound. In [MNA07] an even larger set of more than 2000 process models from practice was analyzed. It could be shown that at least 10 percent of these models are not sound. These examples show that *even if soundness is undecidable, errors can be discovered*. Similarly, for many models it is still possible to guarantee soundness even if the general verification problem is undecidable.

As for many specific process models it is possible to make useful conclusions even if the general question is undecidable, we advocate a more down-to-earth approach. In this section, we discuss three analysis approaches in relation to the verification problems discussed earlier.

7.1. Coverability graph

The construction of a coverability graph is one of the standard approaches for analyzing classical Petri nets [Mur89]. If the state-space is finite, the coverability graph coincides with the reachability graph where nodes correspond to reachable markings and arcs correspond to state transitions. If the state-space is infinite, the coverability graph contains so-called ω markings indicating that the number of tokens on a particular place may grow unbounded. Figure 17 shows a WF-net (please ignore the dashed arcs for the moment) and the corresponding coverability graph is shown in Fig. 18.

Based on the coverability graph one can see that the WF-net without the dashed arcs, i.e., net N, is not sound (classical soundness). Node $p5 + \omega.p3$ in Fig. 18 indicates that it is possible to reach a state with one token in $p5$ and an arbitrarily large number of tokens in $p3$.[6] This proves that N is not sound (no proper completion). Moreover, it can be used to show that N is also not weak sound. For any $k \geq 2$ the coverability graph can be constructed and this will show that the net is also not k-sound. Therefore, it is possible to show that N is also not up-to-k sound or generalized sound. However, using the coverability graph in Fig. 18 it can be shown that N is lazy sound and easy sound. Using the construction illustrated in Fig. 15, a set of adapted coverability graphs can be used to prove relaxed soundness. In general, the coverability graph (or a set of coverability graphs) can be used to decide on any type of soundness except for generalized soundness. For generalized soundness one can use the approach described in [HSV04].

[6] To be more precise: for any $k \in \mathbb{N}$ there exists an $l \geq k$ such that state $[p5, p3^l]$ is reachable.

Let us consider again the WF-net shown in Fig. 17. N_R is the net with the dashed reset arc and without the dashed inhibitor arc. N_R is classical sound, weak sound, k-sound, up-to-k sound, generalized sound, relaxed sound, lazy sound, and easy sound. N_I is the net with the inhibitor arc and not the reset arc and N_{RI} is the net with both arcs. N_I and N_{RI} also satisfy all eight notions of soundness.

In the presence of reset or inhibitor arcs, the coverability graph cannot be used to come to a conclusive answer. If the state space is finite, the reachability graph can be constructed and seven notions of soundness can be checked (all except generalized soundness). Note however that boundedness is undecidable for nets with reset or inhibitor arcs [DJS99]. Nevertheless, the coverability graph of a WF-net where the reset and inhibitor arcs are ignored, provides a kind of "upper-bound" for the model with reset and inhibitor arcs. Let N_X be a WF-net such that after moving the reset and inhibitor arcs, N_X coincides with N in Fig. 17. Based on the coverability graph in Fig. 18, we can conclude that for such an N_X there can never be two tokens in $p5$ or a token in $p2$ and $p4$ at the same time. This illustrates that the coverability graph can be seen as an over-approximation of the true behavior.

7.2. Invariants

As shown in [Aal00, VBA01] structural techniques can be used to analyze WF-nets without reset and/or inhibitor arcs. In this setting, the absence of certain invariants points towards errors as explained below.

A *place invariant* is a weighted sum over the places that is invariant under each possible transition firing. In Fig. 17 the sum of tokens on the places $p1$, $p2$, $p4$, and $p5$ is constant independent of the initial marking, i.e., it is a structural property. The absence of particular place invariants hints at problems. For example, in WF-net N shown in Fig. 17 there is no place invariant adding a positive weight to all places. This typically suggests some structural anomaly. In this case, there is a positive invariant involving all places except $p3$ and there is no such invariant involving $p3$. This is caused by the fact that $p3$ is unbounded and this is indeed the primary reason why N is not classical/weak sound. Hence, the lack of such an invariant is a useful diagnostic.

A *transition invariant* assigns a weight to all transitions such that if each transition is able to fire the number of times indicated by the weight, the system is back in the initial state. Note that it is not guaranteed that a transition invariant is realizable, i.e., it is a structural property independent of the initial marking and there may be too few tokens to allow each of the transitions fire the designated number of times. When applying transition invariants one should consider the so-called short-circuited net mentioned earlier. The short-circuited net is the Petri net obtained by connecting o to i via a new transition t^* thus making the net cyclic. When considering classical soundness or relaxed soundness, there should be a semi-positive transition invariant for each transition, i.e., for each transition t it should be possible to find an invariant that assigns a positive weight to both t and t^*. If it is not possible to find such an invariant for t, the transition cannot contribute to any execution sequence leading from $[i]$ to $[o]$.

The above shows that *using invariants one can generate useful diagnostics* for WF-nets without reset and/or inhibitor arcs. It is easy to see that invariants are *less useful for nets with reset arcs* because these arcs destroy the nice linear algebraic properties that follow from the marking equation [Mur89]. However, *all the properties still hold for WF-nets with inhibitor arcs*. This is shown in detail in [VAH07]. Consider N_I in Fig. 17 (i.e., the WF-net with the inhibitor arc). If N_I is classical sound or relaxed sound, then for each t here has to be an invariant that assigns a positive weight to t and the short-circuiting transition t^*. This is indeed the case. In fact, the transition invariant that assigns weight 1 to all transitions including t^* is such an invariant. If such invariant(s) would not exist, the WF-net with the inhibitor arc could not be classical/relaxed sound. Also note that inhibitor arcs leave place invariants intact, i.e., the behavioral properties of place invariants obtained by ignoring inhibitor arcs still hold when inhibitor arcs are added.

In a practical setting, invariants can be used to discover lots of errors. For example, in [MMN+06, MVD+08] it is shown that many errors in the SAP reference model can be discovered using transition invariants.

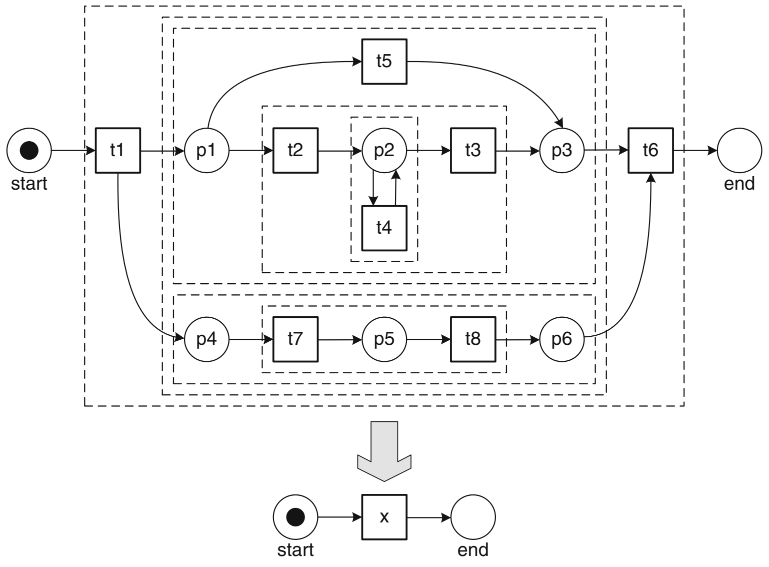

Fig. 19. A sound WF-net that can be reduced to the trivially correct WF-net

7.3. Reduction rules

The construction of a coverability graph can be considered as a "brute force" approach while the use of structural techniques such as invariants aims at a more efficient analysis. However, in many cases the "brute force" takes too long and structural techniques give inconclusive answers. Reductions rules provide a different approach and can be used in combination with other techniques. The goal of a reduction rule is to make the net smaller without changing specific properties such as soundness. There are basically two reasons for using reduction rules in the context of WF-nets. First of all, by making the WF-net smaller without changing its soundness properties, it becomes easier to apply other methods such as the construction of the coverability graph. Second, using reduction rules it is possible to provide better diagnostics, i.e., by reducing the trivially correct parts of a WF-net, the focus can be on the erroneous or suspicious constructs that remain.

To illustrate the use of reduction rules, we consider the sound WF-net shown in Fig. 19 (top one). Note that this net satisfies all eight forms of soundness. We can apply the liveness and boundedness preserving reduction rules described in [Ber87] and by doing so we obtain the net shown at the bottom of Fig. 19. The dashed lines illustrate the scopes of the various applications of the reduction rules. The reduced model is trivially sound, so the original model was also sound. We can apply the liveness and boundedness preserving reduction rules of [Ber87] to get this result, because a WF-net is classical sound if and only if the corresponding short-circuited net is live and bounded [Aal97, Aal98].

Figure 20 shows how reduction rules can be used to show that a WF-net is not sound and highlights the problematic parts. By applying the liveness and boundedness preserving reduction rules, we can reduce the top net into the small net at the bottom. The reduced net clearly shows the problem, i.e., a lack of synchronization. Note that the short-circuited counterparts of both WF-nets are unbounded and hence these WF-nets are not sound. Note that all the classical liveness and boundedness preserving reduction rules [Ber87] that do not depend on the initial marking also preserve the other notions of soundness. However, these rules do not necessarily apply to Petri nets with reset and/or inhibitor arcs.

In [VWAH10, WVA$^+$09] the liveness and boundedness preserving reduction rules are extended to reset and/or inhibitor arcs, i.e., new rules are given and existing rules are modified. This set of rules can be applied to WF-nets such as the one shown in Fig. 21. The top net is classical sound, but is a bit difficult to interpret. (Note that the net is also weak and relaxed sound, but not 2-sound.) Using the rules presented in [VWAH10, WVA$^+$09] the net can be reduced without changing things with respect to weak/classical soundness. In the resulting WF-net only half of the transitions remain.

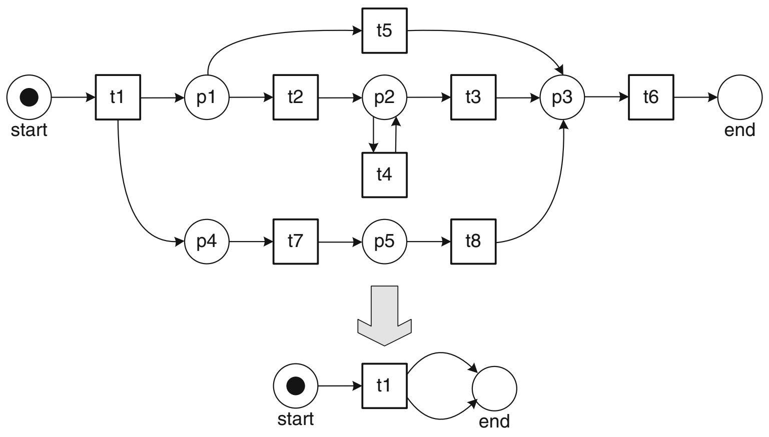

Fig. 20. A WF-net that is not sound and that cannot be reduced completely

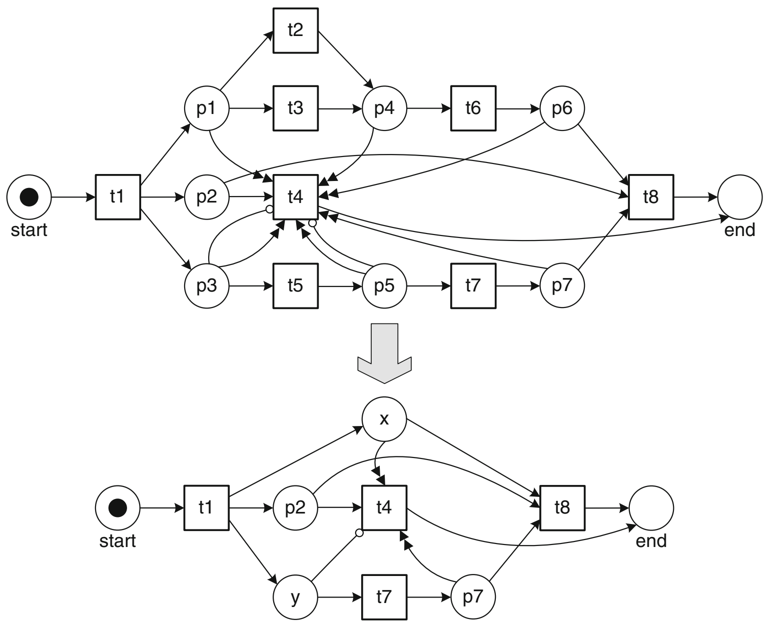

Fig. 21. Partial reduction of a WF-net with reset and inhibitor arcs

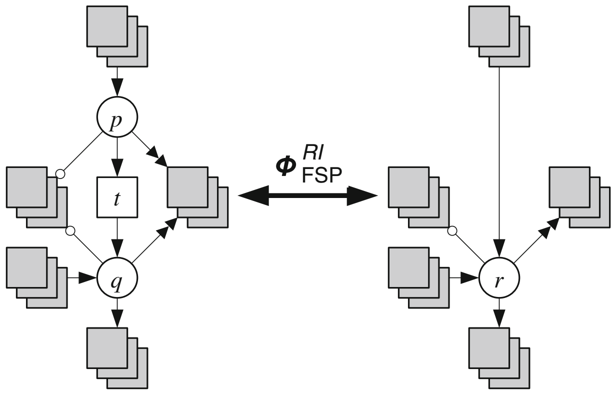

Fig. 22. One of the liveness and boundedness preserving reduction rules presented in [VWAH10]: Fusion of Series of Places (FSP)

A detailed discussion on the application of reduction rules to soundness analysis is beyond the scope of this paper. However, we informally show one of the reduction rules that preserves liveness and boundedness for nets with reset and/or inhibitor arcs. Using the *Fusion of Series Places* (FSP) rule shown in Fig. 22, we can reduce two places and one transition to one place. Thus, this rule effectively removes a transition and a place. The conditions are sketched informally in Fig. 22. Tokens which reside in the first place p can be considered to be "ghost tokens" for the second place q. Since there is no transition having p as input place other than t, tokens in p can/will always end up in q. If some transition needs to consume a ghost token in p, the intermediate transition t should fire first, replacing the ghost token in p by a tangible one in q. Since t should not be blocked and have other side-effects, t should not have any reset or inhibitor arcs. There can be reset and/or inhibitor arcs connected to p and q. However, these two places should be "identical" in terms of such arcs. Under such circumstances, p and q can be merged into r and t is no longer needed. The net before applying the FSP rule is live if and only if the net after applying the FSP rule is live. Moreover, the net before applying the FSP rule is bounded if and only if the net after applying the FSP rule is bounded. This is proven in [VWAH10]. Note that the FSP rule was applied repeatedly in Figs. 19, 20, and 21. For WF-nets without reset and inhibitor arcs, soundness corresponds to liveness and boundedness. This is not the case for WF-nets with reset and inhibitor arcs. However, it is easy to prove that the FSP rule also preserves soundness for arbitrary WF-nets with reset and inhibitor arcs.

The above examples illustrate that reduction rules can be used to simplify the problem and provide diagnostics. For particular languages variants of the above rules can be developed. See [DAV05] for a small set of reduction rules for EPCs. In [MNA07] a more extensive set of reduction rules for EPCs is given. Using these rules in combination with state space analysis, many errors were found in a sample of more than 2000 non-trivial EPCs from industry [MNA07]. Moreover, the reduction rules could also be used to highlight the errors.

In this section, we discussed three different approaches to tackle workflow verification problems. The goal was not to present a particular analysis technique but to provide an overview. This overview shows that for WF-nets without reset and/or inhibitor arcs, standard techniques such as the coverability graph and reduction rules can be used to decide soundness. Moreover, even though reset and/or inhibitor arcs are present and soundness is in principle undecidable, often it is still possible to apply these techniques and obtain valuable answers. Empirical studies show that in such cases many errors can be discovered even if soundness is undecidable in the general case.

8. Conclusion

Over the last decade many papers on workflow verification have appeared. Some of these papers use Petri nets as a representation language. Other papers use a wide variety of similar graph-based languages (e.g., EPCs, BPMN, or simple AND-XOR graphs). Moreover, also process-algebraic notations have been proposed to model and analyze workflows. However, independent of the representation used, there is always the basic notion of the creation of a process instance (case) and the successful completion of it. This naturally leads to various notions of *soundness*, i.e., reasoning about the correctness of a workflow model without any domain knowledge. Typical ingredients are absence of deadlocks and livelocks, proper termination, non-dead tasks, etc. Different soundness notions have been presented in the literature. Therefore, there is a need to compare the different notions, also in term of expressiveness. This paper provides a rigorous analysis of the different notions of soundness.

In order to investigate the various soundness notions, we use the so-called *extended WF-nets* as a starting point. These nets support the basic routing constructs but also allow for more advanced patterns through the so-called reset and inhibitor arcs. Notions such as cancellation, priority, etc. can be modeled using such arcs. In total 32 verification problems have been investigated for four classes of WF-nets and eight notions of soundness. For WF-nets without reset and/or inhibitor arcs all eight notions of soundness are decidable. However, as shown in this paper, most (if not all) notions of soundness become undecidable when reset and/or inhibitor arcs are used. Only for the combination of generalized soundness and WF-nets with reset arcs, did we not find a conclusive answer. This remains an open problem and a topic for future research.

As explained in Sect. 7, undecidability does not make things hopeless. Many errors can be discovered using techniques such as invariants and reduction rules. Applying such techniques to real-life models typically results in the discovery of many errors. This has been demonstrated in some recent case studies [MMN$^+$06, MVD$^+$08, FFJ$^+$09, MNA07, VVL07].

Given the importance of workflow technology, it is important to advance the state-of-the-art in workflow verification. Future work should aim to address the following questions.

- *How to deal with additional perspectives?* The focus of this paper has been on control-flow. However, also interaction, data, and resources play a role. Hence, it is necessary, but definitely not sufficient, to verify (control-flow) soundness. Unfortunately, naive extensions involving interaction, data, and resources will easily lead to verification questions that are undecidable. Therefore, the aim is to find the right abstractions to partially incorporate these other perspectives.
- *Why do workflow designers make errors?* As shown in [MNA07], people designing processes easily make errors. Moreover, initial studies suggest that it is possible to predict when workflow designers make errors. It is important to conduct more empirical studies and use the knowledge extracted in context sensitive editors. For example, the editor could ask questions about the workflow when it suspects that the user may have made an error.
- *When and why do people deviate?* Workflow models can be seen as normative. However, in most cases people can deviate. For example, workflow systems increasingly provide run-time flexibility and classical ERP systems do not enforce processes at all. Therefore, it becomes more and more important to analyze non-conformance afterwards. See [RA08] for techniques to check conformance by comparing event logs with normative models.

References

[Aal97] van der Aalst WMP (1997) Verification of workflow nets. In: Azéma P, Balbo G (eds) Application and theory of Petri nets 1997. Lecture notes in computer science, vol 1248. Springer-Verlag, Berlin, pp 407–426

[Aal98] van der Aalst WMP (1998) The application of Petri nets to workflow management. J Circ Syst Comput 8(1):21–66

[Aal00] van der Aalst WMP (2000) Workflow verification: finding control-flow errors using Petri-net-based techniques. In: van der Aalst WMP, Desel J, Oberweis A (eds) Business process management: models, techniques, and empirical studies. Lecture notes in computer science, vol 1806. Springer-Verlag, Berlin, pp 161–183

[AH04] van der Aalst WMP, van Hee KM (2004) Workflow management: models, methods, and systems. MIT press, Cambridge

[AH05] van der Aalst WMP, ter Hofstede AHM (2005) YAWL: yet another workflow language. Inf Syst 30(4):245–275

[AHH$^+$08] van der Aalst WMP, van Hee KM, ter Hofstede AHM, Sidorova N, Verbeek HMW, Voorhoeve M, Wynn MT (2008) Soundness of workflow nets: classification, decidability, and analysis. Computer Science report no. 08-13. Technische Universiteit Eindhoven, The Netherlands

[AHH$^+$09] van der Aalst WMP, van Hee KM, ter Hofstede AHM, Sidorova N, Verbeek HMW, Voorhoeve M, Wynn MT (2009) Soundness of workflow nets with reset arcs. In: Jensen K, Billington J, Koutny M (eds) Transactions on Petri nets and other models of Concurrency III. Lecture notes in computer science, vol 5800. Springer-Verlag, Berlin, pp 50–70

[AHKB03] van der Aalst WMP, ter Hofstede AHM, Kiepuszewski B, Barros AP (2003) Workflow patterns. Distrib Parallel Databases 14(1):5–51

[AHV02] van der Aalst WMP, Hirnschall A, Verbeek HMW (2002) An alternative way to analyze workflow graphs. In: Banks-Pidduck A, Mylopoulos J, Woo CC, Ozsu MT (eds) Proceedings of the 14th international conference on advanced information systems engineering (CAiSE'02). Lecture notes in computer science, vol 2348. Springer-Verlag, Berlin, pp 535–552

[ALM$^+$08] van der Aalst WMP, Lohmann N, Massuthe P, Stahl C, Wolf K (2008) From public views to private views: correctness-by-design for services. In: Dumas M, Heckel H (eds) Proceedings of the 4th international workshop on Web services and formal methods (WS-FM 2007). Lecture notes in computer science, vol 4937. Springer-Verlag, Berlin, pp 139–153

[AMSW09] van der Aalst WMP, Mooij AJ, Stahl C, Wolf K (2009) Service interaction: patterns, formalization, and analysis. In: Bernardo M, Padovani L, Zavattaro G (eds) Formal methods for Web services. Lecture notes in computer science, vol 5569. Springer-Verlag, Berlin, pp 42–88

[BB00] Basu A, Blanning RW (2000) A formal approach to workflow analysis. Inf Syst Res 11(1):17–36

[Ber87] Berthelot G (1987) Transformations and decompositions of nets. In: Brauer W, Reisig W, Rozenberg G (eds) Advances in Petri nets 1986. Part I: Petri nets, central models and their properties. Lecture notes in computer science, vol 254. pages 360–376. Springer-Verlag, Berlin

[BK02] Basu A, Kumar A (2002) Research commentary: workflow management issues in e-business. Inf Syst Res 13(1):1–14

[BP98] Barkaoui K, Petrucci L (1998) Structural analysis of workflow nets with shared resources. In: van der Aalst WMP, De Michelis G, Ellis CA (eds) Proceedings of workflow management: net-based concepts, models, techniques and tools (WFM'98), vol 98/7 of Computing science reports, Lisbon, Portugal. Eindhoven University of Technology, Eindhoven, pp 82–95

[BZ04] Bi HH, Zhao JL (2004) Applying propositional logic to workflow verification. Inf Technol Manag 5(3–4):293–318

[CW99] Chrzastowski-Wachtel P (1999) Testing undecidability of the reachability in Petri nets with the help of 10th Hilbert problem. In: Donatelli S, Kleijn J (eds) Application and theory of Petri nets. Lecture notes in computer science, vol 1639. Springer-Verlag, Berlin, pp 268–281

[DA04] Dehnert J, van der Aalst WMP (2004) Bridging the gap between business models and workflow specifications. Int J Coop Inf Syst 13(3):289–332

[DAC99] Dwyer MB, Avrunin GS, Corbett JC (1999) Patterns in property specifications for finite-state verification. In: ICSE '99: proceedings of the 21st international conference on software engineering, Los Alamitos, CA, USA. IEEE Computer Society Press, pp 411–420

[DAH05] Dumas M, van der Aalst WMP, ter Hofstede AHM (2005) Process-aware information systems: bridging people and software through process technology. Wiley, Hoboken, New Jersey

[DAV05] van Dongen BF, van der Aalst WMP, Verbeek HMW (2005) Verification of EPCs: using reduction rules and Petri nets. In: Pastor O, Falcão e Cunha J (eds) Proceedings of the 17th conference on advanced information systems engineering (CAiSE'05). Lecture notes in computer science, vol 3520. Springer-Verlag, Berlin, pp 372–386

[DE95] Desel J, Esparza J (1995) Free choice Petri nets. In: Cambridge tracts in theoretical computer science, vol 40. Cambridge University Press, Cambridge

[DFS98] Dufourd C, Finkel A, Schnoebelen Ph (1998) Reset nets between decidability and undecidability. In: Larsen K, Skyum S, Winskel G (eds) Proceedings of the 25th international colloquium on automata, languages and programming. Lecture notes in computer science, vol 1443. Springer-Verlag, Berlin, pp 103–115

[DJS99] Dufourd C, Jančar P, Schnoebelen Ph (1999) Boundedness of reset P/T nets. In: Wiedermann J, van Emde Boas P, Nielsen M (eds) Lectures on concurrency and Petri nets. Lecture notes in computer science, vol 1644. Springer-Verlag, Berlin, pp 301–310

[DR01] Dehnert J, Rittgen P (2001) Relaxed soundness of business processes. In: Dittrich KR, Geppert A, Norrie MC (eds) Proceedings of the 13th international conference on advanced information systems engineering (CAiSE'01). Lecture notes in computer science, vol 2068. Springer-Verlag, Berlin, pp 157–170

[EN94] Esparza J, Nielsen M (1994) Decidability issues for Petri nets: a survey. J Inf Process Cybern 30:143–160

[Esp98a] Esparza J (1998) Decidability and complexity of Petri net problems: an introduction. In: Reisig W, Rozenberg G (eds) Lectures on Petri nets I: basic models. Lecture notes in computer science, vol 1491. Springer-Verlag, Berlin, pp 374–428

[Esp98b] Esparza J (1998) Reachability in live and safe free-choice Petri nets is NP-complete. Theor Comput Sci 198(1–2):211–224

[FBS02] Fu X, Bultan T, Su J (2002) Formal verification of e-Services and workflows. In: Bussler C, Hull R, McIlraith S, Orlowska M, Pernici B, Yang J (eds) Web services, E-business, and the semantic web, CAiSE 2002 international workshop (WES 2002). Lecture notes in computer science, vol 2512. Springer-Verlag, Berlin, pp 188–202

[FBS04] Fu X, Bultan T, Su J (2004) Analysis of interacting BPEL Web services. In: International World Wide Web conference: proceedings of the 13th international conference on World Wide Web, New York. ACM Press, pp 621–630

[FFJ$^+$09] Fahland D, Favre C, Jobstmann B, Koehler J, Lohmann N, Voelzer H, Wolf K (2009) Instantaneous soundness checking of industrial business process models. In: Dayal U, Eder J, Koehler J, Reijers H, (eds) Business process management (BPM 2009). Lecture notes in computer science, vol 5701. Springer-Verlag, Berlin, pp 278–293

[FS01] Finkel A, Schnoebelen Ph (2001) Well-structured transition systems everywhere! Theor Comput Sci 256(1–2):63–92

[GHS95] Georgakopoulos D, Hornick M, Sheth A (1995) An overview of workflow management: from process modeling to workflow automation infrastructure. Distrib Parallel Databases 3(2):119–153

[GW96] van Glabbeek RJ, Weijland WP (1996) Branching time and abstraction in bisimulation semantics. J ACM 43(3):555–600

[HSS05] Hinz S, Schmidt K, Stahl C (2005) Transforming BPEL to Petri nets. In: van der Aalst WMP, Benatallah B, Casati F, Curbera F (eds) International conference on business process management (BPM 2005). Lecture notes in computer science, vol 3649. Springer-Verlag, Berlin, pp 220–235

[HSSV05] van Hee KM, Serebrenik A, Sidorova N, Voorhoeve M (2005) Soundness of resource-constrained workflow nets. In: Ciardo G, Darondeau P (eds) Applications and theory of Petri nets 2005. Lecture notes in computer science, vol 3536. Springer-Verlag, Berlin, pp 250–267

[HSV03] van Hee KM, Sidorova N, Voorhoeve M (2003) Soundness and separability of workflow nets in the stepwise refinement approach. In: van der Aalst WMP, Best E (eds) Application and theory of Petri nets 2003. Lecture notes in computer science, vol 2679. Springer-Verlag, Berlin, pp 335–354

[HSV04] van Hee KM, Sidorova N, Voorhoeve M (2004) Generalised soundness of workflow nets is decidable. In: Cortadella J, Reisig W (eds) Application and theory of Petri nets 2004. Lecture notes in computer science, vol 3099. Springer-Verlag, Berlin, pp 197–215

[JB96] Jablonski S, Bussler C (1996) Workflow management: modeling concepts, architecture, and implementation. International Thomson Computer Press, London

[Jen97] Jensen K (1997) Coloured Petri nets. Basic concepts, analysis methods and practical use, vol 1. EATCS monographs on theoretical computer science. Springer-Verlag, Berlin

[JKJ10] Juhas G, Kazlov I, Juhasova A (2010) Instance deadlock: a mystery behind frozen programs. In: Lilius J, Penczek W (eds) Applications and theory of Petri nets 2010. Lecture notes in computer science, vol 6128. Springer-Verlag, Berlin, pp 1–17

[KA99] Kindler E, van der Aalst WMP (1999) Liveness, fairness, and recurrence. Inf Process Lett 70(6):269–274

[KGMW00] Karamanolis C, Giannakopoulou D, Magee J, Wheater SM (2000) Model checking of workflow schemas. In: Proceedings of the fourth international enterprise distributed object computing conference (EDOC'00), Los Alamitos, CA, USA, 2000. IEEE Computer Society, pp 170–181

[Kin06] Kindler E (2006) On the semantics of EPCs: a framework for resolving the vicious circle. Data Knowl Eng 56(1):23–40

[KMR00] Kindler E, Martens A, Reisig W (2000) Inter-operability of workflow applications: local criteria for global soundness. In: van der Aalst WMP, Desel J, Oberweis A (eds) Business process management: models, techniques, and empirical studies. Lecture notes in computer science, vol 1806. Springer-Verlag, Berlin, pp 235–253

[KNS92] Keller G, Nüttgens M, Scheer AW (1992) Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken

[LMSW06] Lohmann N, Massuthe P, Stahl C, Weinberg D (2006) Analyzing interacting BPEL processes. In: Dustdar S, Fiadeiro JL, Sheth A (eds) International conference on business process management (BPM 2006). Lecture notes in computer science, vol 4102. Springer-Verlag, Berlin, pp 17–32

[LR99] Leymann F, Roller D (1999) Production workflow: concepts and techniques. Prentice-Hall PTR, Upper Saddle River, New Jersey

[LW10] Lohmann N, Weinberg D (2010) Wendy: a tool to synthesize partners for services. In: Lilius J, Penczek W (eds) Applications and theory of Petri nets 2010. Lecture notes in computer science, vol 6128. Springer-Verlag, Berlin, pp 279–307

[LZLC02] Lin H, Zhao Z, Li H, Chen Z (2002) A novel graph reduction algorithm to identify structural conflicts. In: Proceedings of the thirty-fourth annual Hawaii international conference on system science (HICSS-35). IEEE Computer Society Press

[Mar03] Martens A (2003) On Compatibility of Web services. Petri Net Newsl 65:12–20

[Mar05a] Martens A (2005) Analyzing Web service based business processes. In: Cerioli M (ed) Proceedings of the 8th international conference on fundamental approaches to software engineering (FASE 2005). Lecture notes in computer science, vol 3442. Springer-Verlag, Berlin, pp 19–33

[Mar05b] Martens A (2005) Consistency between executable and abstract processes. In: Proceedings of international IEEE conference on e-Technology, e-Commerce, and e-Services (EEE'05). IEEE Computer Society Press, pp 60–67

[MMN$^+$06] Mendling J, Moser M, Neumann G, Verbeek HMW, van Dongen BF, van der Aalst WMP (2006) Faulty EPCs in the SAP reference model. In: Dustdar S, Fiadeiro JL, Sheth A (eds) International conference on business process management (BPM 2006). Lecture notes in computer science, vol 4102. Springer-Verlag, Berlin, pp 451–457

[MNA07] Mendling J, Neumann G, van der Aalst WMP (2007) Understanding the occurrence of errors in process models based on metrics. In: Curbera F, Leymann F, Weske M (eds) Proceedings of the OTM conference on cooperative information systems (CoopIS 2007). Lecture notes in computer science, vol 4803. Springer-Verlag, Berlin, pp 113–130

[MRS05a] Massuthe P, Reisig W, Schmidt K (2005) An operating guideline approach to the SOA. In: Proceedings of the 2nd South-East European workshop on formal methods 2005 (SEEFM05), Ohrid, Republic of Macedonia

[MRS05b] Massuthe P, Reisig W, Schmidt K (2005) An operating guideline approach to the SOA. Ann Math Comput Teleinformatics 1(3):35–43

[Mue04] zur Muehlen M (2004) Workflow-based process controlling: foundation, design and application of workflow-driven process information systems. Logos, Berlin

[Mur89] Murata T (1989) Petri nets: properties, analysis and applications. Proc IEEE 77(4):541–580

[MVD$^+$08] Mendling J, Verbeek HMW, van Dongen BF, van der Aalst WMP, Neumann G (2008) Detection and prediction of errors in EPCs of the SAP reference model. Data Knowl Eng 64(1):312–329

[ODA$^+$09] Ouyang C, Dumas M, van der Aalst WMP, ter Hofstede AHM, Mendling J (2009) From business process models to process-oriented software systems. ACM Trans Softw Eng Methodol 19(1):1–37

[PW06a] Puhlmann F, Weske M (2006) Interaction soundness for service orchestrations. In: Dan A, Lamersdorf W (eds) Proceedings of service-oriented computing (ICSOC 2006). Lecture notes in computer science, vol 4294. Springer-Verlag, Berlin, pp 302–313

[PW06b] Puhlmann F, Weske M (2006) Investigations on soundness regarding lazy activities. In: Dustdar S, Fiadeiro JL, Sheth A (eds) International conference on business process management (BPM 2006). Lecture notes in computer science, vol 4102. Springer-Verlag, Berlin, pp 145–160

[PW09] Puhlmann F, Weske M (2009) A look around the corner: the pi-calculus. In: Jensen K, van der Aalst WMP (eds) Transactions on Petri nets and other models of concurrency II. Lecture notes in computer science, vol 5460. Springer-Verlag, Berlin, Florida, pp 64–78

[RA08] Rozinat A, van der Aalst WMP (2008) Conformance checking of processes based on monitoring real behavior. Inf Syst 33(1):64–95

[SO97] Sadiq W, Orlowska ME (1997) On correctness issues in conceptual modeling of workflows. In: Proceedings of the 5th European conference on information systems (ECIS '97), Cork, Ireland, pp 19–21

[SO00] Sadiq W, Orlowska ME (2000) Analyzing process models using graph reduction techniques. Inf Syst 25(2):117–134

[SW01] Salimifard K, Wright M (2001) Petri net-based modelling of workflow systems: an overview. Eur J Oper Res 134(3):664–676

[TAS09] Trcka N, van der Aalst WMP, Sidorova N (2009) Data-flow anti-patterns: discovering data-flow errors in workflows. In: van Eck P, Gordijn J, Wieringa R (eds) Advanced information systems engineering, proceedings of the 21st international conference on advanced information systems engineering (CAiSE'09). Lecture notes in computer science, vol 5565. Springer-Verlag, Berlin, pp 425–439

[Too04] van der Toorn R (2004) Component-based software design with Petri nets: an approach based on inheritance of behavior. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands

[VA05] Verbeek HMW, van der Aalst WMP (2005) Analyzing BPEL processes using Petri nets. In: Marinescu D (ed) Proceedings of the second international workshop on applications of Petri nets to coordination, workflow and business process management. Florida International University, Miami, Florida, pp 59–78

[VAH07] Verbeek HMW, van der Aalst WMP, ter Hofstede AHM (2007) Verifying workflows with cancellation regions and OR-joins: an approach based on relaxed soundness and invariants. Comput J 50(3):294–314

[VBA01] Verbeek HMW, Basten T, van der Aalst WMP (2001) Diagnosing workflow processes using Woflan. Comput J 44(4):246–279

[VVL07] Vanhatalo J, Völzer H, Leymann F (2007) Faster and more focused control-flow analysis for business process models through SESE decomposition. In: Krämer B, Lin K, Narasimhan P (eds) Proceedings of service-oriented computing (ICSOC 2007). Lecture notes in computer science, vol 4749. Springer-Verlag, Berlin, pp 43–55

[VWAH10] Verbeek HMW, Wynn MT, van der Aalst WMP, ter Hofstede AHM (2010) Reduction rules for reset/inhibitor nets. J Comput Syst Sci 76(2):125–143

[WAHE06] Wynn MT, van der Aalst WMP, ter Hofstede AHM, Edmond D (2006) Verifying workflows with cancellation regions and OR-joins: an approach based on reset nets and reachability analysis. In: Dustdar S, Fiadeiro JL, Sheth A (eds) International conference on business process management (BPM 2006). Lecture notes in computer science, vol 4102. Springer-Verlag, Berlin, pp 389–394

[WEAH05] Wynn MT, Edmond D, van der Aalst WMP, ter Hofstede AHM (2005) Achieving a general, formal and decidable approach to the OR-join in workflow using reset nets. In: Ciardo G, Darondeau P (eds) Applications and theory of Petri nets 2005. Lecture notes in computer science, vol 3536. Springer-Verlag, Berlin, pp 423–443

[Wes07] Weske M (2007) Business process management: concepts, languages, architectures. Springer-Verlag, Berlin

[WG07] Wong PYH, Gibbons J (2007) A process-algebraic approach to workflow specification and refinement. In: Lumpe M, Vanderperren W (eds) Software composition. Lecture notes in computer science, vol 4829. Springer-Verlag, Berlin, pp 51–65

[WG08] Wong PYH, Gibbons J (2008) A process semantics for BPMN. In: Liu S, Maibaum T, Arki K (eds) International conference on formal engineering methods (ICFEM 2008). Lecture notes in computer science, vol 5256. Springer-Verlag, Berlin, pp 27–31

[WG09] Wong PYH, Gibbons J (2009) Property specifications for workflow modelling. In: Proceedings of 7th international conference on integrated formal methods. Lecture notes in computer science, vol 5423. Springer-Verlag, Berlin

[Whi09] White SA et al (2009) Business process modeling notation specification (Version 1.2, OMG Final Adopted Specification)

[Wol09] Wolf K (2009) Does my service have partners? In: Jensen K, van der Aalst WMP (eds) Transactions on Petri nets and other models of concurrency II. Lecture notes in computer science, vol 5460. Springer-Verlag, Berlin, pp 152–171

[Wom06] Wombacher A (2006) Decentralized consistency checking in cross-organizational workflows. In: Proceedings of international conference on e-Technology, e-Commerce and e-Service (CEC/EEE 2006). IEEE Computer Society, pp 39–46

[WVA$^+$09] Wynn MT, Verbeek HMW, van der Aalst WMP, ter Hofstede AHM, Edmond D (2009) Soundness-preserving reduction rules for reset workflow nets. Inf Sci 179(6):769–790

Received 17 August 2009
Revised 24 June 2010
Accepted 6 July 2010 by J. C. P. Woodcock
Published online 3 August 2010