Soundness of Workflow Nets with Reset Arcs

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

[1] Department of Mathematics and Computer Science,
Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands.
`w.m.p.v.d.aalst@tue.nl`

[2] Business Process Management Group, Queensland University of Technology
P.O. Box 2434, Brisbane Qld 4001, Australia.

Abstract. Petri nets are often used to model and analyze workflows. Many workflow languages have been mapped onto Petri nets in order to provide formal semantics or to verify correctness properties. Typically, the so-called *Workflow nets* are used to model and analyze workflows and variants of the classical *soundness property* are used as a correctness notion. Since many workflow languages have *cancelation features*, a mapping to workflow nets is not always possible. Therefore, it is interesting to consider workflow nets with *reset arcs*. Unfortunately, soundness is undecidable for workflow nets with reset arcs. In this paper, we provide a proof and insights into the theoretical limits of workflow verification.

1 Introduction

Information systems have become "process-aware", i.e., they are driven by process models [11]. Often the goal is to automatically configure systems based on process models rather than coding the control-flow logic using some conventional programming language. Early examples of process-aware information systems were called WorkFlow Management (WFM) systems [4, 19, 27]. 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".

The flow-oriented nature of workflow processes makes the Petri net formalism a natural candidate for the modeling and analysis of workflows. This paper focuses on the so-called *workflow nets* (WF-nets) introduced in [1, 2]. A WF-net is a Petri net with a start place i and an end place o such that all nodes are on a path from i to o. A case, i.e., process instance, is initiated via the source place i and successfully completes by putting a token in the sink place o.

In the context of WF-nets a correctness criterion called *soundness* has been defined [1, 2]. A WF-net with source place i and sink place o is *sound* if and only if the following three requirements are satisfied: (1) *option to complete*: for each case starting in source place i it is always still possible to reach the state which just marks sink place o, (2) *proper completion*: if sink place o 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 [1, 2] 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 is live and bounded. In the last decade, the soundness property has become the standard correctness notion for workflow. This is illustrated by the fact that [2] is among the most cited papers both in the workflow/BPM community and Petri net community.

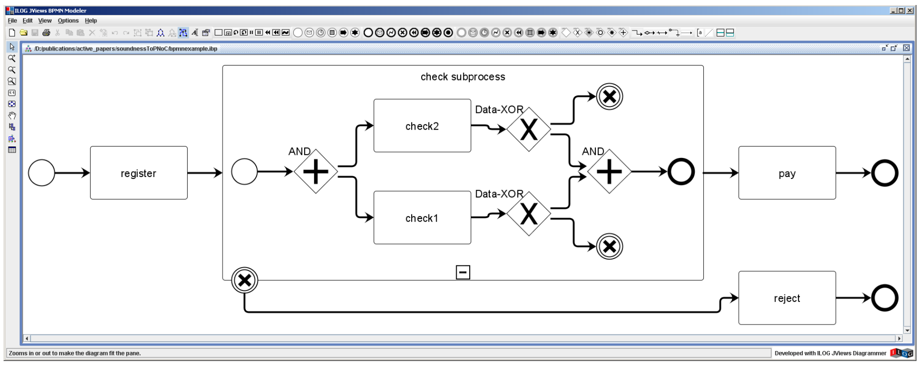
ILOG JViews BPMN Modeler
File
Edit
View
Options
Help
/D:/publications/active_papers/soundnessToPNoC/bpmnexample.ibp
check subprocess
register
AND
check2
check1
Data-XOR
Data-XOR
AND
pay
reject
Zooms in or out to make the diagram fit the pane.
Developed with ILOG JViews Diagrammer

Fig. 1. A BPMN diagram constructed using ILOG JViews. The subprocess is canceled if one of the two checks is negative.

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). However, in many cases a rather simple model is used (WF-nets or even less expressive) and practical features such as *cancelation* are missing. Many practical languages have a cancelation feature, e.g., Staffware has a withdraw construct, YAWL has a cancelation region, BPMN has cancel, compensate, and error events, etc. To illustrate this consider the BPMN diagram shown in Figure 1. The process describes the handing of some claim that requires two checks. The process starts with a *registration* step, followed by the parallel execution of *check1* and *check2*. The outcome of each of these checks may be negative of positive. If both are positive, activity *pay* follows and concludes the process. If one of the checks is negative, no further processing is needed and the process ends with activity *reject*. Figure 1 uses four BPMN gateways depicted using a diamond shape. The gateways with a "+" annotation have an AND-split/join behavior, while the gateways with a "**x**" annotation have an XOR-split/join behavior. Events are depicted by a circle. Events with a "+" annotation correspond to cancelation. Note that in Figure 1 a negative outcome triggers a cancelation event which triggers the cancelation of the whole subprocess *check subprocess*. This means that the first negative results cancels any activities scheduled or taking place inside this subprocess.

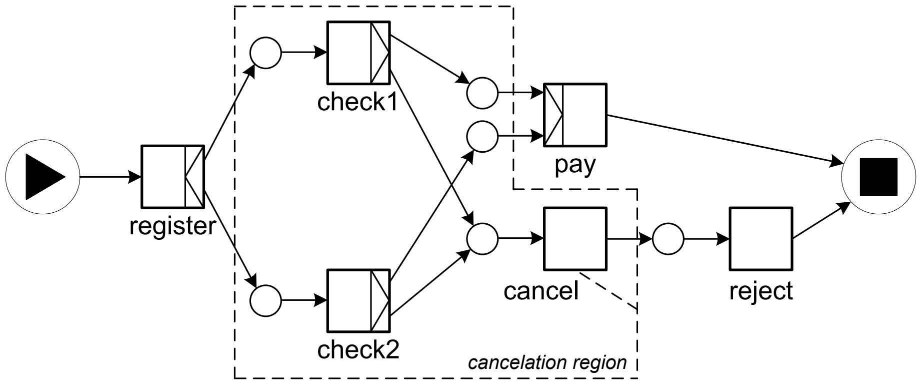

Fig. 2. A YAWL model using cancelation.

Figure 2 shows the same example, but now modeled using YAWL. By comparing Figure 1 and Figure 2 it should be obvious what the semantics of the various YAWL notations are. Task *register* is an AND-split and task *pay* is an AND-join. The two check tasks are XOR-splits. The two input places of *pay* correspond to positive outcomes, while the input place of *cancel* holds a token for every negative outcome. YAWL supports the concept of a "cancelation region", i.e., anything is removed from this region for a particular instance. In Figure 2 the cancelation region is depicted using dashed lines. This region is activated by the execution of task *cancel*, i.e., after executing *cancel* there is only a token in the input place of task *reject*.

BPMN and YAWL are two of many process modeling languages that support cancelation. Table 1 provides some more examples. This table illustrates that many systems and languages support cancelation functionality.

Since we are interested in verification of real-life problems, we need to support cancelation. Therefore, it is interesting to investigate the notion of soundness in the context of WF-nets with *reset arcs* [9, 10, 14]. A reset arc connects a place to a transition. For the enabling of this transition the reset arc plays no role. However, whenever this transition fires, then place is emptied. Clearly, this concept can be used to model various cancelation concepts encountered in modern workflow languages. To illustrate this consider the reset net shown in Figure 3. Note that the two XOR-splits have both been replaced by a small network of transitions. For example, *check1* is followed by two transitions (*OK* and *NOK*) modeling the different outcomes of this check. Moreover, the cancelation region has been replaced by seven reset arcs. The double-headed arcs in Figure 3 correspond

Table 1. Examples of languages supporting cancelation (see also [28]).

BPMN	Cancelation is supported by adding some intermediate event trigger attached to the boundary of the activity to be canceled.
YAWL	Cancelation is supported by the cancelation region which "empties" a selected part of the process.
Staffware	Cancelation is supported using the so-called "withdraw construct", i.e., a connection entering the top of a workflow step.
UML ADs	Cancelation is supported by incorporating the activity in an interruptible region triggered either by a signal or execution of another activity.
SAP Workflow	Cancelation is supported through the use of the "process control" step that can be used to "logically delete" activities by specifying the node number of the corresponding step.
FileNet	Cancelation is supported via a so-called "Terminate Branch" step.
BPEL	Cancelation is supported by fault and compensation handlers.
XPDL	Supported via an error type trigger attached to the boundary of the activity to be canceled.

to reset arcs. These arcs empty all places where tokens may remain after firing *cancel* for the first time. It is easy to see that Figure 3 has indeed the behavior described earlier using BPMN and YAWL.[3]

Note that it is far from trivial to express the desired behavior without reset arcs. Note that in Figure 3, transition *cancel* is the only transition having reset arcs. To remove these reset arcs, we would need to consider all possible markings before this point. In this case, there are (only) 7 possible markings when *cancel* fires, i.e., *cancel* would need to be replaced by 7 transitions. In general there is an exponential number of possible states. If there are n checks in the example (rather than 2), then $4^n - 3^n$ transitions are needed to replace *cancel* and its reset arcs (e.g., for 10 checks, 1048576-59049=989527 transitions are needed). This illustrates the relevance of reset arcs from a modeling point of view. Therefore, it is interesting to investigate the verification of WF-nets with reset arcs.

This paper will prove that *soundness is undecidable for reset WF-nets.* This result is not trivial since other properties such as e.g. coverability are decidable for reset nets. Moreover, as we will show, there is not a simple mapping between soundness and reachability which is known to be undecidable for reset net [9, 10, 14].

The remainder of this paper is organized as follows. First, we briefly present an overview of related work (Section 2). Then, Section 3 presents some of the preliminaries (mathematical notations and Petri net basics). Section 4 presents the basic notion of reset WF-nets. In Section 5 the classical notion of soundness is introduced. Section 6 presents the main result: undecidability of soundness for

[3] Note that here we assume activities to be atomic. It is also possible to describe the more refined behavior using reset nets. However, this would only complicate the presentation.

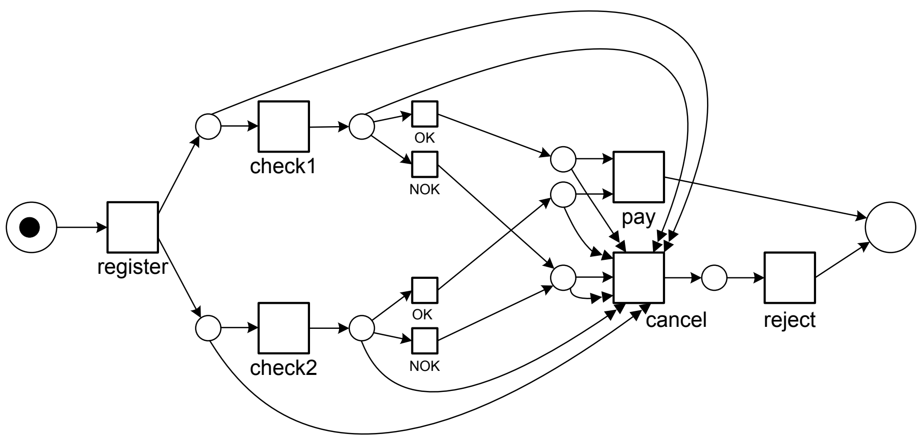

Fig. 3. A WF-net with reset arcs to model cancelation.

reset WF-nets. Moreover, we will show that soundness is also undecidable for weaker notions such as relaxed soundness [6, 7]. Section 7 concludes the paper.

2 Related Work

Since the mid nineties, many researchers have been working on workflow verification techniques. 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.

The use of Petri nets in workflow verification has been studied extensively. In [1, 2] the foundational notions of WF-nets and soundness are introduced. In [15, 16] two alterative 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 [20] the notion of weak soundness is proposed. This notion allows for dead transitions. The notion of relaxed soundness is introduced in [6, 7]. This notion allows for potential deadlocks and livelocks, however, for each transition there should be at least one proper execution. Lazy soundness [22] 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 [24]. More details on these notions proposed in the literature are given in Section 5.

Most soundness notions (except generalized soundness [15, 16]) 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. [3] describes how structural properties of a workflow net can be used to detect the soundness property. [25] 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 into Petri nets with inhibitor arcs. In [29] it is shown that the backward reachability graph can be used to determine the enabling of OR-joins in the context of cancelation. 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 [9, 10, 14]. Moreover, in [26] it is shown that reduction rules can be applied to reset nets (and even to inhibitor nets) to speed-up analysis and improve diagnostics. For decidability results for ordinary Petri nets we refer to [12, 13].

3 Preliminaries

This section introduces some of the basic mathematical and Petri-net related concepts used in the remainder of this paper.

3.1 Multi-sets, Sequences, and Matrices

Let A be a set. $\mathbb{B}(A) = A \rightarrow \mathbb{N}$ is the set of multi-sets (bags) over A, i.e., $X \in \mathbb{B}(A)$ is a multi-set where for each $a \in A$: $X(a)$ denotes the number of times a is included in the multi-set. The sum of two multi-sets $(X + Y)$, the difference $(X - Y)$, the presence of an element in a multi-set $(x \in X)$, and the notion of sub-multi-set $(X \leq Y)$ are defined in a straightforward way and they can handle a mixture of sets and multi-sets. $|X| = \sum_{a \in A} X(a)$ is the size of the multi-set. $\pi_{A'}(X)$ is the projection of X onto $A' \subseteq A$, i.e., $(\pi_{A'}(X))(a) = X(a)$ if $a \in A'$ and $(\pi_{A'}(X))(a) = 0$ if $a \notin A'$.

To represent a concrete multi-set we use square brackets, e.g., $[a, a, b, a, b, c]$, $[a^3, b^2, c]$, and $3[a] + 2[b] + [c]$ all refer to the same multi-set with six elements: 3 a's, 2 b's, and one c. $[\,]$ refers to the empty bag, i.e., $|[\,]| = 0$.

For a given set A, A^* is the set of all finite sequences over A (including the empty sequence $\langle\rangle$). A finite sequence over A of length n is a mapping $\sigma \in \{1, \ldots, n\} \rightarrow A$. Such a sequence is represented by a string, i.e., $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$ where $a_i = \sigma(i)$ for $1 \leq i \leq n$.

For a relation R on A, i.e., $R \subseteq A \times A$, we define R^* as the reflexive transitive closure of R.

3.2 Reset Petri nets

This subsection briefly introduces some basic *Petri net* terminology [8, 17, 23] and notations used in the remainder of this paper. Our starting point is a Petri net with reset arcs and arc weights. Such a Petri net is called a *reset net.*

Definition 1 (Reset net). *A reset net is a tuple (P, T, F, W, R), where:*

- *(P, T, F) is a classical Petri net with a finite set of places P, a finite set of transitions T, and a flow relation $F \subseteq (P \times T) \cup (T \times P)$,*
- *$W \in F \to I\!N \setminus \{0\}$ is an (arc) weight function, and*
- *$R \in T \to 2^P$ is a function defining reset arcs.*

A reset net extends the classical Petri net with arc weights and reset arcs. The arc weights specify the number of tokens to be consumed or produced and the reset arcs are used to remove all tokens from the reset places independent of the number of tokens. To illustrate these concepts we use Figure 4. This figure shows a reset net with seven places and six transitions. The arc from $t1$ to $p3$ has weight 6, i.e., $W(t1, p3) = 6$. Moreover, $W(p5, t5) = 6$, $W(p3, t4) = 2$, and $W(t4, p5) = 2$. All other arcs have weight 1, e.g., $W(p1, t1) = 1$. Transition tr has four reset arcs, i.e., $R(tr) = \{p2, p3, p4, p5\}$, and $R(t) = \emptyset$ for all other transitions t.

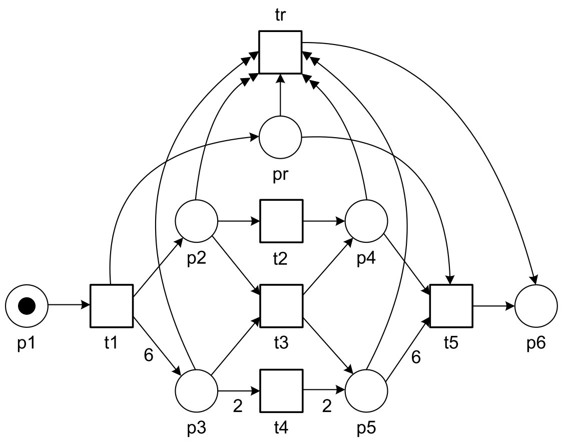

Fig. 4. A reset net. Transition tr is enabled if pr is marked and removes all tokens from $p2$, $p3$, $p4$, $p5$.

Because of the arc weights the classical preset and postset operators return bags rather than sets: $\bullet a = [x^{W(x,y)} \mid (x, y) \in F \ \wedge \ a = y]$ and $a\bullet = [y^{W(x,y)} \mid (x, y) \in F \ \wedge \ a = x]$. For example, $\bullet t5 = [p4, p5^6, pr]$ is the bag of input places of $t5$ and $t1\bullet = [p2, p3^6, pr]$ is the bag of output places of $t1$.

Now we can formalize the notions of enabling and firing.

Definition 2 (Firing rule). *Let $N = (P, T, F, W, R)$ be a reset net and $M \in I\!B(P)$ be a marking.*

- *A transition $t \in T$ is enabled at M, denoted by $(N, M)[t\rangle$, if and only if, $M \geq \bullet t$.*

– *An enabled transition t can fire while changing the state to M', denoted by $(N, M)[t\rangle(N, M')$, if and only if $M' = \pi_{P\setminus R(t)}(M - \bullet t) + t\bullet$.*

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. Note that $\pi_{P\setminus R(t)}$ 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.

In Figure 4, 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 $p5$.

$(N, M)[t\rangle(N, M')$ 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: $R(N, M) = \{M' \in \mathbb{B}(P) \mid \exists_{\sigma\in T^*}(N, M)[\sigma\rangle(N, M')\}$. Note that by definition $M \in R(N, M)$ because the initial marking M is trivially reachable via the empty sequence ($n = 0$).

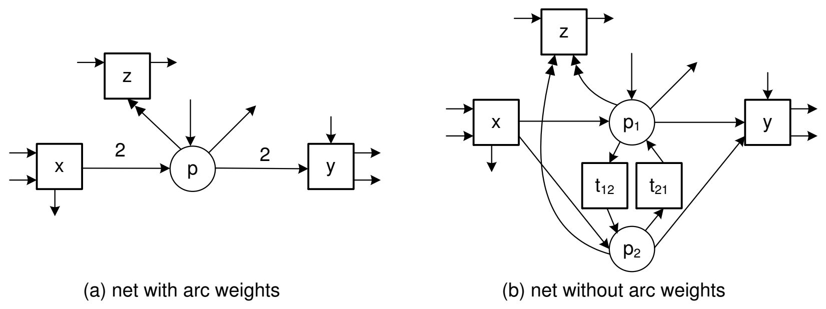

Fig. 5. Construction illustrating that it is possible to transform any reset net with arc weights into an equivalent Petri net without arc weights.

We would like to emphasize that any reset net with arc weights can be transformed into a reset net without arc weights, i.e., all arcs have weight 1. Therefore, in proofs we can assume arc weights of 1. Figure 5 illustrates how a Petri net with arc weights of 2 can be transformed into a Petri net without arc weights. If k is the maximum arc weight, the construction illustrated by Figure 5 requires the splitting of place p into k places $(p_1, \ldots, p_k)$. See [5] for details.

4 Reset 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 including a clear *start state* and *end state*. These basic assumptions lead to the notion of a *WorkFlow net* (WF-net) [1, 2]. In the introduction, we already informally introduced the notion of WF-nets and now it is time to formalize this notion in the presence of reset arcs.

Definition 3 (RWF-net). *A reset net $N = (P, T, F, W, R)$ is a Reset WorkFlow net (RWF-net) if and only if*

- *There is a single source place i, i.e., $\{p \in P \mid \bullet p = [\,]\} = \{i\}$.*
- *There is a single sink place o, i.e., $\{p \in P \mid p\bullet = [\,]\} = \{o\}$.*
- *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 transitive closure of F).*
- *There is no reset arc connected to the sink place, i.e., $\forall_{t \in T}\ o \notin R(t)$.*

Figure 4 shows a RWF-net. Also the example used in the introduction (Figure 3) is a RWF-net. 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).

5 Soundness

Based on the notion of RWF-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. However, without domain knowledge one can only 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* [1, 2].

Definition 4 (Classical soundness [1, 2]). *Let $N = (P, T, F, W, R)$ be a RWF-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: $\forall_{t \in T}\ \exists_{M \in R(N,[i])}\ (N, M)[t\rangle$.*

The RWF-nets depicted in figures 3 and 4 are sound.

The first requirement in Definition 4 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" [18], 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 [18]. 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 $[i]$.

By carefully looking at Definition 4 one can see that the second requirement is implied by the first one. Hence we can ignore the second requirement in Definition 4. The reason that we include it anyway is because it represents an intuitive behavioral requirement.

As pointed out in [1, 2], classical soundness of a WF-net without reset arcs 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. After the initial paper on soundness of WF-nets [1, 2] many other papers followed. Some extend the results while others explore alternative notions of soundness. These notions strengthen or weaken some of the requirements mentioned in Definition 4. Some examples are: k-soundness [15, 16], weak soundness [20], up-to-k-soundness [24], generalized soundness [15, 16], relaxed soundness [6, 7], lazy soundness [22], and easy soundness [24].

A detailed discussion of these soundness notions is beyond the scope of this paper, see [5] for a complete overview. Nevertheless, we would like to define *relaxed soundness* as an example of an alternative soundness notion.

Definition 5 (Relaxed soundness [6, 7]). *Let N be a RWF-net. N is relaxed sound if and only if for each transition $t \in T$:*
$\exists_{M,M' \in R(N,[i])}\ (N, M)[t\rangle(N, M') \ \wedge\ [o] \in R(N, M')$.

Classical soundness considers 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, i.e., for each transition there has to be at least one execution from the initial state to the desired final state that executes this transition.

6 Decidability

In this section we explore the decidability of soundness in the presence of reset arcs. First, we show that classical soundness is undecidable, then we show that relaxed soundness is also undecidable for RWF-nets.

6.1 Classical soundness is undecidable for RWF-nets

In this subsection, we explore the decidability of soundness for RWF-nets. If a WF-net has no reset arcs, soundness is decidable. Such a WF-net $N = (P, T, F)$ (without reset arcs) is sound if and only if the short-circuited net $(\overline{N}, [i])$ with $\overline{N} = (P, T \cup \{t^*\}, F \cup \{(o, t^*), (t^*, i)\})$ and $t^* \not\in T$ 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 [1, 2].

Unfortunately, soundness is not decidable for RWF-nets with reset arcs as is shown by the following theorem.

Theorem 1 (Undecidability of soundness). *Soundness is undecidable for RWF-nets with reset arcs.*

Proof. Let (N, M_I) be an arbitrary marked reset net. In the general case it is known that reachability is undecidable for reset nets [9, 10]. Without loss of generality we can assume that N is connected and that every transition has input and output places, since any reset net can be translated into a behaviorally equivalent net that has these properties. Moreover, since coverability is decidable for reset nets [9, 14], we can assume that all dead transitions have been removed. (Because we can check whether $\bullet t$ is coverable from the initial marking, we can test whether transition t is dead for any $t \in T$.) Hence we may assume that (N, M_I) is connected, every transition has input and output places, and there are no dead transitions.

To show that soundness is undecidable, we construct a new net $(N', [i])$ which embeds (N, M_I) such that N' is sound if and only if some marking M_X is *NOT* reachable from (N, M_I). By doing so, we show that reachability in an arbitrary reset net can be analyzed through soundness, making soundness also undecidable.

The construction is shown in Figure 6. However, to explain this we first need to introduce some notation. P is the set of places in N and T is the set of transitions in N. Assume $\{i, o, u, s, v, w\} \cap P = \emptyset$ and $(\{a, b, c, z\} \cup \{z_p \mid p \in P\}) \cap T = \emptyset$. These are the "fresh" identifiers corresponding to the places and transitions added to N to form N'. $I \subseteq P$ are all the places that are initially marked in (N, M_I) and $X \subseteq P$ are the places that are marked in (N, M_X). As Figure 6 shows, transition c initializes the places in I, i.e., for $p \in I$: $W(c, p) = M_I(p)$.[4] Similarly, transition b can fire and consume all tokens from X if marking M_X is reached, i.e., for $p \in X$: $W(p, b) = M_X(p)$, and transition a marks the places in X appropriately, i.e., for $p \in X$: $W(a, p) = M_X(p)$. The transitions z and z_p ($p \in P$) have reset arcs from all places in N' except the new sink place o. Any transition in the original net has a bidirectional arc with s, i.e., a self-loop. All other connections are as shown in Figure 6.

The constructed net $(N', [i])$ has the following behavior. First a fires, marking u, v and the places in X. No transition $t \in T$ can fire because s is still empty

[4] Note that we are assuming weighted arcs here. However, as shown before these can be removed using the construction in Figure 5.

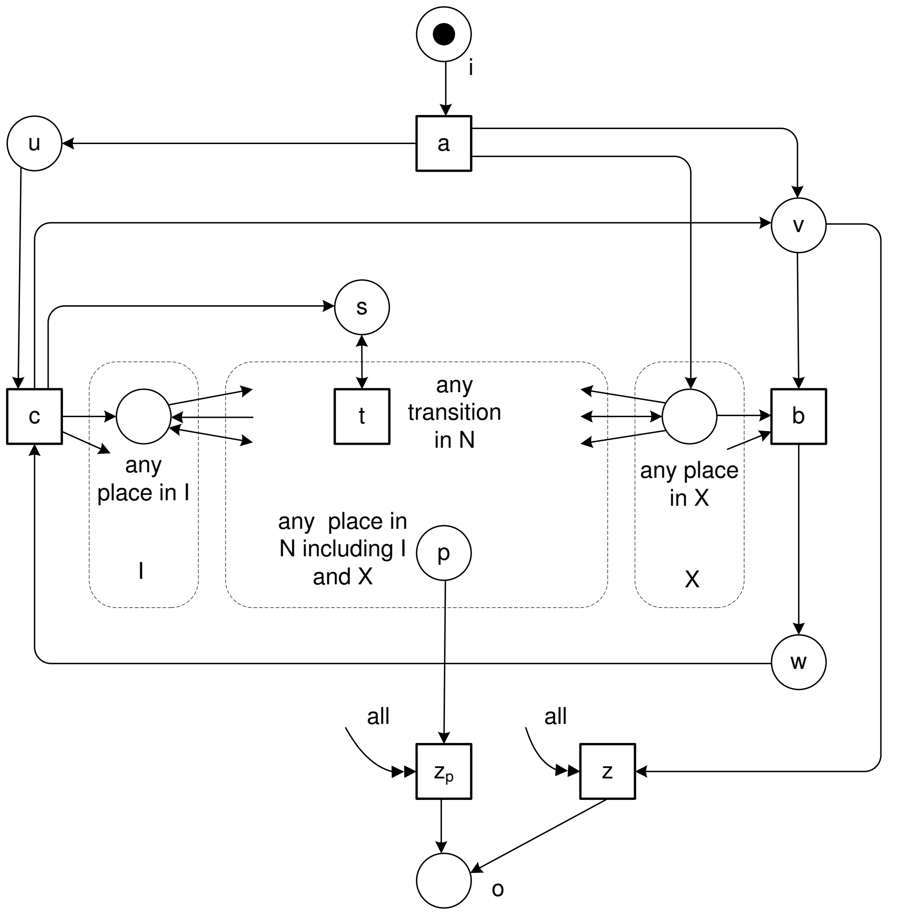

Fig. 6. Construction showing that soundness is undecidable for WF-nets with reset arcs. The original net comprises the three dashed areas: I is the set of places of N initially marked, X is the set of places that are marked in M_X, and all other nodes of N are shown in the dashed area in the middle. Note that I and X may overlap.

and c is also blocked because w is empty. The only two transitions that can fire are b and z. If z occurs, the net ends in marking $[o]$. If b fires, it will be followed by c. The firing of c brings the net into marking $M_I + [s, v]$. Note that in marking $M_I + [s, v]$ the original transitions are not constrained in any way and the embedded subnet can evolve as in (N, M_I) until one of the newly added transitions fires. Transitions $\{z_p \mid p \in P\}$ can fire as long as there is at least one token in a place in P and z can fire as long as there is a token in v. The firing of such a transition always leads to $[o]$, i.e., firing a transition in $\{z\} \cup \{z_p \mid p \in P\}$ always leads to the proper end state. Transition b can fire as soon as the embedded subnet has a marking which covers M_X.

It is obvious that net N' shown in Figure 6 is a WF-net, i.e., there is one source place i, one sink place o, all nodes are on a path from i to o, and there is no reset on o.

Now we can show that N' is sound if and only if the specified marking M_X is *NOT* reachable from (N, M_I):

- Assume marking M_X is reachable from (N, M_I). This implies that from $(N', [i])$ the marking $M_X + [s, v]$ is reachable. Hence b can fire for the second time resulting in a state $[s, w]$. In this state all transitions in T are blocked because transitions have input places and all input places in P are empty. Also all added transitions are dead in $[s, w]$. Hence a deadlock state $[s, w]$ is reachable from $(N', [i])$ implying that N' is not sound.
- Assume marking M_X is not reachable from (N, M_I) and M_X is also not coverable. This implies that b cannot fire for the second time. Hence, there always remain tokens in some place of P after initialization and it is always possible to terminate in state $[o]$ by firing one of the "z transitions". Moreover, none of the transitions is dead in $(N', [i])$ because $\{a, b, c, z\} \cup \{z_p \mid p \in P\}$ can fire and the transitions in T are not dead in (N, M_I) (because of the initial cleaning). Therefore, N' is indeed sound.
- Assume marking M_X is not reachable from (N, M_I) but M_X is coverable. This implies that in the embedded subnet it is only possible to reach states M' that are not covering M_X or that are bigger than M_X, i.e., $M' \geq M_X$ implies $M' \neq M_X$. For states smaller than M_X we have shown that soundness is not jeopardized. For states bigger than M_X, b can fire. However, if b fires, tokens remain in P and b cannot fire anymore. Hence, at least one transition in $\{z_p \mid p \in P\}$ is enabled at any time because one of the places in P is marked. As a result, it is always possible to terminate in state $[o]$ and N' is indeed sound.

Hence, if soundness is decidable for reset nets, then reachability is also decidable. This leads to a contradiction. Hence soundness is not decidable. □

Theorem 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.

6.2 Relaxed soundness is undecidable for RWF-nets

Relaxed soundness differs fundamentally from notions such as classical soundness, because it allows for deadlocks, etc. as long as there is a "good execution" possible for each transition. Like classical soundness, relaxed soundness is decidable for WF-nets without reset arcs. Unfortunately, relaxed soundness is also undecidable for RWF-nets.

Theorem 2 (Undecidability of relaxed soundness). *Relaxed soundness is undecidable for RWF-nets with reset arcs.*

Proof. Let (N, M_I) be an arbitrary marked reset net. Without loss of generality we can assume that N is connected and that every transition has input and output places. Any net can be translated into a behaviorally equivalent net that has these properties.

To show that relaxed soundness is undecidable, we construct a new net $(N', [i])$ which embeds (N, M_I) such that N' is relaxed sound if and only if some specified 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 relaxed soundness, making relaxed soundness undecidable because reachability is undecidable for reset nets [9, 10].

Note that here we choose a different strategy than in Theorem 1 where soundness corresponds to the *non*-reachability of a given marking M_X. Here, we make a construction such that relaxed soundness of N' corresponds to the reachability of M_X in (N, M_I).

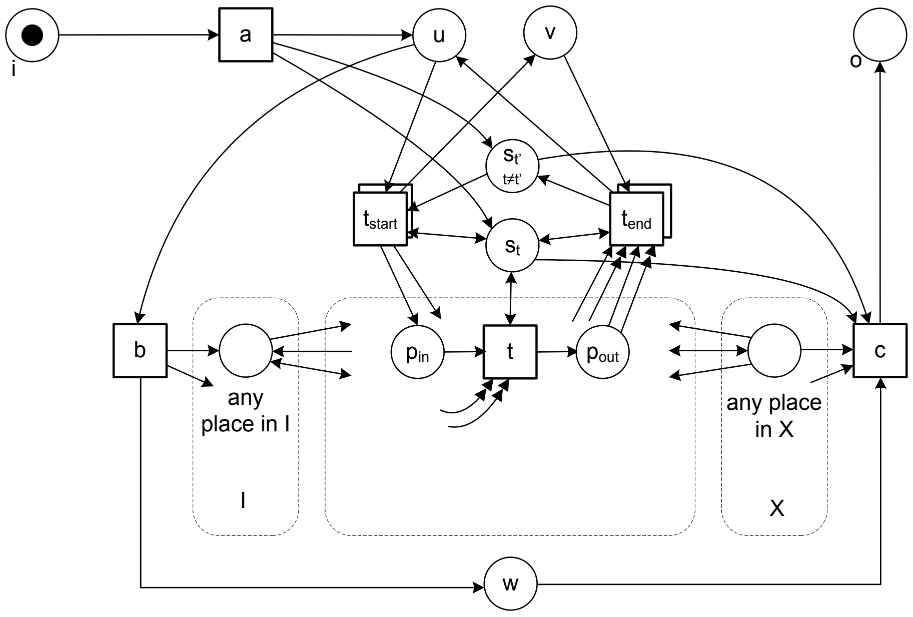

Fig. 7. Construction showing that reachability can be expressed in terms of relaxed soundness for WF-nets with reset arcs. (Note that I and X may overlap.)

Figure 7 shows the basic idea underlying the construction of N' from N. P is the set of places in N and T is the set of transitions in N. $I \subseteq P$ is the set of places marked in M_I and $X \subseteq P$ is the set of places marked in M_X. Although not shown in Figure 7, I and X may overlap. Let $T_{start} = \{t_{start} \mid t \in T\}$ and $T_{end} = \{t_{end} \mid t \in T\}$ be new transitions and let $S = \{s_t \mid t \in T\}$ be new places, i.e., for each $t \in T$ we add a self-loop place s_t and transitions t_{start} and t_{end}. Assume $(\{i, o, u, v, w\} \cup S) \cap P = \emptyset$ and $(\{a, b, c\} \cup T_{start} \cup T_{end}) \cap T = \emptyset$. For any t: $\bullet t_{start} = [u] + S$, $t_{start}\bullet = (\bullet t) + [s_t, v]$, $\bullet t_{end} = (t\bullet) + [s_t, v]$, and $t_{end}\bullet = [u] + S$. Also note the reset arcs of t_{end} and that $s \in \bullet t \cap t\bullet$. As Figure 7 shows, transition b initializes the places in I, i.e., for $p \in I$: $W(b, p) = M_I(p)$. Similarly, transition c consumes all tokens from X if marking M_X is reached, i.e., for $p \in X$: $W(p, c) = M_X(p)$.

To better understand the structure of N' note that there are the following place invariants: $i+u+v+w+o$ and $k.i+\sum_{t \in T} s_t+(k-1).v+k.o$ where $k = |T|$. The first invariant indicates that there will always be one token in exactly one of the places i, u, v, w, and o. The second invariant shows that there is a token in i (weight k), or there is a token in o (weight k), or there are tokens in $S \cup \{v\}$. In the latter case, there may be one token in v with weight $k-1$ and one token in one of the places in S with weight 1. So the sum of these two tokens is also k. Note that t_{start} consumes k tokens with weight one from S, returns one token to place $s_t \in S$, and puts a token with weight $k-1$ in place v. Transition t_{end} consumes one token from place $s_t \in S$ and one token with weight $k-1$ for place v, and produces k tokens with weight one for S. It is easy to show that these are indeed invariants because the reset arcs only affect the places in P and not any of the newly added places.

Initially a fires thus marking u and all places in S. In $[u] + S$, any of the T_{start} transitions can fire. Say t_{start} fires. In the resulting state $((\bullet t) + [s_t, v])$, t is the only transition in T that can fire. Note that all other transitions in T are blocked because the corresponding places in $S \setminus \{s_t\}$ are not marked. If $t\bullet \subseteq \bullet t$, then t does not have to fire and t_{end} may fire directly. However, t can fire. If $\bullet t \subseteq t\bullet$, then t may even fire multiple times. However, after firing one of more times t, t_{end} can fire and remove all tokens from $t\bullet$ using reset arcs if needed. Note that the reset arcs in the original net do not play a role here because transition t removes the tokens in $\bullet t$ and nothing more. In any case, the sequence $\langle t_{start}, t, t_{end} \rangle$ can be executed and results again in marking $[u] + S$. Hence this could be repeated for all $t \in T$, still resulting in marking $[u] + S$. In marking $[u] + S$ also b can fire resulting in marking $M_I + S + [w]$. Hence is it possible to move from marking $[i]$ to marking $M_I + S + [w]$ by firing $\sigma_b = \langle a, \ldots, t_{start}, t, t_{end}, \ldots, b \rangle$, i.e., $(N', [i])[\sigma_b\rangle(N', M_I+S+[w])$. Note that σ_b is such that it contains all transitions except c. After executing σ_b, the transitions in T can fire like in (N, M_I), i.e., not constrained by the added constructs, until c occurs. Suppose that c occurs, then all tokens in S are removed thus blocking all transitions in T. After firing c a token is put into o and no transition can fire anymore.

Now we can show that N' is relaxed sound if and only if the specified marking M_X is reachable in (N, M_I):

- Assume marking M_X is reachable from (N, M_I). There exists a firing sequence σ_N such that $(N, M_I)[\sigma_N\rangle(N, M_X)$. This sequence is also enabled in the state after executing σ_b: $(N', M_I + S + [w])[\sigma_N\rangle(N', M_X + S + [w])$.

Hence, $(N', [i])[\sigma_b \sigma_N c\rangle (N', [o])$ and it becomes clear that N' is indeed relaxed sound.

– Assume N' is relaxed sound. Hence there is a sequence σ: $(N', [i])[\sigma\rangle (N', [o])$. σ needs to have the following structure $\sigma_b = \langle a, \ldots, b, \ldots, c\rangle$ because in order to mark o, c must have been the last step and must have been preceded by b which in turn must have been preceded by a. Recall that $i + u + v + w + o$ is a place invariant illustrating the main control-flow in the net and the linear dependencies between a, b and c. It is also clear that a, b, and c can fire only once. Just before firing c the marking must have been precisely $M_X + S + [w]$ because c does not have any reset arcs. Just after firing b the marking must have been $M_I + S + [w]$. Hence, there exists a firing sequence σ_N such that $(N', M_I + S + [w])[\sigma_N\rangle (N', M_X + S + [w])$. Note that in σ_N only transitions of T can be present ($T_{start} \cup T_{end}$ are dead after removing the token from u). Hence, σ_N is also enabled in the original net, i.e., $(N, M_I)[\sigma_N\rangle (N, M_X)$. Therefore, M_X must be reachable in (N, M_I) thus completing the proof.

$\square$

As shown, relaxed soundness is also undecidable for RWF-nets. It is interesting to note that for proving Theorem 2 we need to use an approach that is completely different from the approach used in the proof of Theorem 1.

7 Conclusion

In this paper we explored decidability of soundness notions in the presence of cancelation. As a basic model, we used RWF-nets, i.e., workflow nets with reset arcs. As shown in Theorem 1, the classical notion of soundness becomes undecidable by adding reset arcs. Moreover, the weaker notion of relaxed soundness is also undecidable for RWF-nets (cf. Theorem 2). Interestingly, the strategies used to prove undecidability are very different for both notions.

In a technical report [5] we also show that most other notions of soundness are undecidable for RWF-nets. Of the many soundness notions described in literature only generalized soundness *may* be decidable (this is still an open problem). All other notions are shown to be undecidable.

We hope that our decidability results are useful for researchers working on workflow verification. The results provide insights into the boundaries of workflow verification. We would like to stress that undecidability does not make things hopeless. Many errors can be discovered using techniques such as invariants and reduction rules [21, 25, 26, 29]. Motivated by the findings in [21], we are planning more empirical studies on workflow verification.

References

1. W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer-Verlag, Berlin, 1997.

2. W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
3. W.M.P. van der Aalst. Workflow Verification: Finding Control-Flow Errors using Petri-net-based Techniques. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 161–183. Springer-Verlag, Berlin, 2000.
4. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems.* MIT press, Cambridge, MA, 2004.
5. W.M.P. van der Aalst, K.M. van Hee, A.H.M. ter Hofstede, N. Sidorova, H.M.W. Verbeek, M. Voorhoeve, and M.T. Wynn. Soundness of Workflow Nets: Classification, Decidability, and Analysis. BPM Center Report BPM-08-02, BPMcenter.org, 2008.
6. J. Dehnert and W.M.P. van der Aalst. Bridging the Gap Between Business Models and Workflow Specifications. *International Journal of Cooperative Information Systems*, 13(3):289–332, 2004.
7. J. Dehnert and P. Rittgen. Relaxed Soundness of Business Processes. In K.R. Dittrich, A. Geppert, and M.C. Norrie, editors, *Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01)*, volume 2068 of *Lecture Notes in Computer Science*, pages 157–170. Springer-Verlag, Berlin, 2001.
8. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, Cambridge, UK, 1995.
9. C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset Nets Between Decidability and Undecidability. In K. Larsen, S. Skyum, and G. Winskel, editors, *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115, Aalborg, Denmark, July 1998. Springer-Verlag.
10. C. Dufourd, P. Jančar, and Ph. Schnoebelen. Boundedness of Reset P/T Nets. In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *Lectures on Concurrency and Petri Nets*, volume 1644 of *Lecture Notes in Computer Science*, pages 301–310, Prague, Czech Republic, July 1999. Springer-Verlag.
11. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology.* Wiley & Sons, 2005.
12. J. Esparza. Decidability and Complexity of Petri Net Problems: An Introduction. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 374–428. Springer-Verlag, Berlin, 1998.
13. J. Esparza and M. Nielsen. Decidability Issues for Petri Nets: A Survey. *Journal of Information Processing and Cybernetics*, 30:143–160, 1994.
14. A. Finkel and Ph. Schnoebelen. Well-structured Transition Systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, April 2001.
15. K.M. van Hee, N. Sidorova, and M. Voorhoeve. Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach. In W.M.P. van der Aalst and E. Best, editors, *Application and Theory of Petri Nets 2003*, volume 2679 of *Lecture Notes in Computer Science*, pages 335–354. Springer-Verlag, Berlin, 2003.
16. K.M. van Hee, N. Sidorova, and M. Voorhoeve. Generalised Soundness of Workflow Nets Is Decidable. In J. Cortadella and W. Reisig, editors, *Application and Theory*

of Petri Nets 2004, volume 3099 of *Lecture Notes in Computer Science*, pages 197–215. Springer-Verlag, Berlin, 2004.

17. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1.* EATCS monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1997.
18. E. Kindler and W.M.P. van der Aalst. Liveness, Fairness, and Recurrence. *Information Processing Letters*, 70(6):269–274, June 1999.
19. F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques.* Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.
20. A. Martens. Analyzing Web Service Based Business Processes. In M. Cerioli, editor, *Proceedings of the 8th International Conference on Fundamental Approaches to Software Engineering (FASE 2005)*, volume 3442 of *Lecture Notes in Computer Science*, pages 19–33. Springer-Verlag, Berlin, 2005.
21. J. Mendling, G. Neumann, and W.M.P. van der Aalst. Understanding the Occurrence of Errors in Process Models Based on Metrics. In F. Curbera, F. Leymann, and M. Weske, editors, *Proceedings of the OTM Conference on Cooperative information Systems (CoopIS 2007)*, volume 4803 of *Lecture Notes in Computer Science*, pages 113–130. Springer-Verlag, Berlin, 2007.
22. F. Puhlmann and M. Weske. Investigations on Soundness Regarding Lazy Activities. In S. Dustdar, J.L. Faideiro, and A. Sheth, editors, *International Conference on Business Process Management (BPM 2006)*, volume 4102 of *Lecture Notes in Computer Science*, pages 145–160. Springer-Verlag, Berlin, 2006.
23. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
24. R. van der Toorn. *Component-Based Software Design with Petri nets: An Approach Based on Inheritance of Behavior*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2004.
25. H.M.W. Verbeek, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Verifying Workflows with Cancellation Regions and OR-joins: An Approach Based on Relaxed Soundness and Invariants. *The Computer Journal*, 50(3):294–314, 2007.
26. H.M.W. Verbeek, M.T. Wynn, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Reduction Rules for Reset/Inhibitor Nets. BPM Center Report BPM-07-13, BPM-center.org, 2007.
27. M. Weske. *Business Process Management: Concepts, Languages, Architectures* . Springer-Verlag, Berlin, 2007.
28. Workflow Patterns Home Page. http://www.workflowpatterns.com.
29. M.T. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Verifying Workflows with Cancellation Regions and OR-joins: An Approach Based on Reset Nets and Reachability Analysis. In S. Dustdar, J.L. Faideiro, and A. Sheth, editors, *International Conference on Business Process Management (BPM 2006)*, volume 4102 of *Lecture Notes in Computer Science*, pages 389–394. Springer-Verlag, Berlin, 2006.