Reduction of Workflow Nets for Generalised Soundness Verification

Hadrien Bride, Olga Kouchnarenko, and Fabien Peureux

Institut FEMTO-ST – UMR CNRS 6174, Univ. Bourgogne Franche-Comté
16, route de Gray, 25030 Besançon, France
{hbride,okouchna,fpeureux}@femto-st.fr

Abstract. This paper proposes a reduction method to verify the generalised soundness of large workflows described as workflow nets–a suited class of Petri nets. The proposed static analysis method is based on the application of six novel *reduction transformations* that transform a workflow net into a smaller one while preserving generalised soundness. The soundness of the method is proved. As practical contributions, this paper presents convincing experimental results obtained using a dedicated tool, developed to validate and demonstrate the effectiveness, efficiency and scalability of this method over a large set of industrial workflow nets.

1 Introduction

Nowadays workflows are extensively used by the economic and scientific communities to model and analyse processes. Indeed, a great diversity of application domains exist today that use workflow management systems on a daily basis in order to control their business processes. These include office automation, healthcare, manufacturing and production, finance and banking, just to name a few. Intuitively, a workflow describes the set of possible runs of a particular system/process by describing the ways in which operations can be carried out to reach its intended goals. With the increasing use of workflows for modelling crucial business processes, analysis and verification of specifications become mandatory to ensure such processes are properly designed and reach the expected level of trust and quality with respect to involving domain and business requirements.

Among proposed workflow modelling languages, workflow Petri nets [1] are well suited for modelling and analysing finite or infinite-state discrete processes exhibiting causalities, concurrencies, and conflicts. Moreover, the development of large and intricate workflow nets can be a difficult task which requires powerful structuring mechanisms [2]. It also forces modellers to follows strict abstraction patterns in order to produce quality workflow nets [3]. For instance, to cope with this challenge, stepwise refinement [4] is often used to ease verification.

Verification of workflow nets is an a posteriori approach: given a workflow net, it checks whether properties (e.g., generalised soundness) hold. Although these properties are usually known to be decidable for workflow nets [5], their verification is a very time consuming task due to the high complexity (EXPSPACE) with respect to the size of the workflow net under analysis [6]. Unfortunately, most often, the abstraction mechanisms, used by modellers of workflow nets, are not explicitly given or deductible to ease the analysis process.

However, within verification approaches, some generic reduction rules [7] have the ability to reduce workflow nets size while strongly preserving properties of interest (e.g., liveness, boundedness). This allows the analysis of studied properties to be performed on reduced workflow nets, in many cases, greatly decreasing its complexity by alleviating state explosion of their state space, which undermines state exploration methods [8]. More generally, reduction rules are abstraction operations: they reduce the level of details of workflow nets, and aim at capturing the abstraction mechanisms used by modellers of workflow nets. It follows that the inversion of reduction rules (i.e. synthesis rules) are refinement operations. Conceptually, this leads to an analysis paradigm where the analysis of workflow nets is substituted by the analysis of their construction.

Within this paradigm, this paper aims to provide an effective and efficient reduction method based on six novel reduction rules to cope with soundness verification of industrial large-scale models. Soundness is indeed a well-established correctness feature for workflow specification that all workflows should verify [5], since it relies on three major properties: weak termination, proper completion, and quasi-liveness. More precisely, the proposed method makes it possible to automate and improve (in terms of calculation time) the generalised soundness verification of large workflows described as workflow nets. Furthermore, in order to conclusively assess the effectiveness, efficiency and scalability of the proposed reduction method, a dedicated tool has been developed and used to conduct intensive experiments over two benchmarks of 1976 industrial workflow nets, which were previously studied in [9–14] by applying others reduction procedures.

The paper is organized as follows. Section 2 introduces related work about the soundness verification of workflows, and motivates the present work. In Sect. 3 we overview the background of the proposed method, i.e. Petri nets and workflow nets. Section 4 details the proposed method to semi-decide the generalised soundness of arbitrary workflow nets. Section 5 describes the tool, called *Hadara-AdSimul-Red*, developed to support the method, and reports on conclusive experimental results. Finally, Sect. 6 concludes the paper and outlines future work.

2 Related Work

On soundness verification. Many techniques and methods have been investigated in order to verify the soundness of workflow nets [12]. It has been proved that generalised soundness of workflow net is decidable [15, 16]. For some subclasses of workflow nets (e.g., well-handled, free choice nets), it has been shown that classical soundness, i.e. (weak) 1-soundness, implies generalised soundness [17]. For these subclasses, generalised soundness can be investigated using model checking techniques. For example, [18] uses the *Woflan* tool to verify soundness of a workflow net through the construction of its reachability graph, whereas [19] uses the well-known *SPIN* model-checker [20]. However, as the state space may be infinite in the general case (even when dealing with particular classes of nets), such approaches cannot be applied without suitable abstractions. Other methods based on structural properties have also been proposed [17, 21]. Nonetheless, establishing these characterizations may similarly become intractable.

On reduction rules. To cope with difficulties arising when facing large workflow nets, some works have investigated reduction techniques to transform a workflow net into a smaller one while preserving some properties of interest such as liveness, boundedness and soundness. For example, when using *Woflan* in [18], reduction rules proposed in [7] are used to reduce workflow nets before analysing soundness. For free-choice Petri nets, a complete set of reduction rules preserving well-formedness is proposed in [22]. Reduction rules preserving deadlock and lack of synchronization conflicts of acyclic workflows are given in [23]. Finally, reduction rules preserving liveness and boundedness are proposed for arbitrary workflow nets in [23–27], and for workflow nets extensions in [13, 28, 29].

The verification method proposed in this paper is based on reduction techniques that are applied to arbitrary workflow nets and that focus on rules preserving generalised soundness. It should be noticed that in contrast with [25] the condition of application of these reduction rules are defined solely structurally and that they extend those previously described in the literature. For instance, all the reduction rules presented in [7] (except the elimination of self-loop place, which cannot be applied to workflow nets) can be seen as special cases of the rules presented in this paper. Further, all abstraction rules (i.e. reduction rules) defined in [24] are also special cases of the rules presented in the present paper.

3 Preliminaries

This section introduces preliminaries on Petri nets [30] and workflow nets [31].

3.1 Petri Nets

Definition 1 (Petri net [30]). *A Petri net is a tuple $\langle P, T, F\rangle$ where 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.*

Let $g \in P \cup T$ and $G \subseteq P \cup T$. We use the following notations: $g^\bullet = \{g' | (g, g') \in F\}$, ${}^\bullet g = \{g' | (g', g) \in F\}$, $G^\bullet = \cup_{g \in G}\ g^\bullet$, and ${}^\bullet G = \cup_{g \in G}\ {}^\bullet g$.

The *marking* of Petri net, representing the number of tokens on each place is a function $M : P \to \mathbb{N}$. It evolves during its execution since transitions change the marking of a Petri net according to the following *firing rules*. A transition t is *enabled* in a marking M if and only if $\forall p \in {}^\bullet t, M(p) \geq 1$. When an *enabled* transition t is *fired*, it *consumes* one token from each place of ${}^\bullet t$ and *produces* one token in each place of $t^\bullet$. Notice that, in the context of *workflow*, specifiers often consider *ordinary* Petri nets [31] (i.e. Petri nets with arcs of weight 1).

Let M_a and M_b be two markings and t a transition of a Petri net N, we denote $M_a \xrightarrow{t} M_b$ the fact that the transition t is *enabled* in marking M_a, and *firing* it results in the marking M_b. The marking M_b is denoted as *directly reachable* from M_a by transition t. Let $M_1, M_2, .., M_n$ be markings and $\sigma = t_1, t_2, .., t_{n-1}$ a sequence of transitions of a Petri net N, we denote $M_1 \xrightarrow{\sigma} M_n$ the fact that $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} .. \xrightarrow{t_{n-1}} M_n$. The marking M_n is then said to be *reachable* from M_1 by the sequence of transitions σ. We denote $\mathcal{R}^N(M)$ the set of markings of N *reachable* from a marking M. Based on these rules, a transition t is *dead* at marking M if it is not *enabled* in any marking M' *reachable* from M. A transition t is *live* if it is not *dead* in any marking *reachable* from the initial marking.

3.2 Workflow Nets

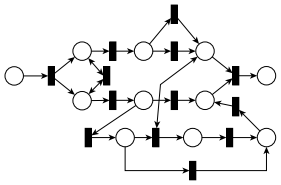


Fig. 1. Example of workflow net

Workflow nets are special cases of Petri nets. They are usually used to model the control-flow dimension of a workflow. They allow the modelling of complex workflow exhibiting concurrencies, conflicts, and causal dependencies of activities. The activities are modelled by transitions, while causal dependencies are modelled by places and arcs. Figure 1 depicts an example of a Petri workflow net. Workflow net, and soundness, k-soundness and generalised soundness within workflow nets, are now defined.

Definition 2 (Workflow net [31]). *A Petri net $N = \langle P, T, F \rangle$ is a workflow net if and only if:*

- *N has two special places i and o, where ${}^\bullet i = \emptyset$ and $o^\bullet = \emptyset$, and*
- *for each node $n \in (P \cup T)$ there exists a path from i to o passing through n.*

We denote $M_{i(k)}$ the initial marking (i.e. $M_i(n) = k$ if $n = i$, and 0 otherwise) and $M_{o(k)}$ the final marking (i.e. $M_o(n) = k$ if $n = o$, and 0 otherwise). Intuitively, soundness means that once a workflow has started it should always be able to terminate without leaving tokens in the net. Formally,

Definition 3 (Soundness [5]). *Let $N = \langle P, T, F \rangle$ be a workflow net, N is sound if and only if:*

- *$\forall M \in \mathcal{R}^N(M_{i(1)}), M_{o(1)} \in \mathcal{R}^N(M)$ (option to complete),*
- *$\forall M \in \mathcal{R}^N(M_{i(1)}), (M(o) > 0) \Rightarrow (M = M_{o(1)})$ (proper completion), and*
- *$\forall t \in T, \exists M, M' \in \mathcal{R}^N(M_{i(1)}), M \xrightarrow{t} M'$ (no dead transitions).*

The notion of k-soundness in [21] extends the classical soundness to k tokens, while a workflow net is generalised sound if it is k-sound for all $k \in \mathbb{N}$.

Definition 4 (k-soundness [21]). *Let $N = \langle P, T, F \rangle$ be a workflow net, and $k \in \mathbb{N}$. N is k-sound if and only if:*

- *$\forall M \in \mathcal{R}^N(M_{i(k)}), M_{o(k)} \in \mathcal{R}(M)$*
- *$\forall t \in T, \exists M, M' \in \mathcal{R}^N(M_{i(1)}), M \xrightarrow{t} M'$*

Definition 5 (Generalised soundness [21]). *Let $N = \langle P, T, F \rangle$ be a workflow net, N is generalised sound if and only if $\forall k \in \mathbb{N}$, N is k-sound.*

4 Verification Method Using Reduction Transformations

This section presents the proposed verification method to check the generalised soundness of arbitrary workflow nets. This method is based on the application of *reduction transformations*. In this way, we define a set of *reduction rules* that allow transforming a workflow net into a smaller one (in terms of the number of nodes) while preserving generalised soundness. We also show that this method is sound: if the successive application of the proposed reduction rules to a workflow net N produces, at the end of this processing (i.e. when no rule can be applied any more), an atomic workflow N_{Atomic}, we can conclude that N is generalised sound. Before describing this method, we first introduce the related basic notions [22].

4.1 Basic Definitions

We define a workflow net transformation rule ϕ as a binary relation on the class of workflow nets. It is fully described by the *conditions of application* under which it can be applied to a *source* workflow net, and the *construction algorithm* that is applied to the *source* workflow net to form a *target* workflow net. Let N, $\tilde{N}$ be two workflow nets and ϕ a transformation rule, the fact that ϕ is *applicable* to N and that applying ϕ to N results in $\tilde{N}$, is denoted $(N, \tilde{N}) \in \phi$. Such a transformation rule ϕ is called a workflow net reduction rule if for all $(N, \tilde{N}) \in \phi$, the number of nodes (i.e. the number of places and transitions) of $\tilde{N}$ is strictly smaller than the number of nodes of N.

Let ψ be a workflow net property, such as generalised soundness (Def. 5), $N \models \psi$ denotes the fact that the workflow net N satisfies ψ. If, for all workflow nets N and $\tilde{N}$, $(N, \tilde{N}) \in \phi$ and $N \models \psi \Rightarrow \tilde{N} \models \psi$, ϕ is said to *preserve* ψ. If the reverse holds as well, i.e. $\tilde{N} \models \psi \Rightarrow N \models \psi$, ϕ is said to *strongly preserve* ψ.

The relation over the class of workflow nets induced by a set of transformation rules is called a kit. Let Φ be the kit induced by the n workflow net transformation rules $\phi_1, .., \phi_n$, Φ defines a binary relation on the class of workflow nets: $(N, \tilde{N}) \in \Phi \Leftrightarrow \exists i \in \{1, .., n\}, (N, \tilde{N}) \in \phi_i$. If a kit is induced by a set of workflow net reduction rules, it is called a reduction kit. We say that Φ (strongly) preserves ψ if and only if $\forall i \in \{1, .., n\}$, ϕ_i (strongly) preserves ψ. Finally, Φ^* denotes the transitive closure of Φ, where $(N_0, N_m) \in \Phi^*$ if and only if there exists a sequence $(\phi_1, N_1), .., (\phi_m, N_m)$ such that $\forall i \in \{1, .., m\}, (N_{i-1}, N_i) \in \phi_i$.

Lemma 1. *Let ψ be a workflow net property, Φ a kit, which strongly preserves ψ, and N, $\tilde{N}$ two workflow nets such that $(N, \tilde{N}) \in \Phi^*$, then $N \models \psi \Leftrightarrow \tilde{N} \models \psi$.*

Our approach is based on Lemma 1. Indeed, defining a kit Φ of reduction rules, which strongly preserve generalised soundness, enables semi-deciding whether a workflow net N is generalised sound. It holds when $(N, N_{Atomic}) \in \Phi^*$, where $N_{Atomic} = \langle \{i, o\}, \{t\}, \{(i, t), (t, o)\} \rangle$ is a generalised sound workflow net.

4.2 Reduction Kit

This section defines the workflow net reduction rules forming a reduction kit, which strongly preserves generalised soundness as introduced in Def. 5. In what follows, each workflow net reduction rule is defined by giving the *conditions of application* under which it can be applied to a source workflow net $N = \langle P, T, F \rangle$, and the *construction algorithm* to apply to N to produce a target workflow net $\tilde{N} = \langle \tilde{P}, \tilde{T}, \tilde{F} \rangle$. Since the *conditions of application* are defined only structurally, avoiding explicit or symbolic exploration of the state-space of the workflow nets under analysis, the approach does not suffer from state explosion. For clarity, every rule is illustrated by a figure depicting two possible applications of this rule. The first one only considers the plain element of the figure and corresponds to the minimal pattern. The second one considers both the mandatory (plain) and the optional (dashed) elements, and corresponds to a possible extended pattern. Figures are thus not exhaustive but aim to clarify the rules formally described by the related conditions of application and the construction algorithm.

R_1: Remove Place.

We define $\phi_{RemoveP}$, a workflow net reduction rule, which strongly preserves generalised soundness and consists in removing a place for which there exists a set of places with the same input transitions as well as the same output transitions. Places removed in such a way are called redundant places as they do not modify the set of correct executions of a workflow net.

Figure 2 illustrates the reduction rule $\phi_{RemoveP}$ formally described as follows.

Conditions on N:	Construction of $\tilde{N}$:
– $\exists\ p \in P \setminus \{i, o\}$ – $\exists\ G = \{g_1, .., g_n\} \subseteq P \setminus \{i, o, p\}$ – $p^\bullet = G^\bullet$ – $^\bullet p = {}^\bullet G$ – $\forall i, j \in \{1, .., n\}, i \neq j \Rightarrow$ $^\bullet g_i \cap {}^\bullet g_j = g_i^\bullet \cap g_j^\bullet = \emptyset$	– $p^\bullet = \{ot_1, .., ot_n\}$ – $outArc := \{p\} \times p^\bullet$ – $^\bullet p = \{it_1, .., it_m\}$ – $inArc := {}^\bullet p \times \{p\}$ – $\tilde{P} := P \setminus \{p\}$ – $\tilde{T} := T$ – $\tilde{F} := F \setminus (inArc \cup outArc)$

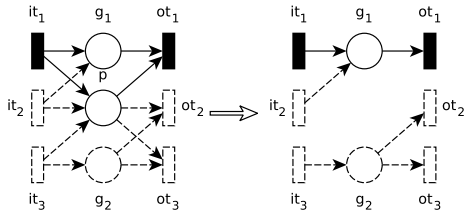


Fig. 2. Reduction rule $\phi_{RemoveP}$ (R_1)

This rule generalises the *Fusion of Parallel Places* rule given in [7] and the *Abstraction of Parallel Places* rule given in [24]. It can also be seen as an adaptation of the rule ϕ_S of [22] (reduction rule proved to be complete with respect to the subclass of free-choice Petri nets) to the context of ordinary workflow nets.

The inverse of this rule is the only synthesis rule able to add a single place to a workflow net while preserving generalised soundness and is notably used to introduce concurrency. In the context of Petri net, a self-loop place (i.e. a place p such that $^\bullet p = p^\bullet$) can be added without compromising liveness and boundedness. However, this requires changing the initial marking which is not possible within workflow nets. However, a generalisation of this rule could be applied to an extension of workflow nets modelling resources by marked places.

The soundness of $\phi_{RemoveP}$, with respect to generalised soundness, is given by the following proposition.

Proposition 1 (Soundness of $\phi_{RemoveP}$). *$\phi_{RemoveP}$ is a workflow net reduction rule which strongly preserves generalised soundness.*

Proof. (Sketch). Let $f : (\tilde{P} \to \mathbb{N}) \to (P \to \mathbb{N})$ be a bijective function such that $f(M)(g) = M(g)$ for all $g \in \tilde{P}$ and $f(M)(p) = M(g_1) + .. + M(g_n)$. By conditions on N, every transition that produces (resp. consumes) a token in any of the places of G also produces (resp. consumes) a token in p. Consequently, $\forall k \in \mathbb{N}$ one has: $M \in \mathcal{R}^{\tilde{N}}(M^{\tilde{N}}_{i(k)}), M^{\tilde{N}}_o(k) \in \mathcal{R}^{\tilde{N}}(M) \Leftrightarrow f(M) \in \mathcal{R}^N(M^N_{i(k)}), M^N_o(k) \in \mathcal{R}^N(f(M))$, and transitions $ot_1, .., ot_n, it_1, .., it_m$ of N are not dead if and only if transitions $ot_1, .., ot_n, it_1, .., it_m$ of $\tilde{N}$ are not dead.

R_2: Remove Transition.

We define $\phi_{RemoveT}$, a workflow net reduction rule, which strongly preserves generalised soundness and consists in removing a transition for which there exists a set of transitions having the same input and output places. Intuitively, the transitions removed by this rule are transitions whose firing can be simulated by the firing of a set of other transitions.

Figure 3 illustrates the reduction rule $\phi_{RemoveT}$ formally described below.

Let D be a set of places, we define the function $\vartheta : D \to (P \to \mathbb{N})$ such that:

$$\forall\ d \in P,\ \vartheta(D)(d) = \begin{cases} 1, & \text{if } d \in D \\ 0, & \text{otherwise} \end{cases}$$

Let $f_1, f_2, f_3 : P \to \mathbb{N}$ be three functions, we overload the operator $+, -$ and $=$ such that $f_3 = f_1\ \Omega\ f_2 \Leftrightarrow \forall p \in P, f_3(p) = f_1(p)\ \Omega\ f_2(p)$ where $\Omega \in \{+, -\}$. The function ϑ is used to compare inputs and outputs of a set of transitions. Note here that this function does not consider self-loop transitions, a desired property as self-loop transition can be added to places (see rule R_3, introduced in the next subsection).

Conditions on N:	Construction of $\tilde{N}$:
– $\exists\ t \in T$ – $\exists\ G = \{g_1, .., g_n\} \subseteq T \setminus \{t\}$ – $\vartheta_t = \vartheta(t^\bullet) - \vartheta({}^\bullet t)$ – $\vartheta_G = \vartheta(g_1^\bullet) + .. + \vartheta(g_n^\bullet) - \vartheta({}^\bullet g_1) - .. - \vartheta({}^\bullet g_n)$ – $\vartheta_t = \vartheta_G$ – $\forall i, j \in \{1, .., n\}, i \neq j \Rightarrow$ $({}^\bullet g_i \cap {}^\bullet g_j = g_i^\bullet \cap g_j^\bullet = \emptyset)$ – $(\exists\ t_s \in T \setminus (\{t\} \cup G), \forall g \in G, {}^\bullet g \subseteq t_s^\bullet) \vee (|G| = 1)$	– $t^\bullet = \{op_1, .., op_{n_1}\}$ – $outArc := \{t\} \times t^\bullet$ – ${}^\bullet t = \{ip_1, .., ip_{n_2}\}$ – $inArc := {}^\bullet t \times \{t\}$ – $\tilde{P} := P$ – $\tilde{T} := T \setminus \{t\}$ – $\tilde{F} := F \setminus (inArc \cup outArc)$

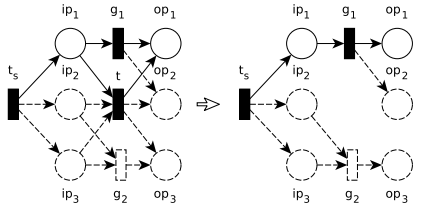


Fig. 3. Reduction rule $\phi_{RemoveT}$ (R_2)

This rule is an original rule generalising the *Fusion of Parallel Transitions* rule of [7] and the *Abstraction of Parallel Transitions* rule of [24]. It is an adaptation of the rule ϕ_S of [22] to the realm of ordinary workflow nets with no restriction on their subclasses. Indeed, outside the scope of free-choice workflow nets, additional constraints are required to ensure that the removed transitions are live. To this end, the liveness of a transition to be removed in such a way is inferred from the liveness of a source transition, a transition that, when fired, enables the transition to be removed. Note that this requirement could be relaxed. Instead of requiring the presence of a source transition, one could require the presence of a sequence of transitions, where each successive transition is enabled by the firing of the previous ones, such that the firing of this sequence of transitions enables the transition to be removed.

The inverse rule of this reduction rule is a synthesis rule able to add a single transition to an arbitrary workflow net based on its structure while preserving generalised soundness and is used to introduce choice.

The soundness of $\phi_{RemoveT}$, with respect to generalised soundness, is given by the following proposition.

Proposition 2. *$\phi_{RemoveT}$ is a workflow net reduction rule which strongly preserves generalised soundness.*

Proof. (Sketch). Let us suppose that $\vartheta_t = \vartheta_G$. By conditions on N, for all k in $\mathbb{N}$, and $\forall M^N$ in $\mathcal{R}^N(M^N_{i(k)})$, transition t is enabled if and only if transitions $g_1, .., g_n$ are also enabled. Moreover, the firing of t must result in the same marking as the successive firing of transitions $g_1, .., g_n$ in any order. It follows that $\forall k \in \mathbb{N}, M \in \mathcal{R}^N(M^N_{i(k)}), M^N_o(k) \in \mathcal{R}^N(M) \Leftrightarrow M \in \mathcal{R}^{\tilde{N}}(M^{\tilde{N}}_{i(k)}), M^{\tilde{N}}_o(k) \in \mathcal{R}^{\tilde{N}}(M)$. To conclude, suppose $|G| = 1$, then t is not dead in N if and only if g_1 is not dead in $\tilde{N}$. Alternatively, suppose $(\exists\ t_s \in T \setminus (\{t\} \cup G), \forall g \in G, {}^\bullet g \subseteq t_s^\bullet)$, then t is not dead in N if and only if t_s is not dead in $\tilde{N}$.

R_3: Remove Self-loop.

We define $\phi_{RemoveST}$, a workflow net reduction rule, which strongly preserves generalised soundness and consists in removing a transition whose input places are its output places.

Figure 4 illustrates $\phi_{RemoveST}$ that is formally described below.

Conditions on N:	Construction of $\tilde{N}$:
– $\exists\ t \in T$ – $t^\bullet = {}^\bullet t$ – $\exists\ t_s \in T \setminus \{t\}, {}^\bullet t \subseteq t_s^\bullet\ \vee\ {}^\bullet t \subseteq {}^\bullet t_s$	– $outArc := \{t\} \times t^\bullet$ – $inArc := t^\bullet \times \{t\}$ – $\tilde{P} := P$ – $\tilde{T} := T \setminus \{t\}$ – $\tilde{F} := F \setminus (inArc \cup outArc)$

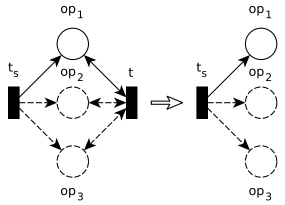


Fig. 4. Reduction rule $\phi_{RemoveST}$ (R_3)

This original rule generalises the *Self-Loop Transition* rule described in [7]. Similarly to rule $\phi_{RemoveT}$, the liveness of a transition removed by this rule also needs to be inferred from the existence of a source transition (alternatively a source sequence of transitions).

The inverse of this reduction rule is a synthesis rule able to add a single transition to an arbitrary workflow net while preserving generalised soundness. It is typically used to introduce choice and repetitive tasks.

The soundness of $\phi_{RemoveST}$, with respect to generalised soundness, is given by the next proposition.

Proposition 3. *$\phi_{RemoveST}$ is a workflow net reduction rule which strongly preserves generalised soundness.*

Proof. (Sketch). By conditions imposed on N, we know that the firing of transition t does not change the markings of N in which it is enabled. It follows that $\forall k \in \mathbb{N}, M \in \mathcal{R}^N(M^N_{i(k)}), M^N_o(k) \in \mathcal{R}^N(M) \Leftrightarrow M \in \mathcal{R}^{\tilde{N}}(M^{\tilde{N}}_{i(k)}), M^{\tilde{N}}_o(k) \in \mathcal{R}^{\tilde{N}}(M)$. Notice that t is not dead in N if and only if t_s is not dead in $\tilde{N}$.

R_4: Remove Transition Place

We define $\phi_{RemoveTP}$, a workflow net reduction rule, which strongly preserves generalised soundness and consists in removing a place and its only input transition. Intuitively, this rule consists in removing a place p and its only input transition t by merging transition t with the output transitions of place p.

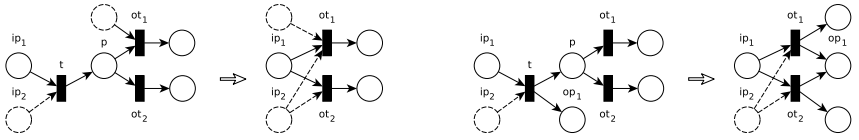
The $\phi_{RemoveTP}$ rule is depicted in Fig. 5, and formally described below.

Conditions on N:

- $\exists\ p \in P \setminus \{i, o\}$
- ${}^\bullet p = \{t\}$
- $t^\bullet \neq \{p\} \Rightarrow$
  $\forall\ ot \in p^\bullet, {}^\bullet ot = \{p\}$
  $\wedge\ t^\bullet \cap ot^\bullet = \emptyset$
- $t^\bullet = \{p\} \Rightarrow$
  $\forall\ ot \in p^\bullet, {}^\bullet t \cap {}^\bullet ot = \emptyset \wedge (\exists\ ot \in p^\bullet,\ {}^\bullet ot = \{p\}\ \vee\ \forall\ ip \in {}^\bullet t,\ ip^\bullet = \{t\})$

Construction of $\tilde{N}$:

- $t^\bullet \setminus p = \{op_1, .., op_{n_1}\}$
- ${}^\bullet t = \{ip_1, .., ip_{n_2}\}$
- $p^\bullet = \{ot_1, .., ot_{n_3}\}$
- $outT := \{t\} \times t^\bullet \setminus p$
- $inT := {}^\bullet t \times \{t\}$
- $outP := \{p\} \times p^\bullet$
- $inArc := {}^\bullet t \times p^\bullet$
- $outArc := p^\bullet \times t^\bullet \setminus p$
- $\tilde{P} := P \setminus \{p\}$,
- $\tilde{T} := T \setminus \{t\}$
- $\tilde{F} := (F \cup inArc \cup outArc) \setminus ((t, p) \cup inT \cup outT \cup outP)$



(a) $t^\bullet = \{p\}$ (b) $t^\bullet \neq \{p\}$

Fig. 5. Reduction rule $\phi_{RemoveTP}$ (R_4)

This rule generalises the *Post-Fusion* rule of [25, 29]. Its inverse is a synthesis rule introducing a sequence of tasks (adding a task that has to be accomplished before others) by factoring common input/output places of a set of transitions.

The soundness of $\phi_{RemoveTP}$, with respect to generalised soundness, is given by the following proposition.

Proposition 4. *$\phi_{RemoveTP}$ is a workflow net reduction rule which strongly preserves generalised soundness.*

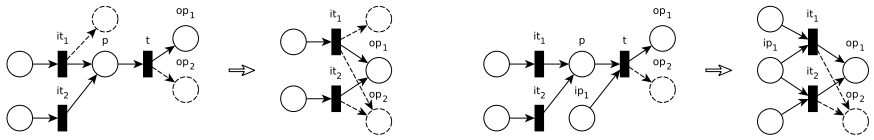
Proof. (Sketch). In N the transitions $ot_1, .., ot_{n_3}$ have to consume a token in place p. All tokens consumed in place p have to be produced by transition t, which consumes a token in places $ip_1, .., ip_{n_2}$ and produces a token in places $op_1, .., op_{n_1}$ and p. Thus, $ot_1, .., ot_{n_2}$ have to consume a token in $ip_1, .., ip_{n_2}$ and produce a token in $op_1, .., op_{n_1}$. Conversely, the same analysis holds on $\tilde{N}$, we conclude that N is generalised sound if and only if $\tilde{N}$ is generalised sound.

R_5: Remove Place Transition

We define $\phi_{RemovePT}$, a workflow net reduction rule, which strongly preserves generalised soundness and consists in removing a place and its only output transition. Intuitively, this rule consists in removing a place p and its only output transition t by merging transition t with the input transitions of place p.

The $\phi_{RemovePT}$ rule is depicted in Fig. 6, and formally described below.

Conditions on N:	Construction of $\tilde{N}$:
– $\exists\ p \in P \setminus \{i, o\}$ – $p^\bullet = \{t\}$ – $^\bullet t \neq \{p\} \Rightarrow$ $\forall\ it \in {}^\bullet p, it^\bullet = \{p\}$ $\wedge\ {}^\bullet t \cap {}^\bullet it = \emptyset \wedge ({}^\bullet it)^\bullet = \{it\}$ – $^\bullet t = \{p\} \Rightarrow$ $\forall\ it \in {}^\bullet p, t^\bullet \cap it^\bullet = \emptyset$	– $t^\bullet = \{op_1, .., op_{n_1}\}$ – ${}^\bullet t \setminus p = \{ip_1, .., ip_{n_2}\}$ – ${}^\bullet p = \{it_1, .., it_{n_3}\}$ – $outT := \{t\} \times t^\bullet$ – $inT := {}^\bullet t \setminus p \times \{t\}$ – $inP := {}^\bullet p \times \{p\}$ – $inArc := {}^\bullet t \setminus p \times {}^\bullet p$ – $outArc := {}^\bullet p \times t^\bullet$ – $\tilde{P} := P \setminus \{p\}$, – $\tilde{T} := T \setminus \{t\}$ – $\tilde{F} := (F \cup inArc \cup outArc) \setminus ((p, t) \cup inT \cup outT \cup inP)$



(a) $^\bullet t = \{p\}$ (b) $^\bullet t \neq \{p\}$

Fig. 6. Reduction rule $\phi_{RemovePT}$ (R_5)

This rule generalises the *Pre-Fusion* rule of [25, 29] as well as the reduction rule ϕ_A proposed in [22].

The inverse of this rule is a synthesis rule introducing a sequence of tasks (adding a task that have to be accomplished after others) and able to factor common input/output places of a set of transitions.

The soundness of $\phi_{RemovePT}$, with respect to generalised soundness, is given by the following proposition.

Proposition 5. *$\phi_{RemovePT}$ is a workflow net reduction rule which strongly preserves generalised soundness.*

Proof. (Sketch). In N the transitions $it_1, .., it_{n_3}$ have to produce a token in place p. All tokens produced in place p have to be consumed by transition t, which consumes a token in places $ip_1, .., ip_{n_2}$ and p, and produces a token in places $op_1, .., op_{n_1}$. Thus, $it_1, .., it_{n_3}$ have to consume a token in $ip_1, .., ip_{n_2}$ and produce a token in $op_1, .., op_{n_1}$. Conversely, the same analysis holds on $\tilde{N}$, we conclude that N is generalised sound if and only if $\tilde{N}$ is generalised sound.

R_6: Remove Ring

We define $\phi_{RemoveR}$, a workflow net reduction rule, which strongly preserves generalised soundness, and consists in merging places among a ring. A ring is a set of places strongly connected by transitions with a single input place and a single output place. The transitions forming the ring are also removed. Intuitively, tokens among the places of a ring can freely move from a place of the ring to an other, therefore they might as well be on the same place.

Figure 7 illustrates the rule $\phi_{RemoveR}$ that is formally described below.

Conditions on N:

- $\exists\ \{p_1, .., p_n\} \subseteq P$
- $\exists\ \{t_1, .., t_m\} \subseteq T$
- $\forall\ i \in \{1, .., m\}, |{}^\bullet t_i| = |t_i^\bullet| = 1$
- $\forall\ i, j \in \{1, .., n\}, {}^\bullet p_i \cap {}^\bullet p_j = p_i^\bullet \cap p_j^\bullet = \emptyset$
- $\forall\ i, j \in \{1, .., m\},\ \exists\ \sigma : \{1, .., k\} \rightarrow \{p_1, .., p_n\} \cup \{t_1, .., t_m\}$ a path of length k such that $\sigma(1) = p_i \wedge \sigma(k) = p_j \wedge \forall\ x \in \{1, .., k-1\}, (\sigma(x), \sigma(x+1)) \in F$

Construction of $\tilde{N}$:

- $ringArc := ((\{p_1, .., p_n\} \times \{t_1, .., t_m\}) \cup (\{t_1, .., t_m\} \times \{p_1, .., p_n\})) \cap F$
- $inT := {}^\bullet p_1 \cup .. \cup {}^\bullet p_n$
- $outT := p_1^\bullet \cup .. \cup p_n^\bullet$
- $removedA := ((inT \times \{p_1, .., p_n\}) \cup (\{p_1, .., p_n\} \times outT)) \cap F$
- $addA := (inT \times p) \cup (p \times outT)$
- $\tilde{P} := (P \cup p) \setminus \{p_1, .., p_n\}$
- $\tilde{T} := T \setminus \{t_1, .., t_m\}$
- $\tilde{F} := (F \cup addA) \setminus removedA$

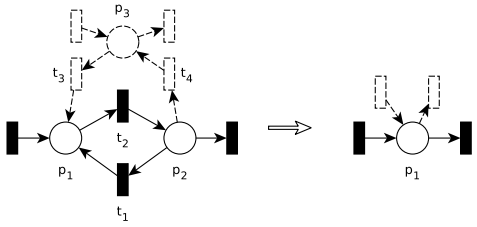


Fig. 7. Reduction rule $\phi_{RemoveR}$ (R_6)

This rule is an original one. Its inverse is a synthesis rule which transforms a place into a ring, distributing its input and output transitions among the places of the created ring.

The soundness of $\phi_{RemoveR}$, with respect to generalised soundness, is given by the following proposition.

Proposition 6 (Soundness of $\phi_{RemoveR}$). *$\phi_{RemoveR}$ is a workflow net reduction rule which strongly preserves generalised soundness.*

Proof. (Sketch). By conditions imposed on N, tokens among the places $p_1, .., p_n$ of a ring can freely move from a place of the ring to an other by firing a sequence of transitions formed with transitions $t_1, .., t_m$. It follows that each token produced (resp. consumed) by an input (resp. output) transition of a place in the ring will eventually be (resp. has been), after (resp. before) the firing of a possibly empty sequence of transitions formed with transitions $t_1, .., t_m$, consumed (resp. produced) by any output (resp. input) transitions of a place of the ring. Likewise, in $\tilde{N}$ each token produced (resp. consumed) by an input (resp. output) transition of p will be (resp. has been) consumed (resp. produced) by an output (resp. input) transition of p. It follows that N is generalised sound if and only if $\tilde{N}$ is generalised sound.

This section defined six reduction rules, which together constitute a generic reduction kit, denoted Φ^*, preserving generalised soundness. These rules generalise the rules previously presented in the literature [25, 7, 29, 26, 27, 23, 22, 24] and thereby extend the range of workflow nets reducible in such a way.

4.3 Verification Algorithm

This section proposes an algorithm, based on the workflow net reduction rules previously described, for semi-deciding the generalised soundness. Our approach is based on Lemma 1. This lemma allows us to infer the generalised soundness a workflow net from its transformed instance as long as the transformation rules applied to obtain it strongly preserve generalised soundness.

The reduction kit Φ^*, composed of the six reduction rules introduced in the previous section, strongly preserves generalised soundness. Therefore, let N and $\tilde{N}$ be two workflow nets such that $(N, \tilde{N}) \in \Phi^*$ then the workflow N is generalised sound if and only if the workflow net $\tilde{N}$ is generalised sound. Furthermore, it is trivial that the workflow net N_{Atomic} (i.e. a workflow net composed of a single transition whose input place is the initial place and output place is the final place) is generalised sound. It follows that if $(N, N_{Atomic}) \in \Phi^*$ then the workflow net N is generalised sound. Since the reduction *kit* Φ^* is not complete with respect to generalized soundness over ordinary workflow nets, this leads to the design of an algorithm to semi-decide whether a workflow net N is generalised sound.

This algorithm proceeds by iteratively trying to apply any of the reduction rules of Φ^* to the input the workflow net N until a fix-point is reached (none of the reduction rules can be applied). If the resulting workflow net equals N_{Atomic}, one can conclude that N is generalised sound. Otherwise, one cannot directly conclude about generalised soundness, but the reduced workflow net is saved to be further analysed using classical techniques such as model-checking.

This procedure is described by Algorithm 1, which is based on: (i) the set of workflow net reduction rules $\Phi = \{R_1, .., R_6\}$, (ii) an auxiliary function $size(N)$, which returns the number of nodes of a workflow net N at each iteration step, (iii) a function $TryApplyRule(\phi, N)$, which returns either $\tilde{N}$ if the rule ϕ can be applied to N to produce $\tilde{N}$, or N otherwise, and (iv) $save(N)$, a function that saves N.

```
Data: N = ⟨P, T, F⟩
Result: Generalised soundness of N
int sizeN = 0;
do
    sizeN = size(N);
    N = ApplyReductionRules(N);
while size(N) < sizeN;
if N = N_Atomic then
    return true;
else
    save(N);
    return unknown;
end
```

```
Function ApplyReductionRules(N)
    forall the φ ∈ Φ do
        int subsizeN = 0;
        do
            subsizeN = size(N);
            N = TryApplyRule(φ, N);
        while size(N) < subsizeN;
    end
    return N;
```

Algorithm 1: Generalised soundness semi-decision algorithm

Theorem 1. *The procedure described by Algorithm 1 terminates.*

Proof. (Sketch). Every rule applied by Algorithm 1 strictly reduces the number of nodes of N. None of the applied rules can produce a workflow net with less than one node. Thus, it always terminates when no workflow net reduction rules can be applied, providing an atomic sound net or saving a reduced workflow net.

Theorem 2. *The procedure described by Algorithm 1 is sound.*

Proof. (Sketch). The set of workflow net reduction rules strongly preserves generalised soundness. By Theorem 1 the procedure of Algorithm 1 is thus sound.

Before concluding this section, let us remark again that the presented reduction procedure is not complete, and leads to a semi-decision procedure. However, this paper presents generalizations of previously known rules [25, 7, 29, 26, 27, 23, 22, 24]. Consequently, in comparison with the previously cited results, which are also not complete for arbitrary workflow nets, the rules introduced in this paper are able to further reduce workflow nets allowing other analysis approaches (e.g., [16, 18]) to be carried out on smaller instances, enabling to increase their scalability. Finally, an extension of the range of reducible workflow nets is a work in progress; to this end, the presented rules would be further generalised or extended to handle generalised Petri nets instead of ordinary ones.

5 Verification Tool and Experimental Results

A dedicated open source tool suite, called *Hadara-AdSimul-Red*[1], has been developed to conduct intensive experiments in order to evaluate the effectiveness, efficiency and scalability of the proposed reduction method. This section presents this tool, as well as convincing experimental results demonstrating its benefits.

5.1 Verification Tool

Figure 8 depicts the global architecture of the verification tool *Hadara-AdSimul-Red*, which has been developed (in `C++`) to support the proposed method.

This tool takes as input an ordinary workflow net (1), saved as an XML file and conform to an ad-hoc and proprietary standard – a dedicated meta-model –, or to the PNML standard [32] – a generic Petri nets XML format – so that third party editor can be used (e.g., VipTool [33], WoPeD [34],Yasper [35], PIPE [36]).

Hadara-AdSimul-Red (2) then tries to apply any of the six reduction rules presented in Section 4.2 to the input workflow net until a fix-point is reached(i.e. none of the reduction rules can be applied) by following the procedure described by Algorithm 1.

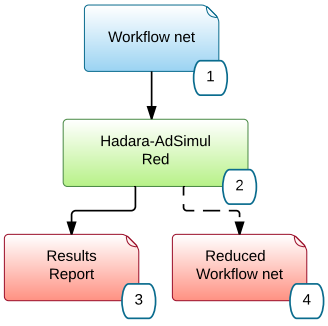


Fig. 8. Tool architecture

[1] The tool *Hadara-AdSimul-Red* (including examples and source code) is available in Github: `https://github.com/LoW12/Hadara-AdSimul`

Once the computation is completed, the tool provides a result report (3) including the status of the verification regarding the generalised soundness, and metrics about execution time and size reduction. Whenever the generalised soundness verification is inconclusive (i.e. the workflow net cannot be completely reduced), the resulting reduced workflow net (4) is saved for further analysis.

In order to ease and foster its use, this tool features a Web interface available online[2]. As an example, Fig. 9 shows the Web screenshot displaying the verification results obtained using an industrial workflow net as input. Information and graphical representations of the original and resulting reduced workflow nets are respectively shown at the left and right of the central frame, which displays the spent reduction time, the status of the generalised soundness verification, as well as the reduction factor obtained.

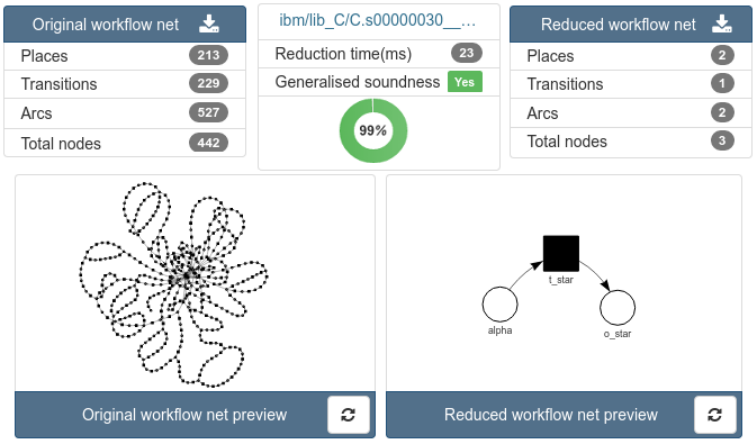


Fig. 9. Screenshot of the *Hadara-AdSimul-Red* verification results

5.2 Experimental Evaluation

This section presents an experimental evaluation of the reduction method proposed in this paper. In a first step, the objectives of this experimentation are stated. In a second step, the experimental protocol, designed to reach the given objectives, is described. Finally, obtained results are presented and discussed.

Objectives – The objectives of this experimental evaluation are to experimentally assess the *effectiveness*, *efficiency* and *scalability* of the proposed reduction method. Formally, the effectiveness of the proposed reduction method is measured with respect to its ability to reduce the size (number of places and transitions) of the considered workflow nets: given a workflow net of size $OriginalSize$, the effectiveness of the proposed reduction method is given by the ratio $ReducedSize/OriginalSize$, where $ReducedSize$ is the size of the workflow net obtained after reduction.

[2] `http://www.adsimul.com/`

Furthermore, the efficiency of the proposed method is evaluated with respect to the time spent to reduce the considered workflow nets. Finally, it follows that the method is scalable over a considered set of workflow nets of growing size if it is able to effectively and efficiently reduce them.

Experimental Protocol – It considers the benchmark suite of 1976 industrial workflow nets previously studied in [9–12] and more recently in [13, 14] where others reduction procedure, also based on reduction rules, have been applied. This benchmark suite is actually composed of two main benchmarks.

The first benchmark, denoted *IBM-bpm*, is a collection of 1386 free choice workflow nets, organized into five libraries (A, B1, B2, B3, and C). All have been derived from industrial business process models provided by IBM®. They have been translated into Petri nets from IBM Web-Sphere Business Modeler[3]'s language (i.e. a language similar to UML activity diagrams [37]) according to [38]. The resulting Petri nets often have multiple sink places and have therefore been completed according to [39] in order to obtain workflow nets. We notably point out that four of the largest workflow nets of this data set are included in the benchmark used by the 2016 Edition of the Model Checking Contest [40]. More information about this data set can be found in the reference paper [12].

The second benchmark, denoted *SAP-ref*, is a collection of 590 workflow nets that have been derived from SAP®'s ERP Software[4] reference models. These 590 industrial workflows have been translated into workflow nets from their original EPCs models [41]. More information about this data set can be found in the reference papers [9–11].

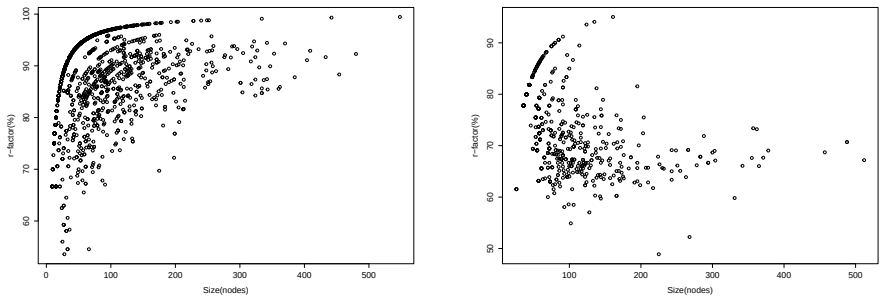
For each workflow net of this benchmark suite, the experimental protocol consists in gathering the obtained reduction factor, the time spent to reach this result, as well as the related generalised soundness verification status.

Results – The experimental results obtained using the dedicated reduction tool described in Sect. 5.1 by applying the experimental protocol introduced previously are now presented. All experimentations have been computed using *Hadara-AdSimul-Red* on a personal laptop featuring an Intel core i7-3740QM @ 2.70GHz processor (using only a single core). The complete data-sets as well as the obtained experimental results are accessible at `https://dx.doi.org/10.6084/m9.figshare.3573756.v5`.

Figures 10(a) and 10(b) respectively present the reduction factors and the sizes of workflow nets of the *IBM-bpm* and *SAP-ref* benchmarks. Table 1 summarizes the overall results of these experiments.

We have analysed 1976 industrial workflow nets and determined that 642 of them are generalised sound (i.e. have been completely reduced). In addition, each workflow has been reduced by an average factor of 82.2%. These experimental results obviously highlight the effectiveness of the proposed reduction method.

[3] `http://www.ibm.com/software/products/en/modeler-basic`
[4] `http://go.sap.com/product/enterprise-management/erp.html`



(a) *IBM-bpm* (b) *SAP-ref*

Fig. 10. Reduction factor with respect to original size

Table 1. Results for the *IBM-bpm* and *SAP-ref* benchmarks

Benchmark	Status	#workflow nets	#Nodes		r-factor(%)		time(ms)	
			avg.	max.	avg.	max.	avg.	max.
IBM-bpm(libA)	Sound	152	61.4	193	93.5	98.4	3.2	44
	unknown	130	99.7	277	86.8	95.1	9.6	51
IBM-bpm(libB1)	Sound	107	38.6	228	86.7	98.7	3.8	67
	unknown	181	98.6	360	82	95.8	7.1	54
IBM-bpm(libB2)	Sound	161	38.9	334	85	99.1	7.3	380
	unknown	202	110.3	404	83	95.8	8.5	60
IBM-bpm(libB3)	Sound	207	47.5	252	87.8	98.8	3.9	56
	unknown	214	125.7	454	85.2	96	8.7	72
IBM-bpm(libC)	Sound	15	127	548	94.4	99.5	7.1	46
	unknown	17	135.6	480	84.7	94.9	5.1	28
***IBM-bpm*(all)**	Sound	642	49	548	**88.4**	99.5	4.6	380
	unknown	744	110.6	480	**84.1**	96	8.3	72
SAP-ref	Sound	0	–	–	–	–	–	–
	unknown	590	97.7	512	**73**	95	9.9	967

With regard to the 1386 free choice workflow nets of *IBM-bpm*, 642 of them were identified as generalised sound by our tool. This results (i.e. the detected soundness) are in agreement with the results obtained during previous experiments applying different analysis techniques to the same data set [12, 14] (i.e. the complete subset of generalised sound workflow nets has been identified by our tool). We underline the fact that all 642 sound workflow nets have been identified through structural reduction by our approach whereas only 464 of them have been identified through structural reduction by Woflan [42]. Furthermore, let us point out that workflow nets of this data set have been on average reduced to 13.9% of their original size (i.e. reduced by a factor of 86.1%). This underlines the greater effectiveness of our method compared with the reduction method of [13] where workflow nets of the same data set have only been reduced to about 23% of their original size (i.e. reduced by a factor of about 77%).

Regarding the 590 workflow nets of *SAP-ref*, only three of them are free choice workflow nets. None of those 590 workflow nets has been found to be generalised sound by our method. However, workflow nets of this data set have been on average reduced to 27% of their original size (i.e. reduced by a factor of 73%). These results further highlight the effectiveness of our reduction method over arbitrary workflow nets. Indeed, once more (albeit to a lesser extent), these results show that our reduction method ability exceeds the one of [13] where workflow nets of the same data set have been reduced to about 35% of their original size (i.e. reduced by a factor of about 65%).

Finally, the results of these experiments based on the *IBM-bpm* and *SAP-ref* benchmarks have also illustrated the efficiency of the proposed reduction method. Indeed, workflow nets of these benchmarks whose sizes are ranging from 9 to 548 nodes have been reduced on average in about 8 ms. Such a short analysis time means that this method could be continuously executed by integrated development environment to provide useful feedback and diagnostic information regarding the generalised soundness of in-development workflow nets.

6 Conclusion

This paper presented an effective, efficient and scalable method to semi-decide the generalised soundness of large-scale workflow nets. This method aims to reduce an arbitrary workflow net into a smaller one while preserving generalised soundness. This enable soundness verification to be carried out over smaller instances of workflow nets, thus drastically decreasing the calculation time required for such a verification. To reach this goal, six novel workflow net reduction rules are proposed and proven to be correct with respect to generalised soundness, a well-established correctness notion in the context of workflow specification. These workflow net reduction rules enable the definition of a reliable and efficient algorithm that semi-decides the generalised soundness of workflow nets.

It also presented conclusive experimental results obtained using a dedicated open-source tool suite implementing the method. Indeed, over a benchmark suite of 1976 industrial workflow nets previously studied in [9–14], it demonstrated convincing size reductions benefits. More precisely, these results illustrated the effectiveness, efficiency and scalability of the proposed reduction method to the verification of generalised soundness by providing, for the considered workflow nets, an average size reduction of 82.2%, in an average time of 8 ms, over industrial workflow nets of size up to 548 nodes.

On the basis of these conclusive results and in order to even more increase the effectiveness of this reduction method, we plan as future work to design and experiment more sophisticated strategies to order the workflow net reduction rules application. From a formal point of view, we also would like to extend the range of workflow nets reducible by adding new rules and further generalising the presented rules by considering generalised Petri nets rather than ordinary Petri nets as stated in Sect. 4 of the present paper. Finally, we are investigating the use of the reduction techniques presented in this paper to optimise the verification of workflow net modal specifications [43] in a way similar to [24].

References

1. van der Aalst, W.M.: Three Good Reasons for Using a Petri-net-based Workflow Management System. Journal of Information and Process Integration in Enterprises **428** (December 1997) 161–182
2. Dittrich, G.: Specification with nets. In: Computer Aided Systems TheoryEUROCAST'89. Springer (1989) 111–124
3. van der Aalst, W.M., Barros, A.P., ter Hofstede, A.H., Kiepuszewski, B.: Advanced workflow patterns. In: International Conference on Cooperative Information Systems, Springer (2000) 18–29
4. Suzuki, I., Murata, T.: A method for stepwise refinement and abstraction of petri nets. Journal of computer and system sciences **27**(1) (1983) 51–76
5. van der Aalst, W.M., van Hee, K.M., ter Hofstede, A.H., Sidorova, N., Verbeek, H., Voorhoeve, M., Wynn, M.T.: Soundness of workflow nets: classification, decidability, and analysis. Journal of Formal Aspects of Computing **23**(3) (2011) 333–363
6. Lipton, R.: The reachability problem requires exponential space. Research report (Yale University. Department of Computer Science). Department of Computer Science, Yale University (1976)
7. Murata, T.: Petri nets: Properties, analysis and applications. IEEE **77**(4) (April 1989) 541–580
8. Valmari, A.: The state explosion problem. In: Lectures on Petri nets I: Basic models. Springer (1998) 429–528
9. Mendling, J., Moser, M., Neumann, G., Verbeek, H., Van Dongen, B.F., van der Aalst, W.M.: Faulty epcs in the sap reference model. In: International Conference on Business Process Management, Springer (2006) 451–457
10. van Dongen, B.F., Jansen-Vullers, M.H., Verbeek, H., van der Aalst, W.M.: Verification of the sap reference models using epc reduction, state-space analysis, and invariants. Computers in Industry **58**(6) (2007) 578–601
11. Mendling, J., Verbeek, H., van Dongen, B.F., van der Aalst, W.M., Neumann, G.: Detection and prediction of errors in epcs of the sap reference model. Data & Knowledge Engineering **64**(1) (2008) 312–329
12. Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Instantaneous soundness checking of industrial business process models. In: Business Process Management. Springer (2009) 278–293
13. Esparza, J., Hoffmann, P.: Reduction rules for colored workflow nets. In: Fundamental Approaches to Software Engineering: 19th International Conference, FASE 2016, Springer Berlin Heidelberg (2016) 342–358
14. Favre, C., Völzer, H., Müller, P.: Diagnostic information for control-flow analysis of workflow graphs (aka free-choice workflow nets). In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer (2016) 463–479
15. Van Hee, K., Sidorova, N., Voorhoeve, M.: Generalised soundness of workflow nets is decidable. Springer (2004)
16. van Hee, K., Oanea, O., Sidorova, N., Voorhoeve, M.: Verifying generalized soundness of workflow nets. In: International Andrei Ershov Memorial Conference on Perspectives of System Informatics, Springer (2006) 235–247
17. Ping, L., Hao, H., Jian, L.: On 1-soundness and soundness of workflow nets. In: Proceedings of the Third Workshop on Modelling of Objects, Components, and Agents Aarhus, Denmark. (2004) 21–36

18. Verbeek, H.M., Basten, T., van der Aalst, W.M.: Diagnosing workflow processes using Woflan. The computer journal **44**(4) (2001) 246–279
19. Yamaguchi, M., Yamaguchi, S., Tanaka, M.: A model checking method of soundness for workflow nets. IEICE transactions on Fundamentals of Electronics, Communications and Computer Sciences **92**(11) (2009) 2723–2731
20. Holzmann, G.J.: The SPIN model checker: Primer and reference manual. Volume 1003. Addison-Wesley Reading (2004)
21. Barkaoui, K., Ben Ayed, R., Sbai, Z.: Workflow soundness verification based on structure theory of Petri nets. Journal of Computing and Information Sciences **5**(1) (2007) 51–61
22. Desel, J., Esparza, J.: Free choice Petri nets. Volume 40. Cambridge university press (2005)
23. Lin, H., Zhao, Z., Li, H., Chen, Z.: A novel graph reduction algorithm to identify structural conflicts. In: System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on, IEEE (2002) 10–pp
24. Hichami, O.E., Al Achhab, M., Berrada, I., Oucheikh, R., El Mohajir, B.E.: An approach of optimisation and formal verification of workflow petri nets. Journal of Theoretical & Applied Information Technology **61**(3) (2014)
25. Berthelot, G.: Transformations and decompositions of nets. In: Petri Nets: Central models and their properties. Springer (1987) 359–376
26. Voorhoeve, M., Van der Aalst, W.: Ad-hoc workflow: problems and solutions. In: Database and Expert Systems Applications, 1997. Proceedings., Eighth International Workshop on, IEEE (1997) 36–40
27. Sadiq, W., Orlowska, M.E.: Analyzing process models using graph reduction techniques. Information systems **25**(2) (2000) 117–134
28. Wynn, M.T., Verbeek, H., van der Aalst, W.M., ter Hofstede, A.H., Edmond, D.: Soundness-preserving reduction rules for reset workflow nets. Information Sciences **179**(6) (2009) 769–790
29. Sloan, R.H., Buy, U.: Reduction rules for time petri nets. Acta Informatica **33**(7) (1996) 687–706
30. Petri, C.A.: Kommunikation mit Automaten. PhD thesis, Darmstadt University of Technology, Germany (1962)
31. van der Aalst, W.M.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems, and Computers **8**(1) (February 1998) 21–66
32. Weber, M., Kindler, E.: The petri net markup language. In: Petri Net Technology for Communication-Based Systems. Springer (2003) 124–144
33. Desel, J., Juhás, G., Lorenz, R., Neumair, C.: Modelling and validation with viptool. In: Business Process Management. Springer (2003) 380–389
34. Freytag, T.: Woped–workflow petri net designer. University of Cooperative Education (2005)
35. Van Hee, K., Oanea, O., Post, R., Somers, L., Van der Werf, J.M.: Yasper: a tool for workflow modeling and analysis. In: Application of Concurrency to System Design, 2006. ACSD 2006. Sixth International Conference on, IEEE (2006) 279–282
36. Bonet, P., Lladó, C.M., Puijaner, R., Knottenbelt, W.J.: PIPE v2.5: A Petri net tool for performance modelling. In: Proc. of the 23^{rd} Latin American Conference on Informatics (CLEI'07), San Jose, Costa Rica (October 2007)
37. Dumas, M., Ter Hofstede, A.H.: Uml activity diagrams as a workflow specification language. In: International Conference on the Unified Modeling Language, Springer (2001) 76–90
38. Fahland, D.: Translating uml2 activity diagrams to petri nets (2008)

39. Kiepuszewski, B., ter Hofstede, A.H., van der Aalst, W.M.: Fundamentals of control flow in workflows. Acta Informatica **39**(3) (2003) 143–209
40. Kordon, F., Garavel, H., Hillah, L.M., Hulin-Hubard, F., Chiardo, G., Hamez, A., Jezequel, L., Miner, A., Meijer, J., Paviot-Adet, E., Racordon, D., Rodriguez, C., Rohr, C., Srba, J., Thierry-Mieg, Y., Trịnh, G., Wolf, K.: Models of the 2016 Edition of the Model Checking Contest. http://mcc.lip6.fr/models.php (June 2016)
41. Lohmann, N., Verbeek, E., Dijkman, R.: Petri net transformations for business processes–a survey. In: Transactions on petri nets and other models of concurrency II. Springer (2009) 46–63
42. Verbeek, E., Van Der Aalst, W.M.: Woflan 2.0 a petri-net-based workflow diagnosis tool. In: International Conference on Application and Theory of Petri Nets, Springer (2000) 475–484
43. Bride, H., Kouchnarenko, O., Peureux, F.: Verifying modal workflow specifications using constraint solving. In: Proceedings of Integrated Formal Methods (IFM'14). Volume 8739 of LNCS., Bertinoro, Italy, Springer (September 2014) 171–186