Soundness-preserving Reduction Rules for Reset Workflow Nets

M.T. Wynn[1], H.M.W. Verbeek[2], W.M.P. van der Aalst[1,2], A.H.M. ter Hofstede[1] and D. Edmond[1]

Business Process Management Group, Queensland University of Technology
GPO Box 2434, Brisbane QLD 4001, Australia.
{*m.wynn,d.edmond,a.terhofstede*}*@qut.edu.au*
Department of Mathematics and Computer Science, Eindhoven University of Technology
PO Box 513, NL-5600 MB Eindhoven, The Netherlands.
{*h.m.w.verbeek,w.m.p.v.d.aalst*}*@tue.nl*

Abstract. The application of reduction rules to any Petri net may assist in its analysis as its reduced version may be significantly smaller while still retaining the original net's essential properties. Reset nets extend Petri nets with the concept of a reset arc, allowing one to remove all tokens from a certain place. Such nets have a natural application in business process modelling where possible cancellation of activities need to be modelled explicitly and in workflow management where such process models with cancellation behaviours should be enacted correctly. As cancelling the entire workflow or even cancelling certain activities in a workflow has serious implications during execution (for instance, a workflow can deadlock because of cancellation), such workflows should be thoroughly tested before deployment. However, verification of large workflows with cancellation behaviour is time consuming and can become intractable due to the state space explosion problem. One way of speeding up verification of workflows based on reset nets is to apply reduction rules. Even though reduction rules exist for Petri nets and some of its subclasses and extensions, there are no documented reduction rules for reset nets. This paper systematically presents such reduction rules. Because we want to apply the results to the workflow domain, this paper focusses on Reset Workflow nets (RWF-nets), i.e. a subclass tailored to the modelling of workflows. The approach has been implemented in the context of the workflow system YAWL.
Keywords: Reset nets, reduction rules, workflow verification, soundness.

1 Introduction

The analysis of a non-trivial concurrent process is a complicated task. There are a number of different approaches to deal with this complexity. Reducing a specification, while preserving its essential properties with respect to a particular analysis problem, is one such approach. There exists a body of research that addresses the concept of reduction in the area of Petri nets (see e.g.,[4, 13]) and its various subclasses (see e.g., free choice nets [6]) and extensions (see e.g., timed petri nets [14]).

Reset nets ([5, 8–11]) extend Petri nets with the concept of a *reset arc*, a type of arc that connects a place and a transition and which semantics is to remove all tokens from

that place when the transition fires. The complexity introduced by a reset arc (when compared with Petri nets in general) is threefold: 1) as the transition removes *all* tokens and not just one, place invariants do not hold for such nets, 2) the reset action can be *ineffective* if a place does not contain any tokens at the exact time when the transition fires and the reset action is carried out, and 3) a reset arc can affect any place in the entire net (i.e., its effect is global), unlike normal arcs of a transition which can only influence their input and output places (i.e., their effect is local). As a result, the notion of reachability is undecidable for reset nets with more than two reset arcs [9].

Reset nets form a natural foundation for workflow languages with explicit support for cancellation. Cancellation is an important concept in workflow management (see e.g., [3]) where the execution of some activities may lead to the termination of other activities *in certain circumstances*. Cancellation can be triggered by either a customer request (e.g., a customer wishes to withdraw a credit card application) or by exceptions (e.g., an order cannot be processed due to insufficient stock level). In general, cancellation results in one of two outcomes: disabling some scheduled activities or stopping currently running activities. The complicating factor is that due to concurrency issues, the cancellation action may or may not result in cancelling certain activities, i.e., the process may be in a state before or after the part that is supposed to be cancelled. This can introduce deadlocks (the state where a business process is stuck and cannot proceed). As it is important to detect errors in workflows before their deployment, it is desirable to speed up the analysis of workflows if possible. When this analysis is performed on reset nets, corresponding to the workflows involved, this boils down to being able to perform efficient reset net analysis. While there are potentially a number of avenues that could be explored in order to achieve this, one approach is the application of reduction rules for reset nets.

When reducing a net it is imperative that certain essential properties are preserved. In the area of workflow verification, soundness is such a property. Soundness is a correctness notion of workflows that requires that a workflow can always terminate, that when termination is signalled no other tokens remain elsewhere in the net, and that it does not contain any dead tasks [15]. A Reset Workflow net (RWF-net) is a reset net with three structural restrictions: there is exactly one source node, one sink node and every node in the graph is on a directed path from the source node to the sink node. See Figure 1 for an example of a sound RWF-net. In Figure 1, transition t can remove tokens from places a, b, and c when it fires (denoted by double-headed arcs). By applying reduction rules, it is possible to reduce the net while preserving the soundness property of the net as shown in Figure 2, where transition y corresponds to transition t and its successors in the original net and place v replaces all three places a, b, and c.

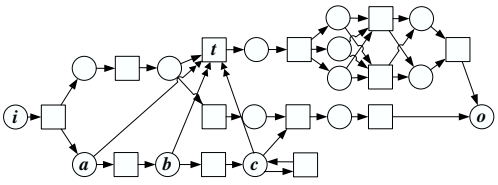

Fig. 1. An example RWF-net

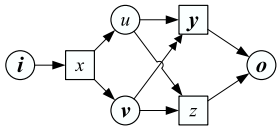

Fig. 2. Reduced RWF-net for the net in Figure 1

In this paper a number of soundness-preserving reduction rules for RWF-nets are documented and proven correct. They are inspired by reduction rules provided for Petri nets in [4, 13] and for Free Choice Petri nets provided in [6]. As mentioned before, there are no documented reduction rules for reset nets that preserve soundness in the current literature, and as it turns out, one has to be careful in determining the conditions under which certain types of reductions can be applied. For example, because of the nature of reset arcs removing *all* tokens when a transition fires, it is possible that an incorrect net that does not satisfy the proper completion criterion (i.e. tokens can be left in the net when it reaches the end) becomes sound when there is a reset arc to remove the leftover tokens before completion. Another interesting finding is that strict conditions are necessary for reset net reduction rules (for instance, two places can only be reduced to one if and only if both places are reset by the same transition). Hence, the presence of reset arcs does not offer new possibilities for reduction, rather it limits them. Also, a reset arc can never be abstracted entirely from a reset net. That is, if a net contains reset arcs, it is not possible to obtain a reduced net *without* any reset arc. Hence, the goal of reduction is to reduce the number of reset arcs when these arcs are connected to more than one place or transition and as a result, to reduce the complexity introduced by multiple reset arcs.

The contributions of the paper are as follows:

- The set of reset net reduction rules represents a practical contribution to achieve a more efficient verification of complex workflows with cancellation behaviours. The rules have been implemented as part of the verification functionality of the workflow language YAWL[1].
- The reduction rules presented in this paper have wider applicability than the area of workflow verification. They are equally applicable to other business process modelling languages that support cancellation such as the Business Process Modelling Notation (BPMN), the Business Process Execution Langauge (BPEL) and the Unified Modelling Language (UML).
- The paper also aims to make a contribution to the body of theory in reset nets. The set of reduction rules presented in this paper are liveness and boundedness preserving as well as soundness preserving.

The organisation of the rest of the paper is as follows. Section 2 provides the formal foundation by introducing reset nets and RWF-nets. Section 3 introduces a simplified credit card application process with cancellation feature and demonstrates how it can be modelled as a RWF-net. Section 4 describes a set of reduction rules for RWF-nets. Section 5 briefly mentions the implementation of the rules in the workflow language YAWL. Section 6 discusses the related work and Section 7 concludes the paper.

2 Preliminaries

This section contains a number of background definitions to make the paper self-contained. A reset net is a Petri net with special *reset arcs*, that can clear the tokens in selected places, and are represented as doubled-headed arrows (see Figure 3).

[1] www.yawl-system.com

Definition 1 (Reset net [8]). *A reset net is a tuple (P, T, F, R) where P is a (non-empty finite) set of places, T is a set of transitions, $P \cap T = \varnothing$, $F \subseteq (P \times T) \cup (T \times P)$ is the set of arcs and $R : T \to \mathbb{P}(P)$ provides the reset places for the transitions.*

Let N be a reset net and $x \in (P \cup T)$, we use $\bullet x$ and $x\bullet$ to denote the set of inputs and outputs. If the net involved cannot be understood from the context, we explicitly include it in the notation and we write ${}^{N}\!\bullet x$ and $x {}^{N}\!\bullet$. Relation F implies a function and $F(x, y)$ evaluates to 1 if $(x, y) \in F$ and 0 otherwise. We write F^* for the reflexive transitive closure of F. The notation $R(t)$ for a transition t returns the (possibly empty) set of places that it resets. We also write $R^{\leftarrow}(p)$ for a place p, which returns the set of transitions that can reset p. A marking is defined as $M : P \to \mathbb{N}$ and, just as with ordinary Petri nets, it can be interpreted as a vector, function, or multiset over the set of places P. $\mathbb{M}(N)$ is used to represent a set of all possible markings of a reset net N. $M(p)$ returns the number of tokens in a place p if $p \in dom(M)$ and 0 otherwise. $M \leq M'$ iff $\forall_{p \in P} M(p) \leq M'(p)$. $M > M'$ iff $\forall_{p \in P} M(p) \geq M'(p) \wedge \exists_{p \in P} M(p) > M'(p)$. $M + M'$ are multisets such that $\forall_{p \in P} : (M + M')(p) = M(p) + M'(p)$. Similarly, $M - M'$ are multisets such that $\forall_{p \in P} : (M - M')(p) = M(p) - M'(p)$ where $M(p) \geq M'(p)$.

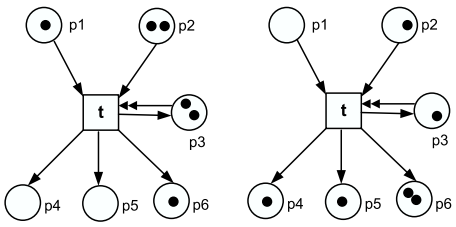

Fig. 3. An example reset net before and after firing transition t

In Figure 3, transition t is enabled at marking $p1 + 2p2 + 2p3 + p6$ as $\bullet t = p1 + p2$ and t may fire. When transition t fires, it removes a token each from its input places $p1$ and $p2$, removes all tokens from its reset place $p3$, and puts one token each in its output places $p3$, $p4$, $p5$, $p6$, resulting in the marking $p2 + p3 + p4 + p5 + 2p6$.

Definition 2 (Forward firing). *Let $N = (P, T, F, R)$ be a reset net, $t \in T$ and $M, M' \in \mathbb{M}(N)$. Transition t is enabled at M, denoted as $M[t\rangle$, iff for all $p \in \bullet t$: $M(p) \geq 1$. We denote $M \overset{N,t}{\to} M'$ iff $M[t\rangle$ and*

$$M'(p) = \begin{cases} M(p) - F(p,t) + F(t,p) & \text{if} \quad p \in P \setminus R(t) \\ F(t,p) & \text{if} \quad p \in R(t). \end{cases}$$

If there can be no confusion regarding the net, the expression is abbreviated as $M \overset{t}{\to} M'$ and if the transition is not relevant, it is written as $M \to M'$. We write $M \overset{N,\sigma}{\to} M_n$ if $\sigma = t_1 t_2 ... t_n$ is an occurrence sequence leading from M to M_n i.e. $M \overset{t_1}{\to} M_1 \overset{t_2}{\to} ... \overset{t_n}{\to} M_n$. The reachability set of the net N from marking M, denoted as $N[M\rangle$, is the minimal set that satisfies the following conditions: (1) $M \in N[M\rangle$ and (2) if transition $t \in T$ and markings $M_1, M_2 \in \mathbb{M}(N)$ exist such that $M_1 \in N[M\rangle$ and $M_1 \overset{N,t}{\to} M_2$,

then $M_2 \in N[M\rangle$. Workflow nets (WF-nets) forms a subclass of Petri nets with unique input and output places, that can be used to represent workflow processes [15]. This notion can be extended to reset nets.

Definition 3 (WF-net and RWF-net). *Let $N = (P, T, F)$ be a Petri net. The net N is a WF-net iff the following three conditions hold:(1) there exists exactly one $i \in P$ such that $\bullet i = \varnothing$, and (2) there exists exactly one $o \in P$ such that $o\bullet = \varnothing$, and (3) for all $n \in P \cup T$: $(i, n) \in F^*$ and $(n, o) \in F^*$. Let $N = (P, T, F, R)$ be a reset net. The net N is an RWF-net iff (P, T, F) is a WF-net.*

Definition 4 (Soundness). *Let $N = (P, T, F, R)$ be an RWF-net and $M_i = i$, $M_o = o$ be the initial and end markings*[2]*. N is sound iff: (1) for every marking M reachable from M_i, there exists an occurrence sequence leading from M to M_o, i.e., for all $M \in N[M_i\rangle : M_o \in N[M\rangle$ (**option to complete**), and (2) the marking M_o is the only marking reachable from M_i with at least one token in place o, i.e., for all $M \in N[M_i\rangle : M \geq M_o \Rightarrow M = M_o$ (**proper completion**), and (3) for every transition $t \in T$, there is a marking M reachable from M_i such that t is enabled at M, i.e., for all $t \in T$ there exists an $M \in N[M_i\rangle$ such that $M[t\rangle$ (**no dead transitions**).*

The soundness definition for an RWF-net is based on the soundness definition for WF-nets [1]. In this definition, the second requirement follows from the first [12]. The option to complete requirement states that it should always be possible to complete the net properly. Suppose we have an RWF-net for which the first item of soundness (option to complete) holds. Assume that the second does not hold, i.e., that there is some marking M reachable such that $M > M_o$. The option to complete guarantees us that from M we can reach M_o. As the tokens in o cannot enable any transition in the net, we thus conclude that from marking $M - M_o$, we can reach the empty marking. As all transitions have non-empty postsets, the empty marking cannot be reached. Hence, the second requirement has to hold as well. Therefore, the soundness definition without the second requirement is sufficient and is equivalent to the original definition. In the next section, we prove for some of our reduction rules for RWF-nets that soundness is preserved by showing that the rules preserve the option to complete and no dead transitions criteria. The soundness property of a WF-net without reset arcs can be determined from its reachability graph and is decidable [1]. Reachability is not decidable for an arbitrary reset net [9], and soundness has been shown to be undecidable for RWF-nets as well [16]. Nevertheless, it is still desirable to be able to perform soundness analysis whenever possible.

3 An Illustrative Example: Credit card application process

In this section, we illustrate a simplified version of a credit card application process using a RWF-net. To showcase the capability of reset arcs, we assume that an applicant can request to cancel the credit card application at any point in time until a decision has been made on the application. We first present the process model using the YAWL

[2] Note the overloading of i and o: they are now used to denote a multiset containing one element.

notation (see Figure 4) and we then present an equivalent RWF-net for the process (see Figure 5). In Section 4, we use this example to demonstrate step-by-step reduction of the model using the proposed reduction rules. In Section 5, the analysis results for this process model using the implementation in YAWL are discussed.

The process starts when an applicant submits a credit card application (with the proposed amount). Upon receiving an application (ra), a credit clerk checks whether the submitted application is complete (cc). If not, the clerk requests additional information from the applicant (rm) and waits until this information is received (ri) before proceeding. At the same time a timer is set (to) so that if a certain period elapses before requested information is received, another request for information is sent again. For a complete application, the clerk first checks the requested loan amount (cl). It is then followed by additional checks to validate the applicant's income and credit history. Different checks are performed depending on whether the requested loan is large (pl) or small (ps). The validated application is then passed on to a manager to make a decision (md). In the case of acceptance, the credit card approval activity can start (sa). The applicant is notified of the decision (na) and at the same time is asked for his/her preference on any extra features (wx). The applicant can choose extra features such as rewards program or secondary cardholders (cf) before a credit card is produced and delivered (dc). This indicates the completion of the approval activity (ca) and the process ends. For a rejected application, the applicant is notified of the rejection (nj) and the process ends. An interesting feature of this process is that an applicant is allowed to cancel an ongoing application at any time after it was received (ra) and before the manager is ready to make a decision (md).

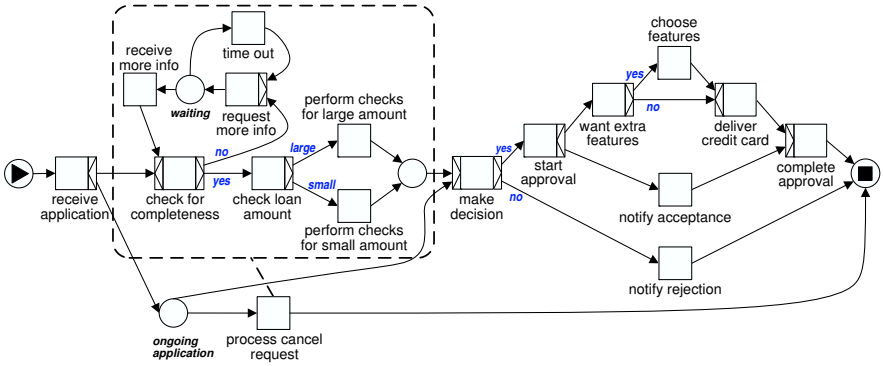

Fig. 4. An example credit card application process modelled in YAWL

Figure 4 depicts a YAWL model of the credit card application process. A task in YAWL is depicted as a rectangle and a place (which represents a state in between tasks) is depicted as a circle. A YAWL model has a unique start point and a unique end point. We will not go through every element of the model but select a number of typical examples for illustration. Firstly, the task check for completeness uses XOR-split to capture the checking result and XOR-join to capture further check after additional information is received. Next, the place waiting models a deferred choice between task receive

more info and task time out. Thirdly, the place ongoing application models a deferred choice between task process cancel request and task make decision. Finally, in the process model, task process cancel request with its associated cancellation region (shown within the dotted lines) capture the withdrawal of an ongoing application before approve/reject decision is made. That is, when task process cancel request is carried out, all tokens from places in the cancellation region are removed and all tasks that are currently executing in the cancellation region are stopped.

Figure 5 depicts an equivalent RWF-net of the credit card application process. In general, a YAWL task is mapped to a start transition and an end transition with a place in between. For instance, task receive application (ra) is modelled as one start transition (ras) and one end transition (rae) with an intermediate place to connect the two transitions. The XOR-split and join behaviour of a YAWL task is modelled using a separate transition for each path. For instance, two possible paths to start task Check for Completeness is depicted as ccs1 and ccs2 and two possible paths after completing check for completeness task is shown as cce1 and cce2. If a task has a cancellation region associated with it, its end transition will have reset arcs (e.g., the end transition of process cancel request (cne) has reset arcs). If a task is in the cancellation region of another task, all tokens from its intermediate place is removed (e.g., task check for completeness is in the cancellation region of task process cancel request and hence, the end transition of process cancel request (cne) will have a reset arc from the intermediate place between transitions ccs1 and cce1). If a place in the YAWL model is in the cancellation region of a task, then its corresponding place in the RWF-net also have a reset arc from that place to the end transition of the task (e.g., place waiting (wt) has a reset arc to transition cne). To make the diagram more readable, we use a region instead of drawing separate reset arcs from every place in the region to transition cne.

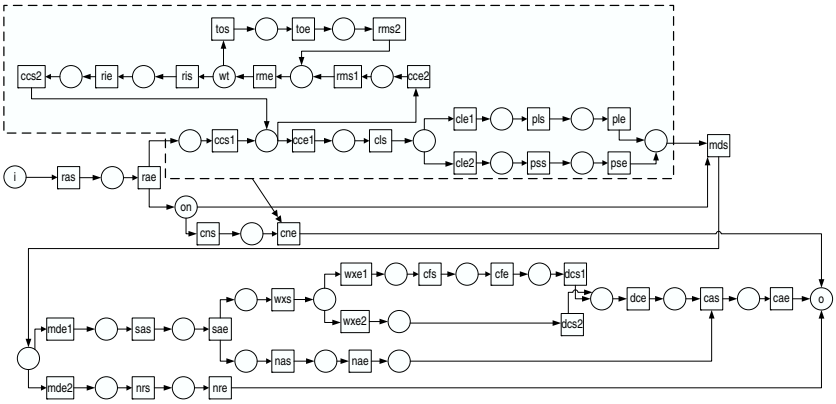

Fig. 5. An equivalent RWF-net for credit card application process in Figure 4

4 Reduction Rules for RWF-nets

In this section, we present seven soundness preserving reduction rules for RWF-nets. For sake of clarity, we have taken a two-step approach: first the reduction rule for WF-nets, then the extension for RWF-nets. The style of this section is taken from [6].

The soundness of WF-nets has been shown to correspond to boundedness and liveness properties of the short-circuited WF-net [1]. Therefore, if a reduction rule for a WF-net preserves boundedness and liveness, then it also preserves soundness. We show that as a reduction rule for a WF-net is boundedness and liveness preserving, it is also soundness preserving. However, soundness of RWF-nets does *not* correspond to boundedness and liveness. It is possible that an unbounded RWF-net is sound due to the presence of reset arcs. In Figure 6, place q is an unbounded place and therefore, the net is unbounded. Transition c resets both preceding places when it fires. As a result, it is not possible for tokens to be left in either p or q when the net completes. Hence, the net is sound and we cannot prove that a reduction rule for RWF-nets preserves soundness by showing that it preserves boundedness and liveness. Therefore, we will show that reduction rules for RWF-nets preserve soundness by proving that they preserve occurrence sequences and hence, preserve the criteria for soundness: the option to complete, and no dead transitions.

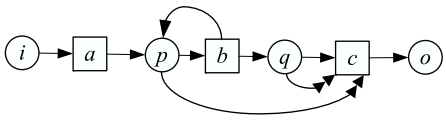

Fig. 6. An example of an unbounded RWF-net which is sound.

Next, we present the first of the seven reduction rules for RWF-nets, the Fusion of series transitions rule, and prove that the soundness property holds for a WF-net and then for an RWF-net.

Please note that the presentation of the rest of the reduction rules follow the exact same structure as the first rule. Our intention is to fully formalise these rules for completeness, for future reference and to present them in a standardised manner for ease of understanding. The reader that wishes to concentrate on the essence of the rules can look at the figures and the conditions and constructions for reset arc extensions of each rule, while the reader that also wishes to convince themselves of the technical correctness of these rules can look at the lemmas, the theorems, and the soundness proofs.

4.1 Fusion of series transitions

The *Fusion of Series Transitions Rule for WF-nets* (ϕ_{FST}) allows for the merging of two sequential transitions t and u with one place p in between these two transitions into only one transition v. The rule requires that there is only one input t and output u for the place p, p is the only input of u, and there are no direct connections between outputs of t and outputs of u. The last requirement ensures that there will only be one arc connecting the new transition v to outputs of t in the reduced net. See the example

in Figure 7 for an application of the ϕ_{FST} rule. Transitions t and u have been merged into a new transition v in the right net. Note that transitions u and x cannot be merged as x has two input places (q and r).

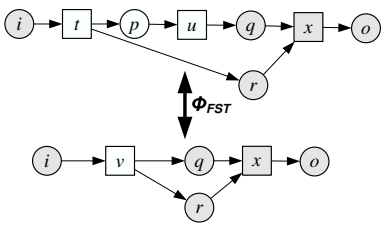

Fig. 7. Reduction of a WF-net using the ϕ_{FST} rule

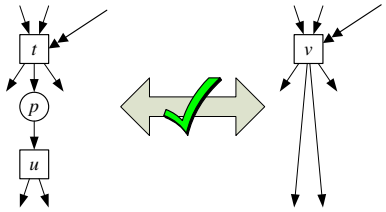

Fig. 8. Fusion of Series Transitions Rule for RWF-nets: ϕ^R_{FST}

Definition 5 (Fusion of Series Transitions Rule for WF-nets: ϕ_{FST}). *Let N_1 and N_2 be two WF-nets, where $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$. $(N_1, N_2) \in \phi_{\mathrm{FST}}$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, a place $p \in P_1$, two transitions $t, u \in T_1$, and a transition $v \in T_2 \setminus T_1$ such that:*

Conditions on N_1:

1. $\bullet p = \{t\}$ *(t is the only input of p)*
2. $p\bullet = \{u\}$ *(u is the only output of p)*
3. $\bullet u = \{p\}$ *(p is the only input of u)*
4. $t\bullet \cap u\bullet = \varnothing$ *(any output of t is not an output of u and vice versa)*

Construction of N_2:

5. $P_2 = P_1 \setminus \{p\}$
6. $T_2 = (T_1 \setminus \{t, u\}) \cup \{v\}$
7. $F_2 = (F_1 \cap ((P_2 \times T_2) \cup (T_2 \times P_2))) \cup ({}^{N_1}\!\bullet t \times \{v\}) \cup (\{v\} \times ((t^{N_1}\!\bullet \cup u^{N_1}\!\bullet) \setminus \{p\}))$

Theorem 1 (The ϕ_{FST} rule is soundness preserving). *Let N_1 and N_2 be two WF-nets such that $(N_1, N_2) \in \phi_{\mathrm{FST}}$. Then N_1 is sound* iff *N_2 is sound.*

Proof The ϕ_{FST} rule is boundedness and liveness preserving [13]. Soundness of a WF-net corresponds to boundedness and liveness of the short-circuited WF-net [1].

The *Fusion of Series Transitions Rule for RWF-nets* (ϕ^R_{FST}) extends the ϕ_{FST} rule by introducing reset arcs. The rule also allows for the merging of two sequential transitions t and u with one place p in between them into a single transition v. Figure 8 visualises the ϕ^R_{FST} rule. Additional requirements (required to allow for reset arcs) are that place p and output places of u should not be the source of any reset arcs and transition u should not reset any place. The rule allows reset arcs from transition t and these

arcs will be assigned to the new transition v in the reduced net. Figure 9(a) shows a counter-example where p is a reset place: transition sequence tx leads to a deadlock, which does not exist in the other net. Figure 9(b) shows a counter-example where transition u has reset arcs: transition sequence tu leads to a deadlock, which does not exist in the other net. Figure 9(c) shows a counter-example where the postset of u contains a reset place: transition sequence txu results in two tokens in place r, which is not possible in the right net. As a result, the left net is not sound whereas the right net is.

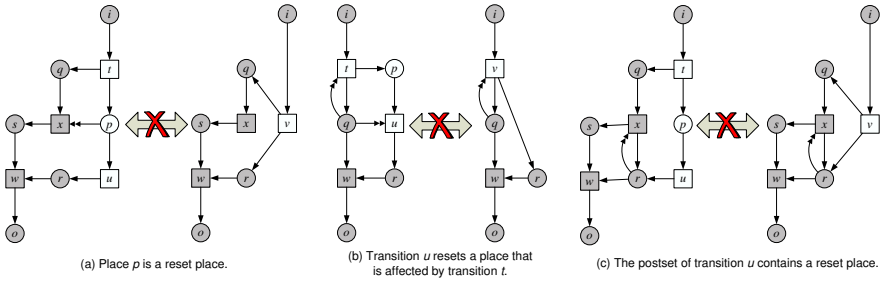

Fig. 9. counter-examples for the ϕ_{FST} rule applied to reset nets

Definition 6 (Fusion of Series Transitions Rule for RWF-nets: ϕ^R_{FST}). *Let N_1 and N_2 be two RWF-nets, where $N_1 = (P_1, T_1, F_1, R_1)$ and $N_2 = (P_2, T_2, F_2, R_2)$. $(N_1, N_2) \in \phi^R_{\mathrm{FST}}$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, a place $p \in P_1$, two transitions $t, u \in T_1$, and a transition $v \in T_2 \setminus T_1$ such that:*

Extension of the ϕ_{FST} rule:

1. $((P_1, T_1, F_1), (P_2, T_2, F_2)) \in \phi_{\mathrm{FST}}$ *(Note that, by definition, the t, u, v, and p mentioned in this definition have to coincide with the t, u, v, and p as mentioned in the definition of ϕ_{FST}.)*

Conditions on R_1:

2. $R_1^{\leftarrow}(p) = \varnothing$ *(p is not a reset place)*
3. $R_1(u) = \varnothing$ *(u does not reset)*
4. *for all $q \in u\bullet$: $R_1^{\leftarrow}(q) = \varnothing$ (any output place of u is not a reset place)*

Construction of R_2:

5. $R_2 = \{(z, R_1(z)) | z \in T_2 \cap T_1\} \cup \{(v, R_1(t))\}$

We now present two lemmas that show that occurrence sequences in N_1 and N_2 correspond to one another. These lemmas are then used to prove that the ϕ^R_{FST} rule preserves the three criteria of soundness: option to complete, proper completion, and no dead transitions.

Lemma 1 (Under the ϕ^R_{FST} rule, sequences in N_1 correspond to sequences in N_2). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi^R_{\mathrm{FST}}$, let $\sigma_1 \in T_1^*$ and $M_1 \in \boldsymbol{M}(N_1)$ be such that $i \overset{N_1,\sigma_1}{\rightarrow} M_1$, and $\sigma_2 = \alpha(\sigma_1)$, where $\alpha \in T_1^* \rightarrow T_2^*$ is defined as follows:*

- $\alpha(\epsilon) = \epsilon$,
- $\alpha(t\sigma) = v\alpha(\sigma)$,
- $\alpha(u\sigma) = \alpha(\sigma)$, *and*
- $\alpha(x\sigma) = x\alpha(\sigma)$, *where* $x \in T_1 \setminus \{t, u\}$.

Thus, α removes every occurrence of u from the sequence, and replaces every occurrence of t with v. Then $i \overset{N_2,\sigma_2}{\rightarrow} M_2$, where $M_2(x) = M_1(x) + M_1(p)$ for every $x \in v^{\bullet^{N_2}}$ and $M_2(x) = M_1(x)$ for every $x \notin v^{\bullet^{N_2}}$.

Proof By induction on the length of σ_1.

Base Assume $\sigma_1 = \epsilon$. Clearly, $i \overset{N_1,\epsilon}{\rightarrow} i$ and also $i \overset{N_2,\epsilon}{\rightarrow} i$.

Step Assume the theorem holds for some σ_1, let M_1 be such that $i \overset{N_1,\sigma_1}{\rightarrow} M_1$, and let M_2 be such that $i \overset{N_2,\alpha(\sigma_1)}{\rightarrow} M_2$. We prove that it also holds if we extend σ_1 by one transition.

- First, assume that we extend σ by t. t and v have the same preset, thus we can extend $\alpha(\sigma)$ by v. t adds a token to place p, whereas v adds tokens to its postset, which does not violate the where-clause.
- Second, assume that we extend σ by u. It is obvious that v does not violate the where-clause.
- Third, assume that we extend σ by x, where $x \in P_1 \setminus \{t, u\}$. As all places in N_2 contains at least as many tokens as their counterparts in N_1 (the where-clause), we know that x is enabled in N_2 as well. Furthermore, x does not violate the where-clause.

Lemma 2 (Under the ϕ^R_{FST} rule, sequences in N_2 correspond to sequences in N_1). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi^R_{\mathrm{FST}}$, let $\sigma_2 \in T_2^*$ and $M_2 \in \boldsymbol{M}(N_2)$ be such that $i \overset{N_2,\sigma_2}{\rightarrow} M_2$, and $\sigma_1 = \beta(\sigma_2)$, where $\beta \in T_2^* \rightarrow T_1^*$ is defined as follows:*

- $\beta(\epsilon) = \epsilon$,
- $\beta(v\sigma) = tu\beta(\sigma)$, *and*
- $\beta(x\sigma) = x\beta(\sigma)$, *if* $x \in T_2 \setminus \{v\}$.

Thus, β replaces every occurrence of v with tu. Then $i \overset{N_1,\sigma_1}{\rightarrow} M_1$, where $M_1(p) = 0$ and $M_1(x) = M_2(x)$ for every $x \in P_1 \setminus \{p\}$.

Proof By induction on the length of σ_2.

Base Assume $\sigma_2 = \epsilon$. Clearly, $i \overset{N_2,\epsilon}{\rightarrow} i$ and also $i \overset{N_1,\epsilon}{\rightarrow} i$.

Step Assume the theorem holds for some σ_2, let M_2 be such that $i \xrightarrow{N_2,\sigma_2} M_2$, and let M_1 be such that $i \xrightarrow{N_1,\beta(\sigma_2)} M_1$. We prove that it also holds if we extend σ_2 by one transition.

- First, assume that we extend σ by v. It is obvious that $M_1[t\rangle$ in N_1, and that afterwards u is also enabled. Furthermore, the combination tu and v does not violate the where-clause.
- Second, assume that we extend σ by x such that $x \in T_2 \setminus \{v\}$. Again it is obvious that $M_1[x\rangle$ in N_1, and that x does not violate the where clause.

Theorem 2 (The ϕ^R_{FST} rule preserves the option to complete). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi^R_{\mathrm{FST}}$. Then N_1 has the option to complete* iff *N_2 has the option to complete.*

Proof Let α and β be as defined in lemmas 1 and 2.

$\Rightarrow$ Assume that N_2 does not have the option to complete, that is, there exists some $M_2 \in N_2[i\rangle$ such that $o \notin N_2[M_2\rangle$. Thus, there exists a $\sigma_2 \in T_2^*$ such that $i \xrightarrow{N_2,\sigma_2} M_2$ but no $\sigma_2' \in T_2^*$ exists such that $M_2 \xrightarrow{N_2,\sigma_2'} o$. As a result, $i \xrightarrow{N_1,\beta(\sigma_2)} M_1$, for a well-defined M_1. Now assume that N_1 does have the option to complete. As a result, there exists a σ_1 such that $i \xrightarrow{N_1,\beta(\sigma_2)\sigma_1} o$. But then $i \xrightarrow{N_2,\alpha(\beta(\sigma_2)\sigma_1)} o$, which contradicts the assumption that no $\sigma_2' \in T_2^*$ exists such that $M_2 \xrightarrow{N_2,\sigma_2'} o$. Thus, N_1 does not have the option to complete.

$\Leftarrow$ Similar to $\Rightarrow$.

Theorem 3 (The ϕ^R_{FST} rule preserves dead transitions). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi^R_{\mathrm{FST}}$. Then N_1 has proper completion* iff *N_2 has proper completion.*

Proof Let α and β be as defined in lemmas 1 and 2 and the observation that proper completion follows from the option to complete.

$\Rightarrow$ Assume that N_2 contains no dead transitions, that is, for every $t_2 \in T_2$ there exists some $M_2 \in N_2[i\rangle$ such that $M_2 \geq^{N_2}_{\bullet} t$. Let t_2 be an arbitrary transition from T_2, and let $M_2 \in N_2[i\rangle$ be such that $M_2 \geq^{N_2}_{\bullet} t_2$. Then there exists a $\sigma_2 \in T_2^*$ such that $i \xrightarrow{N_2,\sigma_2} M_2$. As a result, $i \xrightarrow{N_1,\beta(\sigma_2)} M_1$ and $M_1 \geq^{N_1}_{\bullet} t_2$. As $T_2 = T_1 \cup \{t\}$, only transition t can still be dead. However, t can only be dead if all transitions that mark p are dead, and these transitions exist (as $p \neq i$).

$\Leftarrow$ Assume that N_1 contains no dead transitions, that is, for every $t_1 \in T_1$ there exists some $M_1 \in N_1[i\rangle$ such that $M_1 \geq^{N_1}_{\bullet} t$. Let t_1 be an arbitrary transition from T_1 excluding t, and let $M_1 \in N_1[i\rangle$ be such that $M_1 \geq^{N_1}_{\bullet} t_1$. Then there exists a $\sigma_1 \in T_1^*$ such that $i \xrightarrow{N_1,\sigma_1} M_1$. As a result, $i \xrightarrow{N_2,\alpha(\sigma_1)} M_2$ and $M_2 \geq^{N_2}_{\bullet} t_1$. Thus, N_2 contains no dead transitions.

Theorem 4 (The ϕ^R_{FST} rule is soundness preserving). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi^R_{\mathrm{FST}}$. N_1 is sound iff N_2 is sound.*

Proof Follows from theorems 2 and 3.

In this section, we have shown how the ϕ^R_{FST} rule is derived by first listing out the application conditions for a WF-net and then proposing additional conditions to deal with reset arcs in an RWF-net. The ϕ^R_{FST} rule has been shown to be soundness preserving by providing the proofs that the rule preserves the criteria for soundness.

4.2 Fusion of series places

The *Fusion of Series Places Rule for WF-nets* (ϕ_{FSP}) allows for the merging of two sequential places p and q with one transition t in between them into a single place r. The rule requires that there is only one output arc from p to t, exactly one input p and one output q for t, and that there are no direct connections between inputs of p and inputs of q. The last requirement ensures that there will only be one arc connecting inputs of p in the original net to the new place r in the reduced net (no weighted arcs). Furthermore, the rule is not applicable to places that are either an input place i or an output place o of the net. See the example in Figure 10 for an application of the ϕ_{FSP} rule.

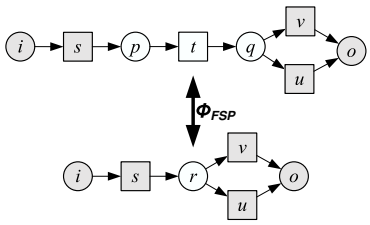

Fig. 10. Reduction of a WF-net using the ϕ_{FSP} rule

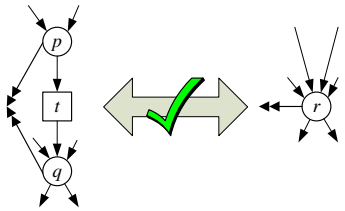

Fig. 11. Fusion of Series Places Rule for RWF-nets: ϕ^R_{FSP}

Definition 7 (Fusion of Series Places Rule for WF-nets: ϕ_{FSP}). *Let N_1 and N_2 be two WF-nets, where $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$. $(N_1, N_2) \in \phi_{\mathrm{FSP}}$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, two places $p, q \in P_1 \setminus \{i, o\}$, a transition $t \in T_1$, and a place $r \in P_2 \setminus P_1$ such that:*

Conditions on N_1:

1. $\bullet t = \{p\}$ *(p is the only input of t)*
2. $t\bullet = \{q\}$ *(q is the only output of t)*
3. $p\bullet = \{t\}$ *(t is the only output of p)*
4. $\bullet p \cap \bullet q = \varnothing$ *(any input of p is not an input of q and vice versa)*

Construction of N_2:

5. $P_2 = (P_1 \setminus \{p, q\}) \cup \{r\}$
6. $T_2 = T_1 \setminus \{t\}$
7. $F_2 = (F_1 \cap ((P_2 \times T_2) \cup (T_2 \times P_2))) \cup (((\,^{N_1}\!\bullet p \cup \,^{N_1}\!\bullet q) \setminus \{t\}) \times \{r\}) \cup (\{r\} \times q^{\,N_1}\!\bullet)$

The *Fusion of Series Places Rule for RWF-nets* (ϕ^R_{FSP}) extends the ϕ_{FSP} rule by introducing reset arcs and strengthening the conditions. The rule also allows for the merging of two sequential places p and q with one transition t in between them into a single place r. Figure 11 visualises the ϕ^R_{FSP} rule. The first additional requirement is that the transition t should not have any reset arcs. See Figure 12(a) for a counter-example where t has reset arcs. Transition t can reset place u in the left net but this behaviour is ignored in the right net. Transition sequence xt leads to a deadlock as t will remove a token from u when it fires, and u does not exist in the right net. As a result, the left net is not sound whereas the right net is. The second additional requirement is that the two places must be reset by the same set of transitions (if any). If p and q are not reset places, then it is clear that the rule holds. If a transition resets place p, it must also resets place q as we are interested in merging these two places. See Figure 12(b) for a counter-example: transition sequence $xtyz$ leads to an unsound net on the left (a leftover token in q), whereas the right net is sound. If all requirements for the ϕ^R_{FST} rule are satisfied, places p and q are merged into a new place r which takes on the same reset arcs as p and q.

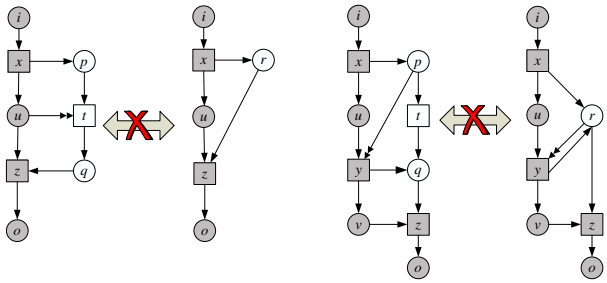

Fig. 12. counter-examples for the ϕ_{FSP} rule applied to reset nets

Definition 8 (Fusion of Series Places Rule for RWF-nets: ϕ^R_{FSP}). *Let N_1 and N_2 be two RWF-nets, where $N_1 = (P_1, T_1, F_1, R_1)$ and $N_2 = (P_2, T_2, F_2, R_2)$. $(N_1, N_2) \in \phi^R_{\mathrm{FSP}}$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, two places $p, q \in P_1 \setminus \{i, o\}$, a transition $t \in T_1$, and a place $r \in P_2 \setminus P_1$ such that:*

Extension of the ϕ_{FSP} rule:

1. $((P_1, T_1, F_1), (P_2, T_2, F_2)) \in \phi_{\mathrm{FSP}}$ *(Note that, by definition, the i, o, p, q, t, and r mentioned in this definition have to coincide with the i, o, p, q, t, and r as mentioned in the definition of ϕ_{FSP}.)*

Conditions on R_1:

2. $R_1(t) = \varnothing$ *(t does not reset)*
3. $R_1^{\leftarrow}(p) = R_1^{\leftarrow}(q)$ *(p and q are being reset by the same transitions)*

Construction of R_2:

4. $R_2 = \{(z, R_1(z) \cap P_2) | z \in T_2 \cap T_1\} \oplus \{(z, (R_1(z) \cap P_2) \cup \{r\}) | z \in R_1^{\leftarrow}(p)\}$[3].

The ϕ^R_{FSP} rule is soundness preserving as the occurrence sequences in the original net and the reduced net correspond to each other. The proof is similar to the soundness proof given for ϕ^R_{FST} rule.

Next, we show that the ϕ^R_{FSP} rule is soundness preserving. We first present two lemmas that show that occurrence sequences in N_1 and N_2 correspond to one another. These lemmas are then used to prove that the ϕ^R_{FSP} rule preserves the three criteria of soundness: the option to complete, proper completion, and dead transitions.

Lemma 3 (Under the ϕ^R_{FSP} rule, sequences in N_1 correspond to sequences in N_2). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi^R_{\mathrm{FSP}}$, let $\sigma_1 \in T_1^*$ and $M_1 \in \boldsymbol{M}(N_1)$ be such that $i \overset{N_1,\sigma_1}{\rightarrow} M_1$, and $\sigma_2 = \alpha(\sigma_1)$, where $\alpha \in T_1^* \rightarrow T_2^*$ is defined as follows: $\alpha(\epsilon) = \epsilon$, $\alpha(t\sigma) = \alpha(\sigma)$, and $\alpha(x\sigma) = x\alpha(\sigma)$, where $x \in T_1 \setminus \{t\}$. Thus, α removes every occurrence of t from the sequence. Then $i \overset{N_2,\sigma_2}{\rightarrow} M_2$, where $M_2(r) = M_1(p) + M_1(q)$ and $M_2(x) = M_1(x)$ for every $x \in P_2 \setminus \{r\}$.*

Proof By induction on the length of σ_1.

Base Assume $\sigma_1 = \epsilon$. Clearly, $i \overset{N_1,\epsilon}{\rightarrow} i$ and also $i \overset{N_2,\epsilon}{\rightarrow} i$.

Step Assume the theorem holds for some σ_1, let M_1 be such that $i \overset{N_1,\sigma_1}{\rightarrow} M_1$, and let M_2 be such that $i \overset{N_2,\alpha(\sigma_1)}{\rightarrow} M_2$. We prove that it also holds if we extend σ_1 by one transition.

First, assume that we extend σ by t. It is easy to see that this extension does not have any effect on $\alpha(\sigma_1)$. Therefore, we need to prove that firing t does not violate the where-clause (i.e, $M_2(r) = M_1(p) + M_1(q)$ and $M_2(x) = M_1(x)$ for every $x \in P_2 \setminus \{r\}$). As t moves only one token from p to q and does not reset any place, this is straightforward.

Second, assume that we extend σ by an $x \in P_1 \setminus \{t\}$. First, we need to prove that $M_2[x\rangle$ in N_2. As r contains at least as many tokens as q, and $M_2(x) = M_1(x)$ for every $x \in P_2 \setminus \{r\}$, we conclude that this is indeed the case. Next, we need to prove that firing x in both nets does not violate the where-clause. This is straightforward as well, as any transition that adds a token to p also adds a token to r and any transition that removes a token from q also removes a token from r, and the remaining transitions are identical.

[3] $\oplus$ represents function override where $f : A \rightarrow B$ and $g : A \rightarrow B$, $f \oplus g = \{(a, b) | a \in dom(g)\} \cup \{(a, b) | a \in dom(f) \setminus dom(g)\}$.

Lemma 4 (Under the ϕ^R_{FSP} rule, sequences in N_2 correspond to sequences in N_1). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi^R_{\mathrm{FSP}}$, let $\sigma_2 \in T_2^*$ and $M_2 \in \boldsymbol{M}(N_2)$ be such that $i \xrightarrow{N_2,\sigma_2} M_2$, and $\sigma_1 = \beta(\sigma_2)$, where $\beta \in T_2^* \to T_1^*$ is defined as follows: $\beta(\epsilon) = \epsilon$, $\beta(x\sigma) = xt\beta(\sigma)$, if $p \in x \overset{N_1}{\bullet}$, and $\beta(x\sigma) = x\beta(\sigma)$, if $p \notin x \overset{N_1}{\bullet}$. Thus, β introduces an extra t whenever place p is marked. As a result, place p is unmarked as soon as possible. Then $i \xrightarrow{N_1,\sigma_1} M_1$, where $M_1(p) = 0$, $M_1(q) = M_2(r)$ and $M_1(x) = M_2(x)$ for every $x \in P_1 \setminus \{p, q\}$.*

Proof By induction on the length of σ_2.

Base Assume $\sigma_2 = \epsilon$. Clearly, $i \xrightarrow{N_2,\epsilon} i$ and also $i \xrightarrow{N_1,\epsilon} i$.

Step Assume the theorem holds for some σ_2, let M_2 be such that $i \xrightarrow{N_2,\sigma_2} M_2$, and let M_1 be such that $i \xrightarrow{N_1,\beta(\sigma_2)} M_1$. We prove that it also holds if we extend σ_2 by one transition.

First, assume that we extend σ by an x such that $p \in x \overset{N_1}{\bullet}$. It is obvious that $M_1[x\rangle$ in N_1, and that afterwards t is also enabled. Furthermore, both x and t do not violate the where-clause (i.e., where $M_1(p) = 0$, $M_1(q) = M_2(r)$ and $M_1(x) = M_2(x)$ for every $x \in P_1 \setminus \{p, q\}$.

Second, assume that we extend σ by an x such that $p \notin x \overset{N_1}{\bullet}$. Again it is obvious that $M_1[x\rangle$ in N_1, and that x does not violate the where clause.

Theorem 5 (The ϕ^R_{FSP} rule is soundness preserving). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi^R_{\mathrm{FSP}}$. N_1 is sound iff N_2 is sound.*

Proof Let α and β be as defined in lemmas 3 and 4.

- The ϕ^R_{FSP} rule preserves the option to complete. The proof is similar to the proof of Theorem 2, but with different α and β.
- The ϕ^R_{FSP} rule preserves dead transitions. The proof is similar to the proof of Theorem 3, but with different α and β.

4.3 Fusion of parallel places

The *Fusion of Parallel Places Rule for WF-nets* (ϕ_{FPP}) is a generalization of the Fusion of Parallel Places rule for Petri nets by Murata [13]. The rule allows for the merging of multiple places (at least two) with the same inputs and outputs into a single place q. See the example in Figure 13 for an application of the ϕ_{FPP} rule. Places p_1 and p_2 have the same input set $\{t_1, t_2, t_3\}$ and the same output set $\{x_1, x_2\}$. The reduced net contains a new place q that has the same input and output sets as places p_1 and p_2.

Definition 9 (Fusion of Parallel Places Rule for WF-nets: ϕ_{FPP}).
Let N_1 and N_2 be two WF-nets, where $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$. $(N_1, N_2) \in \phi_{\mathrm{FPP}}$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, places $Q \subseteq P_1$ where $|Q| \geq 2$ and a place $q \in P_2 \setminus P_1$ such that:

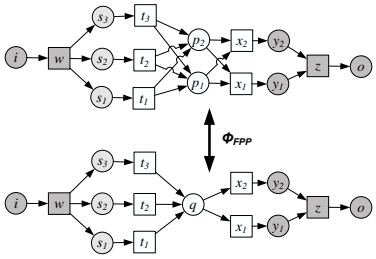

Fig. 13. Reduction of a WF-net using the ϕ_{FPP} rule

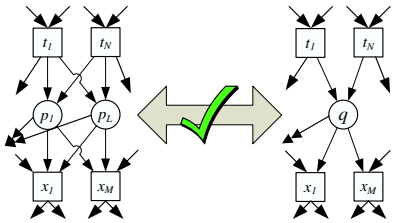

Fig. 14. Fusion of Parallel Places Rule for RWF-nets: ϕ^{R}_{FPP}

Conditions on N_1:

1. *for all $px, py \in Q : {\bullet}px = {\bullet}py$ (input transitions for all places in Q are identical)*
2. *for all $px, py \in Q : px{\bullet} = py{\bullet}$ (output transitions for all places in Q are identical)*

Construction of N_2:

3. $P_2 = (P_1 \setminus Q) \cup \{q\}$
4. $T_2 = T_1$
5. $F_2 = (F_1 \cap ((P_2 \times T_2) \cup (T_2 \times P_2))) \cup ({}^{N_1}_{\bullet}p \times \{q\}) \cup (\{q\} \times p^{N_1}_{\bullet})$ *where* $p \in Q$

The *Fusion of Parallel Places Rule for RWF-nets* (ϕ^{R}_{FPP}) extends the ϕ_{FPP} rule by introducing reset arcs. The rule also allows for the merging of places in Q (i.e., p_1 to p_L) that have the same inputs and outputs into a single place q. The additional requirement is that these places are reset by the same set of transitions. If none of the places are reset places, then it is obvious that the rule holds. If one is a reset place, then other places should also be reset by the same set of transitions. Figure 14 visualises the ϕ^{R}_{FPP} rule. As all places in $Q = \{p_1, ..., p_L\}$ have the same input, output and reset arcs, these identical places can be merged into a single place while preserving the soundness property. Place q in the reduced net has the same input, output and reset arcs as any place in Q.

Definition 10 (Fusion of Parallel Places Rule for RWF-nets: ϕ^{R}_{FPP}). *Let N_1 and N_2 be two RWF-nets, where $N_1 = (P_1, T_1, F_1, R_1)$ and $N_2 = (P_2, T_2, F_2, R_2)$. $(N_1, N_2) \in \phi^{R}_{\mathrm{FPP}}$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, places $Q \subseteq P_1$ where $|Q| \geq 2$ and a place $q \in P_2 \setminus P_1$ such that:*

Extension of the ϕ_{FPP} rule:

1. $((P_1, T_1, F_1), (P_2, T_2, F_2)) \in \phi_{\mathrm{FPP}}$ *(Note that, by definition, the i, o, Q, and q mentioned in this definition have to coincide with the i, o, Q, and q as mentioned in the definition of ϕ_{FPP}.)*

Condition on R_1:

2. *for all $px, py \in Q : R_1^{\leftarrow}(px) = R_1^{\leftarrow}(py)$ (all places in Q are being reset by the same transitions)*

Construction of R_2:

3. $R_2 = \{(z, R_1(z) \cap P_2) | z \in T_2 \cap T_1\} \oplus \{(z, (R_1(z) \cap P_2) \cup \{q\}) | z \in R_1^{\leftarrow}(p) \wedge p \in Q\}$

Theorem 6 (The ϕ^R_{FPP} rule is soundness preserving). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi^R_{\mathrm{FPP}}$. N_1 is sound iff N_2 is sound.*

Proof It is easy to see that the state spaces of both nets are identical, except that the markings differ: A marking in the state space of N_1 contains places Q, and every one of them contains n tokens, whereas a marking in the state space of N_2 contains one place q which contains n tokens.

4.4 Fusion of parallel transitions

The *Fusion of Parallel Transitions Rule for WF-nets* (ϕ_{FPT}) is a generalization of the Fusion of Parallel Transitions rule for Petri nets by Murata [13]. The rule allows for the merging of multiple transitions (at least two) that have the same inputs and outputs into a single transition. See the example in Figure 15 for an application of the ϕ_{FPT} rule. Transitions t_1 and t_2 have the same input set $\{p_1, p_2, p_3\}$ and the same output set $\{x_1, x_2\}$. The reduced net contains a new transition v that has the same input and output sets as t_1 and t_2.

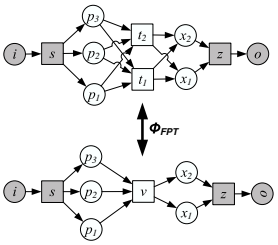

Fig. 15. Reduction of a WF-net using the ϕ_{FPT} rule

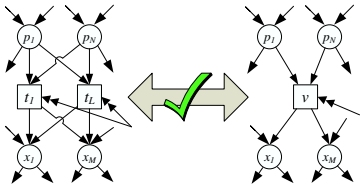

Fig. 16. Fusion of Parallel Transitions Rule for RWF-nets: ϕ^R_{FPT}

Definition 11 (Fusion of Parallel Transitions Rule for WF-nets: ϕ_{FPT}). *Let N_1 and N_2 be two WF-nets, where $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$. $(N_1, N_2) \in \phi_{\mathrm{FPT}}$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, transitions $V \subseteq T_1$ where $|V| \geq 2$, and a transition $v \in T_2 \setminus T_1$ such that:*

Conditions on N_1:

1. *for all $tx, ty \in V : \bullet tx = \bullet ty$(input places for all transitions in V are identical)*
2. *for all $tx, ty \in V : tx\bullet = ty\bullet$(output places for all transitions in V are identical)*

Construction of N_2:

3. $P_2 = P_1$
4. $T_2 = (T_1 \setminus V) \cup \{v\}$
5. $F_2 = (F_1 \cap ((P_2 \times T_2) \cup (T_2 \times P_2))) \cup (\{v\} \times {}^{N_1}\bullet t) \cup (t {}^{N_1}\bullet \times \{v\})$ *where* $t \in V$

The *Fusion of Parallel Transitions Rule for RWF-nets* (ϕ^R_{FPT}) extends the ϕ_{FPT} rule by introducing reset arcs. The rule allows for the merging of transitions V (i.e., t_1 to t_L) that have the same inputs and outputs into a single transition v. The additional requirement is that these transitions should reset the same set of places (if any). If no transition has reset arcs, then it is obvious that the rule holds. If one transition resets a place, then other transitions must also reset the same place. Figure 16 visualises the ϕ^R_{FPT} rule. As all transitions in $V = \{t_1, ..., t_L\}$ now have the same input, output and reset arcs, these identical transitions could be merged into a single transition while preserving the soundness property. Transition v in the reduced net has the same input, output and reset arcs as any transition $t \in V$.

Definition 12 (Fusion of Parallel Transitions Rule for RWF-nets: ϕ^R_{FPT}). *Let N_1 and N_2 be two RWF-nets, where $N_1 = (P_1, T_1, F_1, R_1)$ and $N_2 = (P_2, T_2, F_2, R_2)$. $(N_1, N_2) \in \phi^R_{\mathrm{FPT}}$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, transitions $V \subseteq T_1$ where $|V| \geq 2$, and a transition $v \in T_2 \setminus T_1$ such that:*

Extension of the ϕ_{FPT} rule:

1. $((P_1, T_1, F_1), (P_2, T_2, F_2)) \in \phi_{\mathrm{FPT}}$ *(Note that, by definition, the i, o, V, and v mentioned in this definition have to coincide with the i, o, V, and v as mentioned in the definition of ϕ_{FPT}.)*

Condition on R_1:

2. *for all $tx, ty \in V : R_1(tx) = R_1(ty)$ (all transitions in V reset the same places)*

Construction of R_2:

3. $R_2 = \{(z, R_1(z)) | z \in T_2 \cap T_1\} \cup \{(v, R_1(t))\}$, *where* $t \in V$

Theorem 7 (The ϕ^R_{FPT} rule is soundness preserving.). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi^R_{\mathrm{FPT}}$. N_1 is sound iff N_2 is sound.*

Proof It is obvious that the state spaces of both nets are identical, except that some edges differ: where the state space of N_1 contains edges for transitions t_1 up to t_L, the state space of N_2 only contains one edge for transition v.

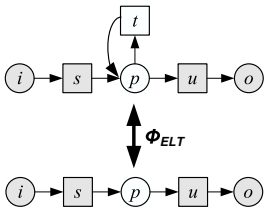

Fig. 17. Reduction of a WF-net using the ϕ_{ELT} rule

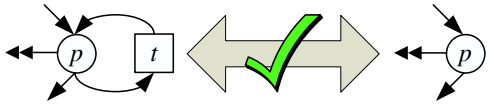

Fig. 18. Elimination of Self-Loop Transitions Rule for RWF-nets: ϕ^R_{ELT}

4.5 Elimination of self-loop transitions

The *Elimination of Self-Loop Transitions Rule for WF-nets* (ϕ_{ELT}) rule allows the removal of a self-loop transition. A self-loop transition is one that has one input place which is also the only output place of the transition. See the example in Figure 17 for an application of the ϕ_{ELT} rule. Transition t has been abstracted from in the reduced net as p is the only input place and the only output place of t.

Definition 13 (Elimination of Self-Loop Transitions for WF-nets: ϕ_{ELT}). *Let N_1 and N_2 be two WF-nets, where $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$. $(N_1, N_2) \in \phi_{\mathrm{ELT}}$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, a place $p \in P_1 \cap P_2$, and a transition $t \in T_1$ such that:*

Conditions on N_1:

1. $\bullet t = \{p\}$ *(p is the only input place of t)*
2. $t\bullet = \{p\}$ *(p is the only output place of t)*

Construction of N_2:

3. $P_2 = P_1$
4. $T_2 = T_1 \setminus \{t\}$
5. $F_2 = (F_1 \cap ((P_2 \times T_2) \cup (T_2 \times P_2)))$

The *Elimination of Self-Loop Transitions Rule for RWF-nets* (ϕ^R_{ELT}) extends the ϕ_{ELT} rule by introducing reset arcs. The rule also allows removal of a transition t which has a single place as its input and its output. The additional requirement is that transition t has no reset arcs. Figure 18 visualises the ϕ^R_{ELT} rule.

Definition 14 (Elimination of Self-Loop Transitions Rule for RWF-nets: ϕ^R_{ELT}). *Let N_1 and N_2 be two RWF-nets, where $N_1 = (P_1, T_1, F_1, R_1)$ and $N_2 = (P_2, T_2, F_2, R_2)$. $(N_1, N_2) \in \phi^R_{\mathrm{ELT}}$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, a place $p \in P_1 \cap P_2$, and a transition $t \in T_1$ such that:*

Extension of the ϕ_{ELT} rule:

1. $((P_1, T_1, F_1), (P_2, T_2, F_2)) \in \phi_{\mathrm{ELT}}$ *(Note that, by definition, the i, o, t, and p mentioned in this definition have to coincide with the i, o, t, and p as mentioned in the definition of ϕ_{ELT}.)*

Condition on R_1:

2. $R_1(t) = \varnothing$ *(t does not reset)*

Construction of R_2:

3. $R_2 = \{(z, R_1(z)) | z \in T_2 \cap T_1\}$

Theorem 8 (The ϕ^R_{ELT} rule is soundness preserving). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi^R_{\mathrm{ELT}}$. N_1 is sound iff N_2 is sound.*

Proof It is obvious that the state spaces of both nets are identical, except that the state space of N_1 contains additional self-edges. Furthermore, it is clear that t can only be dead if every transition that marks p is dead. Therefore, removing t preserves dead transitions.

We have presented five reduction rules for RWF-nets based on the reduction rules defined by Murata [13]. We have omitted the sixth rule, "Elimination of Self-Loop Places" as this rule requires a place to be marked in an initial marking of a net. For WF-nets and RWF-nets, this is not possible as the input place i is the only place that could be marked in an initial marking. By definition, i cannot be a self-loop (i.e., it cannot have any incoming arcs $\bullet i = \varnothing$) and therefore, this rule is not applicable to WF-nets and RWF-nets. In addition to the "Murata rules" we also present two additional rules.

4.6 Abstraction

The *Abstraction Rule for WF-nets* (ϕ_{A}) is based on the Abstraction rule from Desel and Esparza [6]. The rule allows the removal of a place s and a transition t, where s is the only input of t, t is the only output of s and there is no direct connection between the inputs of s with the outputs of t. See the example in Figure 19 for an application of the ϕ_{A} rule. The reduced net abstracts from place s and transition t and provides direct connections between the inputs of s and the outputs of t.

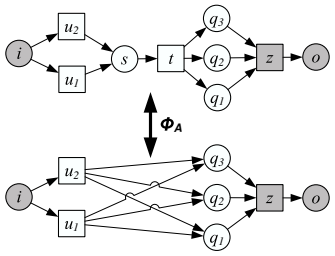

Fig. 19. Reduction of a WF-net using the ϕ_{A} rule

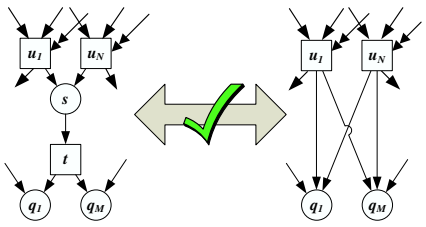

Fig. 20. Abstraction Rule for RWF-nets: ϕ^R_{A}

Definition 15 (Abstraction Rule for WF-nets: ϕ_{A}). *Let N_1 and N_2 be two WF-nets, where $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$. $(N_1, N_2) \in \phi_{\mathrm{A}}$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, places $Q \subseteq P_1 \cap P_2$, a place $s \in P_1 \setminus Q$, transitions $U \subseteq T_1 \cap T_2$, and a transition $t \in T_1 \setminus U$ such that:*

Conditions on N_1:

1. $\bullet t = \{s\}$ *(s is the only input of t)*
2. $s\bullet = \{t\}$ *(t is the only output of s)*
3. $\bullet s = U$ *(transitions in U are input transitions for s)*
4. $t\bullet = Q$ *(transitions in Q are output transitions for t)*
5. $(\bullet s \times t\bullet) \cap F = \varnothing$ *(any input of s is not connected to an output of t and vice versa)*

Construction of N_2:

6. $P_2 = P_1 \setminus \{s\}$
7. $T_2 = T_1 \setminus \{t\}$
8. $F_2 = (F_1 \cap ((P_2 \times T_2) \cup (T_2 \times P_2))) \cup ({}^{N_1}\!\bullet s \times t\,{}^{N_2}\!\bullet)$

The *Abstraction Rule for RWF-nets* (ϕ_{A}^R) extends the ϕ_{A} rule by introducing reset arcs. The rule allows for the removal of a place s and a transition t, where s is the only input of t, t is the only output of s and there is no direct connection between the inputs for s with the outputs for t. Additional requirements are that transition t does not reset any place, place s is not reset by any transition, and outputs for t are not reset by any transition. Input transitions for place s can have reset arcs. Figure 20 visualises the ϕ_{A}^R rule.

Definition 16 (Abstraction Rule for RWF-nets: ϕ_{A}^R). *Let N_1 and N_2 be two RWF-nets, where $N_1 = (P_1, T_1, F_1, R_1)$ and $N_2 = (P_2, T_2, F_2, R_2)$. $(N_1, N_2) \in \phi_{\mathrm{A}}^R$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, places $Q \subseteq P_1 \cap P_2$, a place $s \in P_1 \setminus Q$, transitions $U \subseteq T_1 \cap T_2$, and a transition $t \in T_1 \setminus U$ such that:*

Extension of the ϕ_{A}^R rule:

1. $((P_1, T_1, F_1), (P_2, T_2, F_2)) \in \phi_{\mathrm{A}}$ *(Note that, by definition, the i, o, s, t, Q, and U mentioned in this definition have to coincide with i, o, s, t, Q, and U as mentioned in the definition of ϕ_{A}.)*

Conditions on R_1:

2. $R_1^{\leftarrow}(s) = \varnothing$ *(s is not a reset place)*
3. $R_1(t) = \varnothing$ *(t does not reset)*
4. *for all $q \in t\bullet$: $R_1^{\leftarrow}(q) = \varnothing$ (all output places for t are not reset places)*

Construction of R_2:

5. $R_2 = \{(z, R_1(z) \cap P_2) | z \in T_2 \cap T_1\}$

Theorem 9 (The ϕ_{A}^R rule is soundness preserving). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi_{\mathrm{A}}^R$. N_1 is sound iff N_2 is sound.*

Proof This rule is quite close to the ϕ_{FST}^R rule (i.e., the fusion of two subsequent transitions), except that it allows for s (p for the ϕ_{FST}^R rule) to have multiple inputs. Using the ϕ_{FST}^R rule, the proof is quite simple. It is obvious that we can replace s and t by $s_1, ..., s_N$ and $t_1, ..., t_N$ in such a way that $\bullet s_i = \{u_i\}$, $s_i\bullet = \{t_i\}$, $\bullet t_i = \{s_i\}$, and $t_i\bullet = Q$ while preserving soundness. Next, we can use the ϕ_{FST}^R rule to reduce every s_i and t_i. Figure 21 visualises the proof of the ϕ_{A}^R rule.

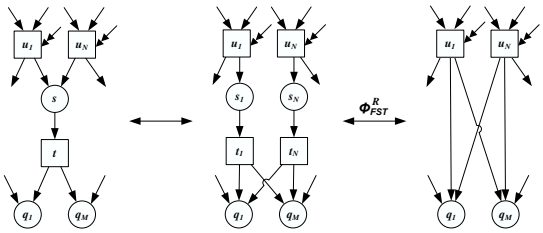

Fig. 21. Proof sketch for the ϕ_{A}^{R} rule

The other two linear dependency rules described by Desel and Esparza [6] to remove nonnegative linearly dependent places and nonnegative linearly dependent transitions are only applicable to free-choice nets. The rules are said to be not strongly sound for arbitrary nets (i.e., N is well-formed if and only if N' is well-formed) [6]. Hence, they cannot be used for WF-nets and RWF-nets.

4.7 Fusion of equivalent subnets

The *Fusion of Equivalent Subnets Rule for WF-nets* (ϕ_{FES}) allows removal of multiple identical subnets by replacing them with only one subnet. The rule requires that pairs of transitions have the same input and output places. See the example in Figure 22 for an application of the ϕ_{FES} rule. The set of transitions V_1 has been merged into V_3. The set of transitions V_2 has been merged into V_4, and places in Q_2 have been merged into one place r. The *Fusion of Equivalent Subnets Rule for RWF-nets* (ϕ_{FES}) extends the ϕ_{FES} rule by introducing reset arcs. Figure 23 visualises the ϕ_{FES}^{R} rule. Additional requirements are that all places in Q_2 are reset by the same set of transitions and all transition pairs in V_1 and V_3 also reset the same places. Formal definitions of the ϕ_{FES} rule and the ϕ_{FES}^{R} rule follow the same structure as the other rules. Note that the name of the rule may be a bit misleading. This rule only applies to subnets having the structure shown in Figure 22. The reason that this rule has been added is that it is very effective in reducing YAWL models (cf. [16]).

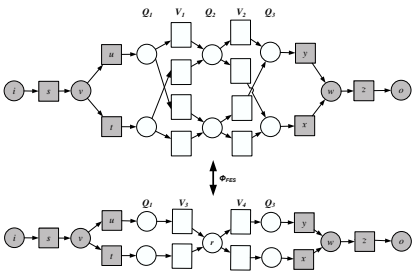

Fig. 22. Reduction of a WF-net using the ϕ_{FES} rule

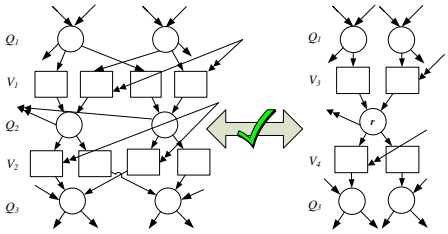

Fig. 23. Fusion of Equivalent Subnets Rule for RWF-nets: ϕ_{FES}^{R}

Definition 17 (Fusion of Equivalent Subnets Rule for WF-nets: ϕ_{FES}). *Let N_1 and N_2 be two WF-nets, where $N_1 = (P_1, T_1, F_1)$ and $N_2 = (P_2, T_2, F_2)$. $(N_1, N_2) \in \phi_{\text{FES}}$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, places $Q_1, Q_3 \subseteq P_1 \cap P_2$, $Q_2 \subseteq P_1$ where $|Q_2| \geq 2$, $r \in P_2 \setminus P_1$, transitions $V_1, V_2 \subseteq T_1$, and $V_3, V_4 \subseteq T_2 \setminus T_1$ such that:*

Conditions on N_1:

1. $V_1 = \{v_1^{q_1,q_2} | q_1 \in Q_1 \wedge q_2 \in Q_2\}$ *(every transition of V_1 is of the form $v_1^{q_1,q_2}$)*
2. $V_2 = \{v_2^{q_2,q_3} | q_2 \in Q_2 \wedge q_3 \in Q_3\}$ *(every transition of V_2 is of the form $v_2^{q_2,q_3}$)*
3. *for all* $p \in Q_2 : {}^{N_1}\bullet p \subseteq V_1 \wedge p {}^{N_1}\bullet \subseteq V_2$ *(preset and postset of all places in Q_2 are from V_1 and V_2 respectively)*
4. *for all* $v_1^{q_1,q_2} \in V_1 : {}^{N_1}\bullet v_1^{q_1,q_2} = \{q_1\} \wedge v_1^{q_1,q_2} {}^{N_1}\bullet = \{q_2\}$ *(preset of $v_1^{q_1,q_2}$ is q_1 and postset is q_2)*
5. *for all* $v_2^{q_2,q_3} \in V_2 : {}^{N_1}\bullet v_2^{q_2,q_3} = \{q_2\} \wedge v_2^{q_2,q_3} {}^{N_1}\bullet = \{q_3\}$ *(preset of $v_2^{q_2,q_3}$ is q_2 and postset is q_3)*

Construction of N_2:

6. $P_2 = (P_1 \setminus Q_2) \cup \{r\}$
7. $T_2 = (T_1 \setminus (V_1 \cup V_2)) \cup (V_3 \cup V_4)$ *where* $V_3 = \{v_3^{q_1,r} | q_1 \in Q_1\}$ *and* $V_4 = \{v_4^{r,q_3} | q_3 \in Q_3\}$
8. $F_2 = (F_1 \cap ((P_2 \times T_2) \cup (T_2 \times P_2))) \cup (V_3 \times \{r\}) \cup (\{r\} \times V_4) \cup \{(q_1, v_3^{q_1,r}) | q_1 \in Q_1 \wedge v_3^{q_1,r} \in V_3\} \cup \{(v_4^{r,q_3}, q_3) | q_3 \in Q_3 \wedge v_4^{r,q_3} \in V_4\}$

Theorem 10 (The ϕ_{FES} rule is soundness preserving). *Let N_1 and N_2 be two WF-nets such that $(N_1, N_2) \in \phi_{\text{FES}}$. N_1 is sound iff N_2 is sound.*

Proof The state spaces of both nets are comparable, such that where the state space of N_1 contains edges for transitions in V_1, the state space of N_2 only contains edges for transitions in V_3. Similarly, the set of transitions V_2 in N_1 is now V_4 in N_2. The set of places Q_2 has been replaced with r.

FESRThe *Fusion of Equivalent Subnets Rule for RWF-nets* (ϕ_{FES}) extends the ϕ_{FES} rule by introducing reset arcs. The rule allows the removal of multiple identical subnets by replacing them with only one subnet. Additional requirements are that all places in Q_2 are reset by the same set of transitions and all transition pairs in V_1 and V_3 also reset the same places. Figure 23 visualises the ϕ_{FES}^R rule.

Definition 18 (Fusion of Equivalent Subnets Rule for RWF-nets: ϕ_{FES}^R). *Let N_1 and N_2 be two RWF-nets, where $N_1 = (P_1, T_1, F_1, R_1)$ and $N_2 = (P_2, T_2, F_2, R_2)$. $(N_1, N_2) \in \phi_{\text{FES}}^R$ if there exists an input place $i \in P_1 \cap P_2$, an output place $o \in P_1 \cap P_2$, places $Q_1, Q_3 \subseteq P_1 \cap P_2$, $Q_2 \subseteq P_1$ where $|Q_2| \geq 2$, $r \in P_2 \setminus P_1$, transitions $V_1, V_2 \subseteq T_1$, and $V_3, V_4 \subseteq T_2 \setminus T_1$ such that:*

Extension of the ϕ_{FES} rule:

1. $((P_1, T_1, F_1), (P_2, T_2, F_2)) \in \phi_{\text{FES}}$ *(Note that, by definition, the i, o, Q_1, Q_2, Q_3, V_1, V_2, V_3, and V_4 mentioned in this definition have to coincide with the i, o, Q_1, Q_2, Q_3, V_1, V_2, V_3, and V_4 as mentioned in the definition of ϕ_{FES}.)*

Condition on R_1:

2. *for all $q_1 \in Q_1$ and $q_2, q_2' \in Q_2 : R(v_1^{q_1,q_2}) = R(v_1^{q_1,q_2'})$ (transitions in V_1 that have the same input set should also have the same reset arcs)*
3. *for all $q_3 \in Q_3$ and $q_2, q_2' \in Q_2 : R(v_1^{q_2,q_3}) = R(v_1^{q_2',q_3})$ (transitions in V_2 that have the same output set should also have the same reset arcs)*
4. *for all $q_2, q_2' \in Q_2$, $R_1^{\leftarrow}(q_2) = R_1^{\leftarrow}(q_2')$ (places in Q_2 are reset by the same set of transitions)*

Construction of R_2:

5. $R_2 = \{(z, R_1(z) \cap P_2) | z \in T_2 \cap T_1\}$
   $\oplus\{(z, (R_1(z) \cap P_2 \cup \{r\})) | z \in R_1^{\leftarrow}(q_2) \wedge q_2 \in Q_2\}$
   $\cup\{(v_3^{q_1,r}, R_1(v_1^{q_1,q_2}) \cap P_2) | q_1 \in Q_1 \wedge q_2 \in Q_2\}$
   $\cup\{(v_4^{r,q_3}, R_1(v_2^{q_2,q_3}) \cap P_2) | q_2 \in Q_2 \wedge q_3 \in Q_3\}$

Theorem 11 (The ϕ_{FES}^R rule is soundness preserving). *Let N_1 and N_2 be two RWF-nets such that $(N_1, N_2) \in \phi_{\mathrm{FES}}^R$. N_1 is sound iff N_2 is sound.*

Proof The proof is similar to the one for the ϕ_{FES} rule. The state spaces of both nets are comparable, such that where the state space of N_1 contains edges for transitions in V_1, the state space of N_2 only contains edges for transitions in V_3. Similarly, the set of transitions V_2 in N_1 is now V_4 in N_2. The set of places Q_2 has been replaced with r. Additional requirements for reset arcs ensure that the transitions can be abstracted.

In this section, seven reduction rules for RWF-nets have been presented. Please note that we do not claim the set of rules to be complete. We have shown that transitions with reset arcs or places that can be reset cannot be entirely abstracted from the net. Hence, it is not possible to reduce an RWF-net from a large net to a trivial net with just a source node, a single transition and a sink node even if the net is correct (i.e. sound) (cf. Figure 1 and Figure 2). Nevertheless, reduction rules presented in this paper can be used to reduce the size of an RWF-net and to improve the complexity of performing verification. We have applied these reduction rules to workflows modelled in YAWL and found them to be very effective. It is easy to see that all seven rules also apply to arbitrary reset nets while preserving liveness and/or boundedness. Only trivial requirements like pre and postsets not being empty are sufficient.

4.8 Reducing the credit card application process model

We now use the RWF-net in Figure 5 to illustrate the reduction of the credit card application example in a step by step manner. Five reduction rules; the ϕ_{FST}^R rule, the ϕ_{FSP}^R rule, the ϕ_{FPT}^R rule, the ϕ_{FPP}^R rule and the ϕ_{ELT}^R rule, are applied repeated to the model to obtain the reduced RWF-net. The number of elements (transitions and places) in the original reset net is 77 and the number of elements in the final reduced net is 11.

Figure 24 depicts where the ϕ_{FST}^R rule and the ϕ_{FSP}^R rule can be applied to the model. You can see that the ϕ_{FST}^R rule, for instance, cannot be applied to any of the transitions within the reset region of transition cne as one the requirements of this rule is that the intermediate place between two transitions cannot be a reset place. The ϕ_{FST}^R

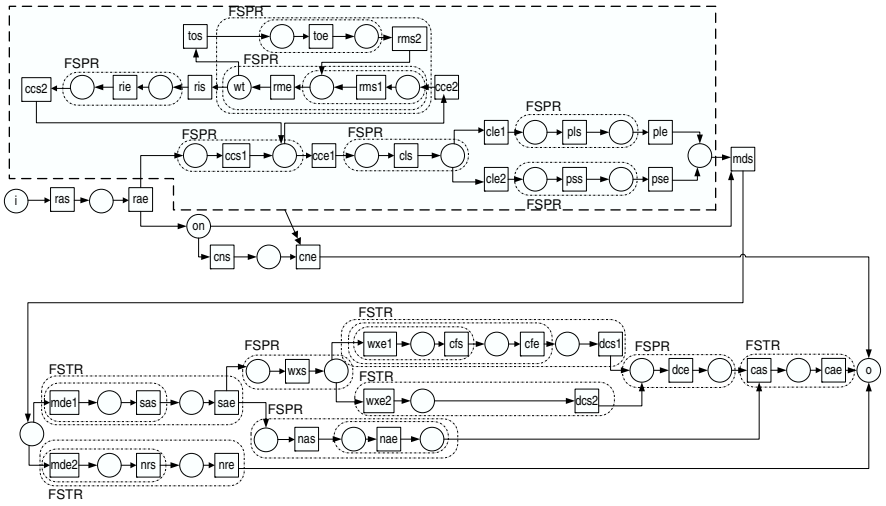

Fig. 24. Applying the ϕ^R_{FST} rule and ϕ^R_{FSP} rule repeatedly to Figure 5

rule has been applied successfully to the other half of the net where there are no reset arcs. Similarly, transitions ras and rae cannot be reduced using the ϕ^R_{FST} rule as one of the output places of transition rae is a reset place. On the other hand, we can apply the ϕ^R_{FSP} rule a number of times to reduce the number of transitions and places within the reset region.

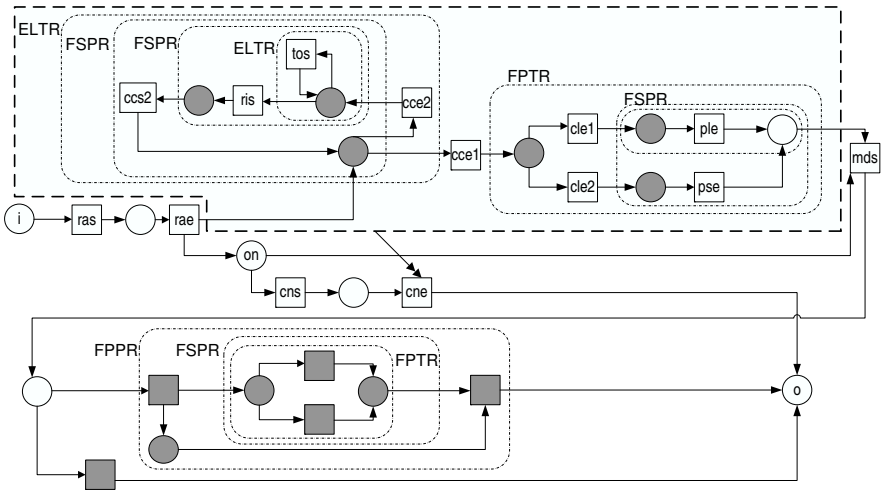

Fig. 25. A reduced RWF-net after applying the reduction rules repeatedly to Figure 24

Figure 25 depicts the reduced net after applying the ϕ^R_{FST} rule and the ϕ^R_{FSP} rule repeatedly. The reduced transitions and places are coloured in grey. It also demonstrates three additional reduction rules that are applicable to the model. One of them is the ϕ^R_{ELT}

rule, which is applied twice to the loop structures involving timeout (tos) and receive more information (ris). This reduction has a significant impact on the calculation of state space during reachability analysis. We can also observe the ϕ^R_{FPT} rule being applied twice, once to the transitions which has reset arcs on their input and output places and the second time to the transitions without any reset arcs on their input and output places. The ϕ^R_{FPP} rule is applied once to reduce the elements in the bottom half of the model.

Figure 26 depicts applicable reduction rules for a final round of reduction. Figure 27 depicts the final reduced net with 11 elements. Please note that according to the ϕ^R_{FSP} rule, it is not possible to reduce the input place i or the output place o. Similarly, the ϕ^R_{FST} rule prevents transitions ras and rae to be reduced due to one of the output places (the grey place) of transition rae having a reset arc. Transitions cns and cne cannot be reduced using the ϕ^R_{FST} rule either, because transition cne has a reset arc. From this, it is clear that no other reduction rules can be applied to this net. Also, note that if an original model has reset arcs, the final reduced net will have one or more reset arcs. It is not possible to entirely abstract from them. In this case, the entire cancellation region is now represented by one place. Next, we discuss how these reduction rules are used in the verification of YAWL process models with cancellation regions.

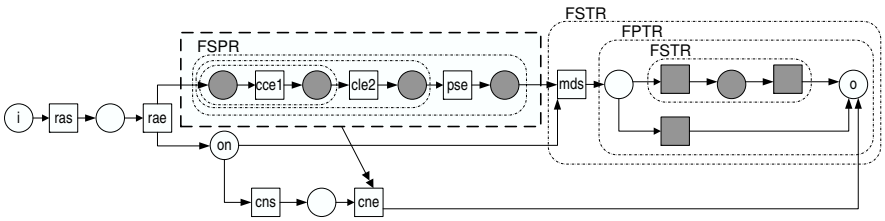

Fig. 26. A reduced RWF-net after applying reduction rules repeatedly to Figure 25

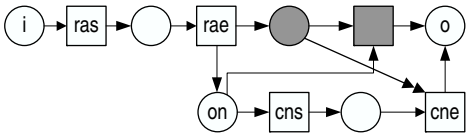

Fig. 27. A reduced RWF-net after applying reduction rules repeatedly to Figure 26

5 Implementation in YAWL

Reduction rules presented in this paper have been implemented in the YAWL Editor (version 1.4.4) to manage the complexity of detecting various desirable properties for YAWL workflows[4]. Using the verification feature, it is possible to detect four desirable properties for YAWL workflows: *soundness*, *weak soundness*, *irreducible cancellation*

[4] Downloadable from http://sourceforge.net/projects/yawl/

regions, and *immutable OR-joins*. Coverability and reachability analyses of reset nets have been used to perform verification of YAWL workflows with cancellation.

Figure 28 shows the results from the soundness property check for a YAWL workflow with cancellation, that has similar semantics to the example RWF-net shown in Figure 1(task F cancels $c1$, C, $c2$, D, $c3$) using reduction rules. The YAWL workflow is first transformed into an equivalent RWF-net with 54 elements. The reduced net has 13 elements after applying these rules recursively. The ϕ^R_{FSP} rule has been applied 18 times, then the ϕ^R_{FST} rule 2 times and then the ϕ^R_{FSP} rule 1 time again.

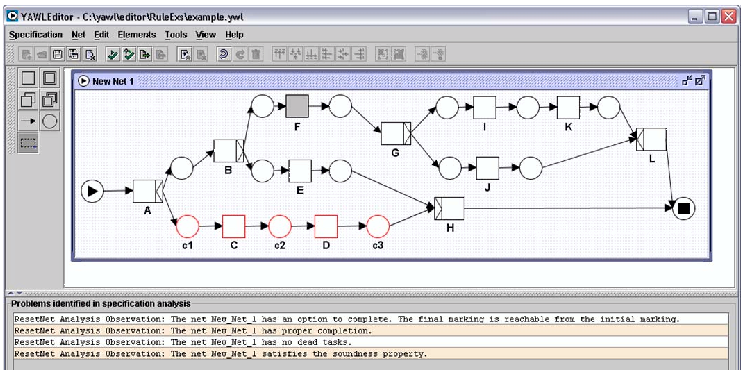

Fig. 28. Soundness property results for a YAWL workflow with cancellation

Figure 29 shows the soundness property check for the credit card application process. The time it takes to verify the soundness property of the credit card application process with 77 elements took approximately 0.35 secs without using reduction rules. When using the reduction rules, the reduced net contains 11 elements and it took only 0.1 secs to perform both the reduction and the soundness check.

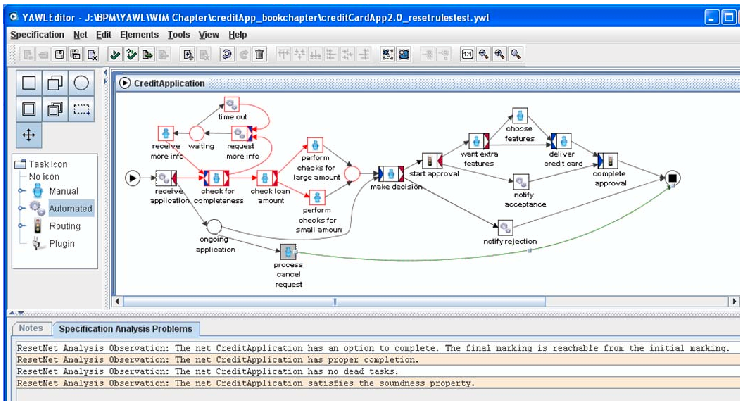

Fig. 29. Soundness property results for credit card application process

We also carried out a case study by modelling the visa application process to Australia as a set of four YAWL workflows with cancellation and OR-joins. The resulting YAWL workflows are then verified for correctness with and without using reduction rules. The analysis of this model became intractable when verifying the soundness property of these four nets without using reduction rules (for details: see the PhD thesis of the first author [16]). The findings support that the reduction rules are necessary for the efficient verification of YAWL workflows as they could speed up the time it takes to generate reachability and coverability sets needed to detect these properties. For instance, the time it takes to verify the soundness property of one of the nets with 89 elements containing many reset arcs took approximately 25 secs without using reduction rules. When using the reduction rules, the reduced net contains 35 elements and took only 4 secs to perform both the reduction and the soundness check.

6 Related work

Reduction rules have been suggested to be used together with Petri nets for the verification of workflows (cf. Chapter 4 of [2]). We follow a similar approach with a set of reduction rules for workflow nets with cancellation regions using reset nets. 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 [8, 9]. A number of authors have investigated reduction rules for Petri nets and for various subclasses of Petri nets. Berthelot presents a set of reduction rules for general Petri nets [4]. They include transformation on places such as structurally redundant places, double places and equivalent places and fusion of transitions such as post-fusion, pre-fusion and lateral fusion and these rules provided inspiration for our work. Six reduction rules are presented for Petri nets In [13] and this set of rules has been used as a starting point for the rules in this paper. In [6], a set of reduction rules are proposed for free-choice Petri nets while preserving well-formedness. Even though reduction rules exist for Petri nets, the nature of reset arcs could invalidate the transformation rules applicable to Petri nets. For example, it is possible that an incorrect net that does not satisfy the proper completion criterion (i.e., tokens can be left in the net when it reaches the end) becomes sound when there is a reset arc to remove the leftover token before completion. Therefore, we proposed extension to the requirements for Petri net reduction rules with additional restrictions with respect to reset arcs. The abstraction rule defined in [6] for free-choice Petri nets has been extended for reset nets. In [14], authors extend the reduction rules given by Berthelot for Time Petri nets. Six reduction rules that preserve correctness for EPCs including reduction rules for trivial constructs, simple splits and joins, similar splits and joins, XOR loop and optional OR-loop are proposed in [7]. However, these reduction rules do not take cancellation into account.

7 Conclusion

In this paper, seven reduction rules for a subclass of reset nets, RWF-nets, were proposed and proven to be correct with respect to the soundness property, a well-established correctness notion in the context of workflow specification. These reduction rules for

RWF-nets were inspired by earlier reduction rules for Petri nets without reset arcs (except for the fusion of equivalent subnets rule). As mentioned in the introduction, a reset arc introduces significant complexities when determining the correct behaviour of a net. This is reflected by the addition of extra context conditions in the reduction rules. It is perhaps not a surprise that reset arcs do not offer new possibilities for reduction rather they limit them. In most cases, the conditions ensure that 1) the place or transition that is removed from the net is not connected to a reset arc or 2) all places and transitions that are merged are connected to the same node via a reset arc. We also demonstrated how a business process model with cancellation feature can be reduced using the proposed reduction rules. The proposed reduction rules have found a practical application in the YAWL open source workflow management system, which provides support for cancellation regions, for the purposes of speeding up verification and reducing the state space. Even though the paper focuses on a subclass of reset nets, RWF-nets, the reduction rules presented for RWF-nets in this paper can be generalised beyond RWF-nets and the rules are *equally applicable to arbitrary reset nets* (cf. Section 4). As the requirements for these rules ensure that occurrence sequences in the original net and the reduced net correspond to each other, these reduction rules are liveness and boundedness preserving.

Acknowledgements. We would like to acknowledge Marc Voorhoeve for providing the construction which proves that the general problem of determining soundness for RWF-nets is not decidable. (See [16] for details.)

References

1. W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azéma and G.Balbo, editors, *Proceedings of Application and Theory of Petri Nets*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426, Toulouse, France, 1997. Springer-Verlag.
2. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods and Systems*. MIT press, Cambridge, MA, 2004.
3. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14:5–51, 2003.
4. G. Berthelot. Transformations and Decompositions of Nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties, Advances in Petri Nets, Proceedings of an Advanced Course, Part 1*, volume 254 of *Lecture Notes in Computer Science*, pages 359–376, Bad Honnef, September 1986. Springer-Verlag.
5. P. Darondeau. Unbounded Petri net Synthesis. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 413–428, Eichstätt, Germany, 2003. Springer-Verlag.
6. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, United Kingdom, 1995.
7. B.F. van Dongen, W.M.P. van der Aalst, and H.M.W. Verbeek. Verification of EPCs: Using Reduction rules and Petri Nets. In O.Pastor and J. Falcão e Cunha, editors, *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE 2005)*, volume 3520 of *Lecture Notes in Computer Science*, pages 372–386, Porto, Portugal, June 2005. Springer-Verlag.
8. 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 Inter-*

national Colloquium on Automata, Languages and Programming, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115, Aalborg, Denmark, July 1998. Springer-Verlag.
9. 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.
10. A. Finkel, J.-F. Raskin, M. Samuelides, and L. van Begin. Monotonic Extensions of Petri Nets: Forward and Backward Search Revisited. *Electronic Notes in Theoretical Computer Science*, 68(6):1–22, 2002.
11. A. Finkel and Ph. Schnoebelen. Well-structured Transition Systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, April 2001.
12. K. van Hee, N. Sidorova, and M. Voorhoeve. Generalised Soundness of Workflow Nets Is Decidable. In J. Cortadella and W. Reisig, editors, *Proceedings of Application and Theory of Petri Nets 2004*, volume 3099 of *Lecture Notes in Computer Science*, pages 197–215. Springer-Verlag, 2004.
13. T. Murata. Petri nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
14. R.H. Sloan and U.A. Buy. Reduction Rules for Time Petri Nets. *Acta Informatica*, 33(7):687–706, 1996.
15. H.M.W. Verbeek. *Verification of WF-nets*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, June 2004.
16. M.T. Wynn. *Semantics, Verification, and Implementation of Workflows with Cancellation Regions and OR-joins*. PhD Thesis, Faculty of Information Technology, Queensland University of Technology, Nov 2006.