# Verifying Workflows with Cancellation Regions and OR-joins: An Approach Based on Reset Nets and Reachability Analysis

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

1.School of Information Systems, Queensland University of Technology
GPO Box 2434, Brisbane Qld 4001, Australia.
{*m.wynn,d.edmond,a.terhofstede*}*@qut.edu.au*
2.Department of Technology Management, Eindhoven University of Technology
PO Box 513, NIL-5600 MB Eindhoven, The Netherlands.
{*w.m.p.v.d.aalst*}*@tm.tue.nl*

**Abstract.** When dealing with complex business processes (e.g., in the context of a workflow implementation or the configuration of some process-aware information system), it is important but sometimes difficult to determine whether a process contains any errors. Cancellation and OR-joins are important features that are common in many business processes. The presence of cancellation and OR-joins makes it difficult to perform verification. Therefore, existing approaches and tools are typically restricted to process models without such features. In this paper, we explore verification techniques for processes with cancellation and OR-joins. We present these techniques in the context of workflow language YAWL that provides direct support for these features. We have extended the graphical editor of YAWL with diagnostic features based in the results presented in this paper. The approach relies on reset nets and can easily be adapted to support other languages allowing for cancellations and OR-joins.

**Keywords**: Workflow management, Verification, Cancellation, OR-joins, Reset nets, YAWL.

## 1 Introduction

Verification of workflows is an important and necessary aspect of process modelling. Verification is concerned with determining, *in advance*, whether a workflow exhibits certain desirable behaviours. Significant organisational resources are needed when introducing new workflow processes and it is important that proper consideration is given to the model at the design stage. By performing this analysis at design time, it is possible to identify potential problems, and if we can identify such problems, the model can be modified before the workflow is executed. This will greatly improve the reliability of a workflow specification.

There are certain desirable characteristics that we expect every business process to exhibit. Firstly, it is important to know that a process, when started, can complete. Secondly, it should not have any other tasks still running for that process when the process ends. Thirdly, the process should not contain tasks that will never be executed. These characteristics closely relate to the soundness property [6]. In this paper, we explore how the introduction of cancellation and OR-joins can affect these properties. *Cancellation* is used to capture the interference of one task in the execution of others. If a task is within the cancellation region of another task, it may be prevented from being started or its execution may be terminated. This is quite common behaviour that needs to be modelled in workflows. For example, you might want to simply cancel other order processing tasks if a customer's credit card payment did not go though. Cancellation is useful but it makes it difficult to verify workflows that use this feature. An *OR-join* is used in situations when we need to model "wait and see" behaviour for synchronisation. For example, a purchase process could involve the separate purchase of two different items and the customer can decide whether he/she wants to purchase one or the other or both. The subsequent payment task is to be performed only once and this requires synchronisation if the customer has selected both products. If the customer selects only one product, no synchronisation is required before payment. Many commercial workflow systems and business process modelling tools support OR-join-like constructs. However, they struggle with the semantics and implementation of the OR-join because synchronisation may depend on the analysis of future execution paths. Like cancellation, OR-joins are useful but they make the verification process quite challenging. For detailed discussion on OR-join semantics, we refer to [4, 7, 14, 15].

The OR-join and cancellation are two of the workflow patterns described in [7]. An in-depth analysis and a comparison of a number of commercially available workflow management systems had been performed [7] and the findings highlight a need for an expressive workflow language that can support all of these workflow patterns including cancellation and OR-joins. Twenty workflow patterns were proposed to address control flow requirements in a language independent style [7]. The workflow language YAWL provides direct support for all but one of these patterns [6] and verification will be performed in the context of this language.

There are established results in the verification of workflow specifications using Petri nets [1, 17]. We explore how these results can be used for workflows with cancellation and OR-joins. We propose to use reset nets which are Petri nets with reset arcs [10, 11]. For verification purposes, YAWL specifications are divided into those with OR-joins and those without OR-joins. This distinction is necessary as a different verification technique is needed in each case. A YAWL net without OR-joins can be mapped to a reset net and it is possible to perform verification on the resulting reset net. However, due to the non-local semantics of OR-joins, it is not possible to map a YAWL net with OR-joins to a reset net (without some approximation) and it is not possible to detect the soundness property for a YAWL net with OR-joins using verification techniques available for reset nets. We therefore propose an alternative verification technique using YAWL formal semantics as defined in [6, 19]. The verification techniques presented here are transferable to any other workflow langauge that is expressive enough to support cancellation regions and OR-joins.

The remainder of this paper is organised as follows. Section 2 discusses correctness notions in the context of YAWL. Section 3 provides the formal foundation for our approach, by introducing reset nets and RWF-nets. Section 4 and 5 present the core results of the paper. First, we focus on YAWL nets without OR-joins. Then, we provide results for YAWL nets with OR-joins. Section 6 describes the implementation of our approach in the YAWL editor. Section 7 discusses related work and concludes the paper.

## 2 Correctness in YAWL

### 2.1 Yet Another Workflow Language (YAWL)

A YAWL specification is made up of tasks, conditions and a flow relation between tasks and conditions. YAWL uses the terms tasks and conditions to avoid confusion with Petri net terminology (transitions and places). The overview of YAWL can be found in [6]. Figure 1 shows some of the YAWL constructs used in this paper and we will explain these YAWL concepts using the example process shown in Figure 2. This process model describes the "lifecycle" of a student who is required

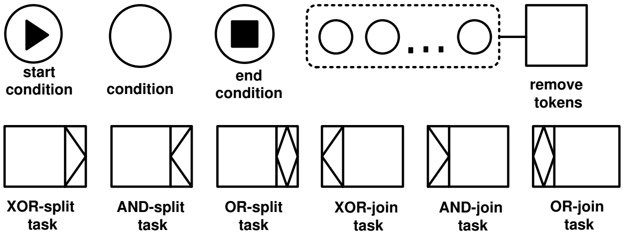


**Fig. 1.** Symbols in YAWL

to take an exam and in parallel may already book a flight to go on holidays after passing the exam. In this "holiday scenario", a student decides to reward himself/herself by going on holidays if he/she passes the exam and cancel the plans if he/she fails the exam. The first task of the process is *Initiate plans* which is directly connected to the start (input) condition. The AND-split behaviour of the *Initiate plans* task indicates that the two tasks *Take exam* and *Book flight* can be done concurrently after *Initiate plans* task is completed. When a token is present in condition $c2$, the *Book flight* task is enabled. Similarly, task *Take exam* is enabled when there is a token in $c1$. After taking the exam, the student waits for the exam results (pass or fail). This is modelled as an XOR-split. If the student passes the exam (a token in $c4$) and the fights have been booked (a token in c3), the student will go on holidays (*Take holiday*). If the student fails the exam (a token in $c5$), he/she resits the exam and also needs to stop holiday planning. This is modelled as a cancellation region linked to the *Resit exam* task and includes the conditions $c2, c3$ and the task *Book flight*. If the holiday plans have been made, the student might also need to contact the travel agent and cancel the flights (*Cancel flight*). This extra task *Cancel flight* is modelled as an alternative route after the *Resit exam* task. Regardless of whether *Take holiday*, *Resit exam* or *Cancel flight* completes, the *Finalise plans* task will be enabled afterwards (XOR-join behaviour). The process will end when *Finalise plans* is completed and a token is placed in the output (end) condition.

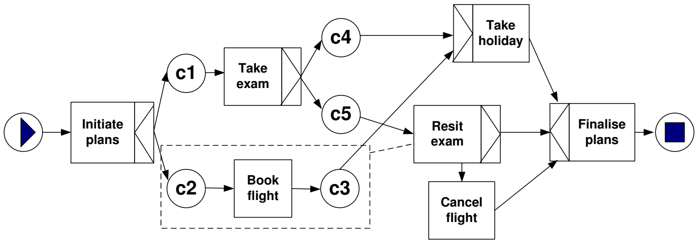


**Fig. 2.** Holiday scenario

It is possible to create a reachability graph of this model and we can observe that the holiday scenario as modelled in Figure 2 is sound. Figure 3 describes a slightly modified version that has neither the weak soundness nor the soundness property. There are two differences: $c3$ is not in the cancellation region of *Resit exam*, and *Cancel flight* is now an AND-join task. Consider the case where the student has failed the exam and has to resit the exam, after booking the flights. The way this process is now modelled, it is possible for task *Finalise Plans* to be executed, without performing

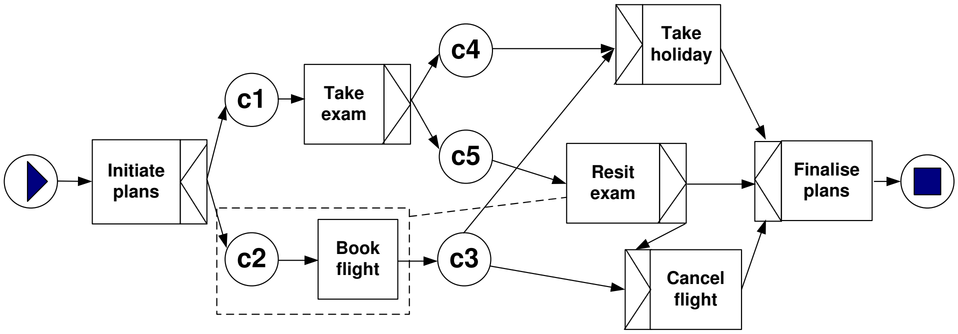


**Fig. 3.** Holiday scenario - with error

task *Cancel Flight* first. Hence, the following occurrence sequence is possible:[1] $i \xrightarrow{I} c1 + c2 \xrightarrow{B} c1 + c3 \xrightarrow{E} c3 + c5 \xrightarrow{R} c3 + c_{RF} \xrightarrow{F} c3 + o$. A token is left in condition $c3$ when a token is put into the output condition $o$ which signals the end of the process. Therefore, the model does not satisfy the proper completion criterion. This example highlights how subtle differences in modelling business processes can adversely affect the correctness of a YAWL specification.

A YAWL specification is formally defined as a nested collection of Extended Workflow Nets (EWF-nets) [6]. A YAWL specification supports hierarchy and a composite task is unfolded into another EWF-net. We refer the reader to [6] for formal definitions. In an EWF-net, it is possible for two tasks to have a direct connection (cf. see tasks *Resit Exam* and *Finalise Plans* in Figure 2). We define the corresponding explicit EWF-net (E2WF-net) for an EWF-net by adding conditions between tasks with direct connections [19]. An E2WF-net can be represented by the tuple $(C, \mathbf{i}, \mathbf{o}, T, F, \mathit{split}, \mathit{join}, \mathit{rem}, \mathit{nofi})$ where $C$ is a set of conditions, $T$ is a set of tasks, $i, o$ are unique input and output conditions, $F$ is the flow relation, $\mathit{split}$ and $\mathit{join}$ specifies the split and join behaviours of each task, $\mathit{rem}$ specifies the cancellation region for a task and $\mathit{nofi}$ specifies the multiplicity of each task. For simplicity, we propose synonyms a YAWL net and an eYAWL-net (explicit YAWL net) for an EWF-net and an E2WF-net respectively. We assume here that all YAWL nets considered in this paper are first transformed into eYAWL-nets.

Let N be an eYAWL-net and $x$ an element of $N$, we use $\bullet x$ and $x \bullet$ to denote the set of inputs and outputs of a node. If the net involved cannot be understood from the context, we explicitly include it in the notation and we write $\overset{N}{\bullet} x$ and $x \overset{N}{\bullet}$. A marking is denoted by $M$ and, just as with ordinary Petri nets, it can be interpreted as a vector, function, and multiset. $M$ is an $m$-vector, where $m$ is the total number of conditions. Let $\mathcal{C}$ be all possible conditions and $M : C \to \mathbb{N}$, where $C \subseteq \mathcal{C}$. $M(c)$ returns the number of tokens in a condition $c$ if $c \in dom(M)$ and zero otherwise. We use notations such as $M \leq M'$, $M + M'$, and $M \dot{-} M'$. $M \leq M'$ iff $\forall_{c \in \mathcal{C}} M(c) \leq M'(c)$. $M + M'$ and $M \dot{-} M'$ are multisets such that $\forall_{c \in \mathcal{C}}$: $(M + M')(c) = M(c) + M'(c)$ and $(M \dot{-} M')(c) = M(c) \dot{-} M'(c)^2$. We represent a multiset by simple enumerating the elements, e.g., 2a+3b+c is the multiset containing two a's, three b's and one c. If X is a set over Y, it could also be interpreted as a bag which assigns to each element a weight of 1.

[1] $I$ stands for *Initiate plans*, $B$ for *Book flight*, $E$ for *Take exam*, $R$ for *Resit exam*, $F$ for *Finalise plans* and $c_{RF}$ for the implicit condition between the two tasks, *Resit exam* and *Finalise plans*.

[2] For any natural numbers $a, b$: $a \dot{-} b$ is defined as $\max(a - b, 0)$.

### 2.2 Structural properties

We now present four structural properties of an eYAWL-net: *soundness*, *weak soundness*, *irreducible cancellation regions*, and *immutable OR-joins.*

An eYAWL-net is sound if and only if it satisfies the following three criteria: option to complete, proper completion and no dead tasks. The soundness property of an eYAWL-net (Definition 1) is closely related to the soundness property of an RWF-net (Definition 18). The only difference is the type of model being considered (RWF-net vs eYAWL-net).

**Definition 1 (Soundness).** *Let $N$ be an eYAWL-net and $M_i, M_o$ be the initial and end markings. $N$ is sound iff:*

1. *option to complete: for every marking $M$ reachable from $M_i$, there exists an occurrence sequence leading from $M$ to $M_o$, and*
2. *proper completion: the marking $M_o$ is the only marking reachable from $M_i$ with at least one token in condition $o$, and*
3. *no dead transitions: for every task $t \in T$, there is a marking $M$ reachable from $M_i$ such that $M[t\rangle$.*

To detect the soundness property, all reachable markings need to be generated and it is not possible to generate reachable markings for a YAWL specification with infinite state space. Therefore, we propose a weaker property called *weak soundness* that describes minimal requirements for the soundness property and that can be used for a YAWL specification with an infinite state space. Definition 19 is closely related to Definition 2. The only difference is the type of model considered (RWF-net vs eYAWL-net). The concept of reachability is defined using YAWL semantics as in [6, 19].

**Definition 2 (Weak soundness).** *Let $N$ be an eYAWL-net and $M_i, M_o$ be the initial and end markings. $N$ satisfies the weak soundness property iff:*

1. *weak option to complete: $M_o$ is coverable from $M_i$, and*
2. *proper completion: there is no marking $M$ coverable from $M_i$ such that $M > M_o$, and*
3. *for every task $t \in T$, no dead transitions: there is a marking $M$ coverable from $M_i$ such that $M[t\rangle$.*

The concept of weak soundness and soundness are discussed in detail using a number of examples. In these examples, we identify a task by it name when no confusion can occur. Also from this point onwards, whenever the term net is left unqualified it refers to a eYAWL-net.

Figure 4 describes a net with an OR-split task A and an AND-join task D. Let $M_i$ and $M_o$ the initial and end markings. First, let us see whether this net satisfies the *weak option to complete* criterion for the weak soundness property. As A is an OR-split task, it is possible to enable B or C or both after firing A. The following occurrence sequence is possible: $i \xrightarrow{A} c1 + c2 \xrightarrow{B} c1 + c3 \xrightarrow{C} c3 + c4 \xrightarrow{D} o$. Therefore, $M_o$ is reachable from $M_i$ and hence, $M_o$ is also coverable from $M_i$. Thus, this net satisfies the *weak option to complete* criterion. The option to complete criterion for the soundness property states that from all reachable markings from $M_i$, $M_o$ should be reachable. Due to the OR-split behaviour of A, there are three possible reachable markings after firing A, $c1$, $c2$ and $c1 + c2$. It has been noted that $c1 + c2$ can reach $M_o$. As for the two reachable markings the following occurrence sequences are possible: $i \xrightarrow{A} c1 \xrightarrow{C} c4$ and $i \xrightarrow{A} c2 \xrightarrow{B} c3$. It can be seen

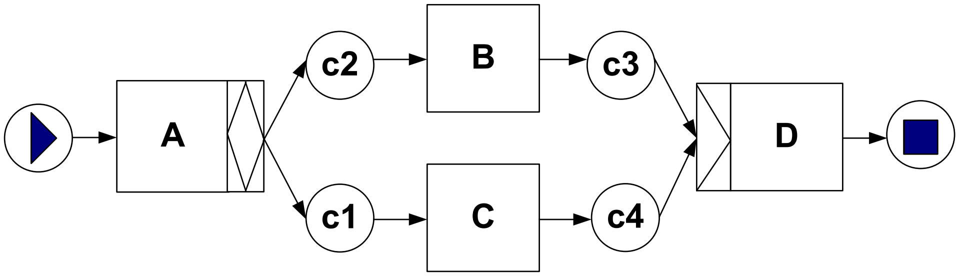


**Fig. 4.** A YAWL net with an OR-split task A and an AND-join task D

that these two sequences cannot reach $M_o$ due to the AND-join behaviour of D and the model will deadlock. As these are reachable markings from $M_i$ that cannot reach $M_o$, it does not satisfy the option to complete criterion for the soundness property.

By following the same principle, we can test whether it satisfies the proper completion and the dead transitions criteria. It is not possible to reach a marking larger than $o$ and therefore, the net satisfies the proper completion criterion. Also, all tasks $A$, $B$, $C$ and $D$ can be enabled at some reachable markings and therefore, no dead transitions criterion is also satisfied. Finally, we can conclude that the net is *weak sound* but not *sound* as it does not satisfy option to complete criterion for the *soundness* property.

Next, we present a net with cancellation in Figure 5 that does not satisfy neither the *weak soundness property* nor the *soundness property*. In this net, A is an AND-split task, D is an AND-join task and the cancellation region for task C includes $c2, c3$ and B. Now, consider a possible occurrence sequence: $i \overset{A}{\rightarrow} c1 + c2 \overset{B}{\rightarrow} c1 + c3 \overset{C}{\rightarrow} c4$. Note that when C fires at the marking $c1 + c3$, a token has been removed from $c3$ and the marking $c4$ is reached. From the marking $c4$, it is not possible to enable D (AND-join) and the net is in deadlock. As $M_o$ is not coverable from $M_i$, the net does not satisfy the weak option to complete criterion and hence, it is not *weak sound*. If the net is not weak sound, then it is also not *sound*.

Note that with a slight modification to the cancellation region in Figure 5, it is possible to create a net that satisfies the weak soundness property. Let us assume now that the cancellation region only contains $c2$ and B and not $c3$. Using the same occurrence sequence as before, it is now possible to have $i \overset{A}{\rightarrow} c1 + c2 \overset{B}{\rightarrow} c1 + c3 \overset{C}{\rightarrow} c3 + c4 \overset{D}{\rightarrow} o$. In this case, the net satisfies the *weak option to complete* criterion as $M_o$ is coverable from $M_i$ and is therefore *weak sound*. However, it is still not *sound* as the following occurrence sequence is possible: $i \overset{A}{\rightarrow} c1 + c2 \overset{C}{\rightarrow} c4$ where a token is removed from $c2$ after firing C at $c1 + c2$. Marking $c4$ is a reachable marking from $M_i$ and from $c4$, $M_o$ cannot be reached. Therefore, the *option to complete* criterion for the *soundness property* cannot be fulfilled.

These examples illustrate that it is not easy to detect potential problems without performing the full state space analysis and they motivate us to develop an analysis technique for detecting the correctness of YAWL specifications. In addition to the weak soundness property and the soundness property for YAWL nets, two additional properties for nets with cancellation regions and OR-joins are proposed: *irreducible cancellation regions* and *immutable OR-joins*. These properties are propose to decide whether a net contains any unnecessary OR-joins and cancellation regions.

Reducible elements in the cancellation region of a task represent elements that can never be active and therefore, can never be cancelled by the task. For instance, in Figure 6, condition $c3$ is modelled

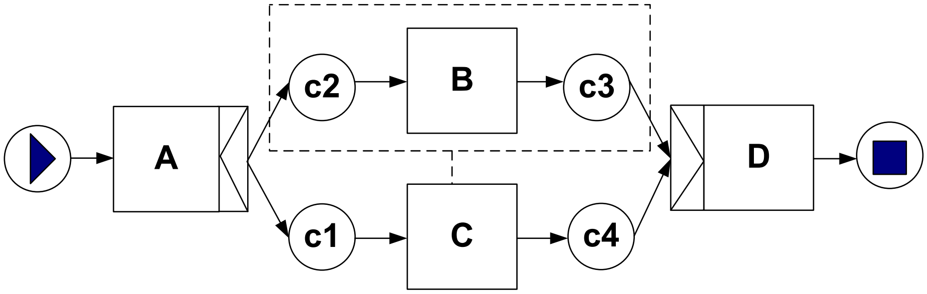


**Fig. 5.** A YAWL net with cancellation

to be in the cancellation region of task $CT$. However, after executing task $A$, a decision is made to either execute task $B$ or $CT$ but not both as A is an XOR-split. Therefore, it is never possible to mark condition $c3$ while task $CT$ is executing. In the above example, the term "executing" has been used loosely. According to the YAWL semantics in [6], a task can be in one of the three task states mi_$e_t$, exec$_t$, and mi_$c_t$[3]. These states represent the intermediate states for enabling, executing and completing a YAWL task. For our purpose, we consider a task to be executing if there is a token in any one of these three states. We denote this set of places for a task $t$ as $Q_t^E = \{\text{mi\_}e_t, \text{exec}_t, \text{mi\_}c_t\}$.

**Definition 3 (Reducible cancellation element).** *Let $N$ be an eYAWL-net. $N$ has a reducible cancellation element $x$, iff there is a task $t \in T$ such that $x \in rem(t)$ and*

- *if $x \in C$, a marking where $t$ is executing and $x$ is marked is not coverable from $M_i$, i.e., $\forall_{p \in Q_t^E} \neg \exists_{M \in N[M_i\rangle} (M \geq p + x)$,*
- *if $x \in T$, a marking where both $t$ and $x$ are executing is not coverable from $M_i$, i.e., $\forall_{p \in Q_t^E} \forall_{q \in Q_x^E} \neg \exists_{M \in N[M_i\rangle} (M \geq p + q)$.*

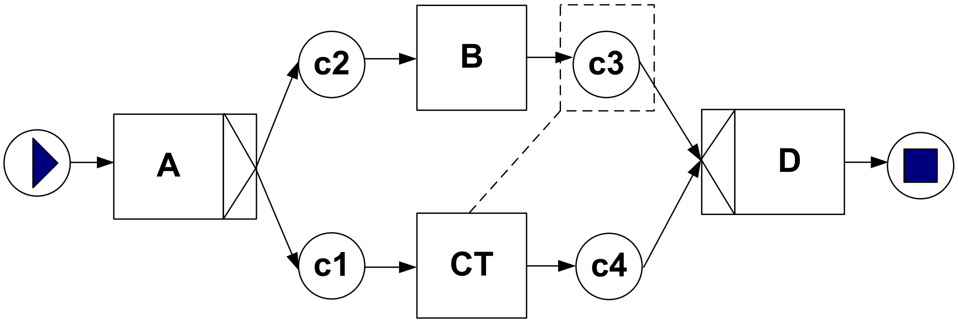


**Fig. 6.** A YAWL net with a (reducible) condition $c3$ in the cancellation region of $CT$

[3] There is one other state for a YAWL task, mi_$a_t$, which is used for multiple instances.

**Definition 4 (Irreducible cancellation regions).** *Let $N$ be an eYAWL-net. $N$ satisfies the irreducible cancellation regions property iff for all $x \in ran\ rem$, $x$ is not a reducible cancellation element.*

An OR-join task is said to be convertible, when it could be better represented as either an XOR-join or an AND-join task. Such tasks arise in two circumstances: (i) when it is never possible to reach a marking which marks more than one input conditions of the task and (ii) when all input conditions of the task are marked in all markings that enables the OR-join task. The objective is to detect unnecessary OR-join tasks at design time as the non-local semantics of OR-join requires expensive runtime analysis. This can be detected by looking at markings in the reachability set that enable an OR-join task. In Figure 7, OR-join task $D$ is only enabled when all input conditions are marked (due to an AND-split task $A$) and therefore, $D$ should be modelled as an AND-join instead of an OR-join.

**Definition 5 (Convertible OR-join).** *Let $N$ be an eYAWL-net and $t$ be an OR-join task in $N$. OR-join task $t$ can be modelled as*

- *an XOR-join if only one condition in $\bullet t$ is ever marked in the enabling markings of $t$, i.e., $\forall_{M \in N[M_i\rangle}(M[t\rangle \implies \exists!_{p \in \bullet t}(M(p) > 0))$,*
- *an AND-join if for all conditions in $\bullet t$ are always marked in the enabling markings of $t$, i.e., $\forall_{M \in N[M_i\rangle}(M[t\rangle \implies \forall_{p \in \bullet t}(M(p) > 0))$.*

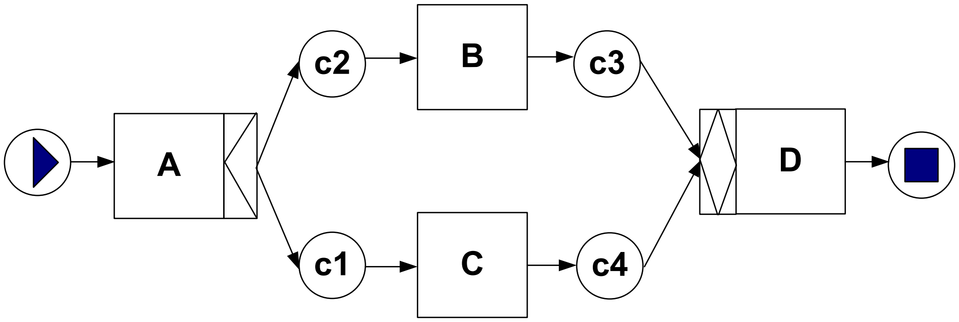


**Fig. 7.** A YAWL net with a (convertible) OR-join task $D$

**Definition 6 (Immutable OR-joins).** *Let $N$ be an eYAWL-net. $N$ satisfies the immutable OR-joins property iff for all $t \in T$, $join(t) = OR$ implies that $t$ is not a convertible OR-join.*

## 3 Formal Foundation

While inspired by Petri nets [16], YAWL should not be seen as an extension of these. YAWL constructs such as the OR-join, cancellation and multiple instances are not directly supported by Petri nets. The cancellation feature of YAWL is theoretically closely related to reset nets. The reset arcs are used to underpin the *rem* function that models the cancellation feature of YAWL. Our verification approach involves translation of YAWL specifications in terms of reset nets. This translation is made possible by abstracting from multiple instances and hierarchy in YAWL [19]. This approach allows us to leverage existing results and techniques in the area of Petri nets and reset nets in particular [8–13]. In this section, we present the definitions associated with reset nets.

### 3.1 Reset nets

A reset net is a Petri net with special reset arcs, that can clear the tokens in selected places [10, 11]. Reset arcs do not change the requirements of enabling a transition but when a transition fires, they will remove *all* tokens from the specified places.

**Definition 7 (Reset net).** *A Petri net is a tuple* $(P, T, F)$ *where* $P$ *is a set of places,* $T$ *is a set of transitions,* $P \cap T = \varnothing$ *and* $F \subseteq (P \times T) \cup (T \times P)$*. A reset net is a tuple* $(P, T, F, R)$ *where* $(P, T, F)$ *is a Petri net and* $R \in T \nrightarrow \mathbb{P}(P)$ *provides the reset places for a subset of transitions.*

In the remainder of the paper, when we use the expression $F(x, y)$, it denotes 1 if $(x, y) \in F$ and 0 if $(x, y) \notin F$. We write $F^+$ for the transitive closure of the flow relation $F$ and $F^*$ for the reflexive transitive closure of $F$. The notation $\mathbb{M}(N)$ is used to represent possible markings of a reset net $N$. Let $N = (P, T, F, R)$ be a reset net, then $\mathbb{M}(N) = P \rightarrow \mathbb{N}$. A transition is *enabled* when there are enough tokens in its input places. Note that reset arcs do not change the requirements of enabling a transition.

**Definition 8 (Enabling rule).** *Let* $N$ *be a reset net,* $t \in T$*, and* $M \in \mathbb{M}(N)$*. Transition* $t$ *is enabled at* $M$*, denoted as* $M[t\rangle$*, if and only if* $\forall p \in \bullet t : M(p) \geq 1$.

The concept of firing a transition $t$ in a net $N$ is formally defined in Definition 9 and denoted as $M \overset{N,t}{\rightarrow} M'$. If there can be no confusion regarding the net, we will abbreviate the expression to $M \overset{t}{\rightarrow} M'$ and if the transition is not relevant we write $M \rightarrow M'$.

**Definition 9 (Forward firing).** *Let* $N = (P, T, F, R)$ *be a reset net,* $t \in T$ *and* $M, M' \in \mathbb{M}(N)$.

$$M \overset{N,t}{\rightarrow} M' \Leftrightarrow M[t\rangle \wedge$$
$$M'(p) = \begin{cases} M(p) - F(p, t) + F(t, p) & \textit{if} \quad p \in P \setminus R(t) \\ F(t, p) & \textit{if} \quad p \in R(t). \end{cases}$$

It is possible to fire a sequence of transitions from a given marking in a reset net resulting in a new marking using the forward firing rule defined above. This sequence of transitions is represented as an occurrence sequence.

**Definition 10 (Occurrence sequence).** *Let* $N = (P, T, F, R)$ *be a reset net and* $M, M_1, ..., M_n \in \mathbb{M}(N)$*. If* $M \overset{t_1}{\rightarrow} M_1 \overset{t_2}{\rightarrow} ... \overset{t_n}{\rightarrow} M_n$ *are firing occurrences then* $\sigma = t_1 t_2 ... t_n$ *is an occurrence sequence leading from* $M$ *to* $M_n$ *and we write* $M \overset{\sigma}{\rightarrow} M_n$.

We now define the concepts of reachability and coverability of markings from a given marking in a reset net. A marking $M'$ is reachable from another marking $M$ in a reset net, if there is an occurrence sequence leading from $M$ to $M'$.

**Definition 11 (Reachability).** *Let* $N = (P, T, F, R)$ *be a reset net and* $M, M' \in \mathbb{M}(N)$*.* $M'$ *is reachable in N from* $M$*, denoted* $M \overset{N}{\rightarrow} M'$*, if there exists an occurrence sequence* $\sigma$ *such that* $M \overset{\sigma}{\rightarrow} M'$.

**Definition 12 (Coverability).** *Let $N = (P, T, F, R)$ be a reset net and $M_1, M_2 \in \mathbb{M}(N)$. $M_2$ is coverable from $M_1$ in N, if there exists a reachable marking $M'$ from $M_1$ such that $M' \geq M_2$.*

Next, we present two notations: *projection* and *filtering* to allow operations on selected places of a marking in a reset net. The notation $M[P']$ restricts $M$ to a set of places $P'$, i.e., a projection. For places not in $P'$, the number of tokens is zero. Let $M_1 = p1 + p2 + p3$ and $P' = \{p1, p2\}$. $M_1[P'] = p1 + p2 + 0p3$ and $\text{dom}(M_1[P']) = \{p1, p2, p3\}$. Let $M_2 = p1 + 2p2$, $M_2[P'] > M_1[P']$ is true as the comparison between $M$ and $M'$ is restricted to the set of places in $P'$ and $M_2$ has more tokens in $p2$.

**Definition 13 (Projection).** *Let $N = (P, T, F, R)$ be a reset net, $M \in \mathbb{M}(N)$ and $P' \subseteq P$. $M[P']$ returns a projection such that dom$(M[P'])$ = dom$(M)$ and*

$$M[P'](p) = \begin{cases} M(p) & \textit{if} \quad p \in P' \\ 0 & \textit{if} \quad p \notin P'. \end{cases}$$

The notation $M \restriction P'$ is used to alter a marking based on a set of places $P'$. Let $M = p1 + p2 + p3$ and $P' = \{p1, p2\}$. $M \restriction P' = p1 + p2$ and $\text{dom}(M \restriction P') = \{p1, p2\}$. If $P' = \{p1, p2, p3, p4\}$, $M \restriction P' = p1 + p2 + p3 + 0p4$ and $\text{dom}(M \restriction P') = \{p1, p2, p3, p4\}$.

**Definition 14 (Filtering $\restriction$).** *Let $N = (P, T, F, R)$ be a reset net, $M \in \mathbb{M}(N)$ and $P \subseteq P'$. $M \restriction P'$ returns a function such that dom$(M \restriction P') = P'$ and*

$$M \restriction P'(p) = \begin{cases} M(p) & \textit{if} \quad p \in P' \cap \textit{dom}(M) \\ 0 & \textit{if} \quad p \in P' \setminus \textit{dom}(M). \end{cases}$$

We conclude this section with the notion of *Backward firing* that is used to generate coverable markings for a reset net by firing transitions backwards.

**Definition 15 (Backward firing).** *Let $(P, T, F, R)$ be a reset net and $M, M' \in \mathbb{M}(N)$. $M' \dashrightarrow^t M$ iff it is possible to fire a transition $t$ backwards starting from $M$ and resulting in $M'$.*

$$M' \dashrightarrow^t M \Leftrightarrow M[R(t)] \leq t \bullet [R(t)] \wedge$$
$$M'(p) = \begin{cases} (M(p) \dot{-} F(t, p)) + F(p, t) & \textit{if} \quad p \in P \setminus R(t) \\ F(p, t) & \textit{if} \quad p \in R(t). \end{cases}$$

For places that are not reset places, the number of tokens in $M'$ is determined by the number of tokens in $M$ for $p$. If a place is an output place of $t$ and not a reset place, one token is removed from $M(p)$ if $M(p) > 0$. If a place is an input place of $t$ and not a reset place, one token is added to $M(p)$. For any reset place $p$, $M(p) \leq F(t, p)$ because it is emptied when firing and then $F(t, p)$ tokens are added. We do not require $M(p) = F(t, p)$ because the aim is coverability and not reachability. $M'$, i.e., the marking before (forward) firing $t$, should *at least* contain the *minimal* number of tokens required for enabling $t$ and resulting in a marking of at least $M$. Therefore, only $F(p, t)$ tokens are assumed to be present in a reset place $p$.

### 3.2 Reset WorkFlow Nets (RWF-nets)

In this subsection, we propose a subclass of reset nets called RWF-nets and define soundness and weak soundness properties for these nets. An RWF-net satisfies the following restrictions. There is a unique begin place and a unique end place and also every node in the graph is on a directed path from the begin place to the end place.

**Definition 16 (RWF-net).** *Let $N = (P, T, F, R)$ be a reset net. The net $N$ is an RWF-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 $(n, o) \in F^*$.*

In an RWF-net, there is an input place $i$ and an output place $o$ and we now define an initial marking $M_i$ and an end marking $M_o$ as follows:

**Definition 17 (Initial marking and End marking).** *Let $N = (P, T, F, R)$ be an RWF-net and $i, o$ be the input and output places of the net. The initial marking of $N$ is denoted as $M_i$ and it represents a marking where there is a token in the input place $i$ (i.e., $M_i = i$). Similarly, the end marking of $N$ is denoted as $M_o$ and it represents a marking where that is a toke in the output place $o$ (i.e., $M_o = o$).*

We now define two structural properties for an RWF-net: *soundness* and *weak soundness*. The *soundness* definition for an RWF-net is based on the soundness definition from [6] for WF-nets. An RWF-net is sound if and only if it satisfies three criteria: *option to complete*, *proper completion* and *no dead transitions*.

**Definition 18 (Soundness).** *Let $N = (P, T, F, R)$ be an RWF-net and $M_i, M_o$ be the initial and end markings. $N$ is sound iff:*

1. *option to complete: for every marking $M$ reachable from $M_i$, there exists an occurrence sequence leading from $M$ to $M_o$, i.e., for all $M \in N[M_i\rangle : M_o \in N[M\rangle$, and*
2. *proper completion: the marking $M_o$ is the only marking reachable from $M_i$ with at least one token in place $o$, i.e, for all $M \in N[M_i\rangle : M \geq M_o \Rightarrow M = M_o$, and*
3. *no dead transitions: for every transition $t \in T$, there is a marking $M$ reachable from $M_i$ such that $M[t\rangle$, i.e, for all $t \in T$ there exists an $M \in N[M_i\rangle$ such that $M[t\rangle$.*

Note that reachability is not decidable for arbitrary reset nets [11] and hence, its applicability is limited to reset nets with finite state space. As the soundness property definition relies on reachability results, the soundness property is only decidable for an RWF-net with a finite state space. Fortunately, coverability is decidable for a reset net using the backward firing rule of Definition 15 [10–13]. We thus propose a weaker property called *weak soundness* which can be decided using coverability results. The weak soundness definition for an RWF-net relaxes the first criterion and reformulates the second and third criteria using coverability results. For the weak option to complete criterion, it only checks whether it is possible to cover the final marking $M_o$ from $M_i$ (i.e. is there at least a path that leads from $M_i$ to $M_o$). It does not check whether all paths lead to the final marking and hence, it will not detect partial deadlocks. Therefore, if an RWF-net satisfies the soundness property, it also satisfies the weak soundness property but not vice versa.

**Definition 19 (Weak soundness).** *Let $N = (P, T, F, R)$ be an RWF-net and $M_i, M_o$ be the initial and end markings. $N$ satisfies the weak soundness property iff:*

1. *weak option to complete: $M_o$ is coverable from $M_i$, and*
2. *proper completion: there is no marking $M$ coverable from $M_i$ such that $M > M_o$, and*
3. *no dead transitions: for every transition $t \in T$, there is a marking $M$ coverable from $M_i$ such that $M[t\rangle$.*

## 4 Verifying YAWL nets without OR-joins

In this section, we focus our attention on verification techniques for YAWL nets without OR-joins. We propose to transform an eYAWL-net (without OR-joins) into an RWF-net to exploit the analysis techniques available for reset nets [19]. This is achieved by first abstracting from multiple instances and hierarchy in YAWL and then applying function *transE2WF* to transform an eYAWL-net into an RWF-net. Formal definition of *transE2WF* is given in [19]. The transformation returns an RWF-net where input and output conditions $\mathbf{i}, \mathbf{o} \in C$ map to unique begin and end places $\mathbf{i}, \mathbf{o} \in P$ in the corresponding RWF-net and where every node in the graph $(P \cup T', F')$ is on a directed path from $\mathbf{i}$ to $\mathbf{o}$.

**Lemma 1.** *Let $N = (C, \mathbf{i}, \mathbf{o}, T, F, \mathit{split}, \mathit{join}, \mathit{rem}, \mathit{nofi})$ be an eYAWL-net without OR-joins. $N' =$ transE2WF$(N) = (P, T', F', R)$ is an RWF-net.*

**Definition 20 (transE2WF).** *Let $N = (C, \mathbf{i}, \mathbf{o}, T, F, \mathit{split}, \mathit{join}, \mathit{rem}, \mathit{nofi})$ be an eYAWL-net without OR-joins. The function transE2WF$(N)$ returns $N' = (P, T', F', R)$ such that*

$P = C \cup \{p_t | t \in T\}$ *is a set of places,*
$T' = T_{start} \cup T_{end}$ *such that*
$T_{start} = \{t_S | t \in T \ \wedge \ join(t) = AND\}$
$\qquad \cup \{t_S^p | t \in T \ \wedge \ join(t) = XOR \ \wedge \ p \in \bullet t\}$,
$T_{end} = \{t_E | t \in T \ \wedge \ split(t) = AND\}$
$\qquad \cup \{t_E^p | t \in T \ \wedge \ split(t) = XOR \ \wedge \ p \in t \bullet\}$
$\qquad \cup \{t_E^x | t \in T \ \wedge \ split(t) = OR \ \wedge \ x \subseteq t \bullet \ \wedge x \neq \varnothing\}$,
$F' = \{(p, t_S) | t \in T \ \wedge \ join(t) = AND \ \wedge \ p \in \bullet t\}$
$\quad \cup \{(t_S, p_t) | t \in T \ \wedge \ join(t) = AND\}$
$\quad \cup \{(p_t, t_E) | t \in T \ \wedge \ split(t) = AND\}$
$\quad \cup \{(t_E, p) | t \in T \ \wedge \ split(t) = AND \ \wedge \ p \in t \bullet\}$
$\quad \cup \{(p, t_S^p) | t \in T \ \wedge \ join(t) = XOR \ \wedge \ p \in \bullet t\}$
$\quad \cup \{(t_S^p, p_t) | t \in T \ \wedge \ join(t) = XOR \ \wedge \ p \in \bullet t\}$
$\quad \cup \{(p_t, t_E^p) | t \in T \ \wedge \ split(t) = XOR \ \wedge \ p \in t \bullet\}$
$\quad \cup \{(t_E^p, p) | t \in T \ \wedge \ split(t) = XOR \ \wedge \ p \in t \bullet\}$
$\quad \cup \{(p_t, t_E^x) | t \in T \ \wedge \ split(t) = OR \ \wedge \ x \subseteq t \bullet \ \wedge x \neq \varnothing\}$
$\quad \cup \{(t_E^x, p) | t \in T \ \wedge \ split(t) = OR \ \wedge \ p \in x \ \wedge \ x \subseteq t \bullet \ \wedge x \neq \varnothing\}$,
$R = \{(t_E, \{p_{t'} | t' \in rem(t) \cap T\} \cup (rem(t) \cap C)) | t \in T \ \wedge \ split(t) = AND\}$
$\quad \cup \{(t_E^p, \{p_{t'} | t' \in rem(t) \cap T\} \cup (rem(t) \cap C)) | t \in T \ \wedge split(t) = XOR$
$\quad \wedge \, p \in t \bullet\}$
$\quad \cup \{(t_E^x, \{p_{t'} | t' \in rem(t) \cap T\} \cup (rem(t) \cap C)) | t \in T \ \wedge split(t) = OR$
$\quad \wedge \, x \subseteq t \bullet \ \wedge x \neq \varnothing\}$
$\quad \cup \{(t_E, \varnothing) | t \in T \setminus dom\, rem \ \wedge \ split(t) = AND\}$
$\quad \cup \{(t_E^p, \varnothing) | t \in T \setminus dom\, rem \ \wedge \ split(t) = XOR \wedge p \in t \bullet\}$
$\quad \cup \{(t_E^x, \varnothing) | t \in T \setminus dom\, rem \ \wedge \ split(t) = OR \wedge \ x \subseteq t \bullet \ \wedge x \neq \varnothing\}$
$\quad \cup \{(t, \varnothing) | t \in T_{start}\}$.

A transformation function transE2WF converts a net without OR-joins into the corresponding RWF-net. Function $R$ stores all transitions and its associated reset places. As a task in an eYAWL-net is now split into a number of $t_S$ and $t_E$ transitions depending on the split and join behaviour, a place $p_t$

is introduced for each task $t$ to represent an internal place between $t_S$ and $t_E$. The flow relation $F'$ is also modified so that the newly introduced places in $P'$ and transitions $T'$ are properly connected. Figure 9 shows the RWF-net corresponding to the YAWL net in Figure 2. Places $i$ and $o$ represent

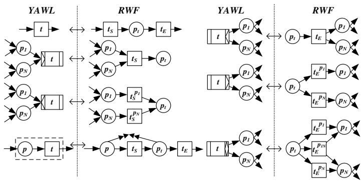


**Fig. 8.** Reset net transformations for YAWL split and join behaviours

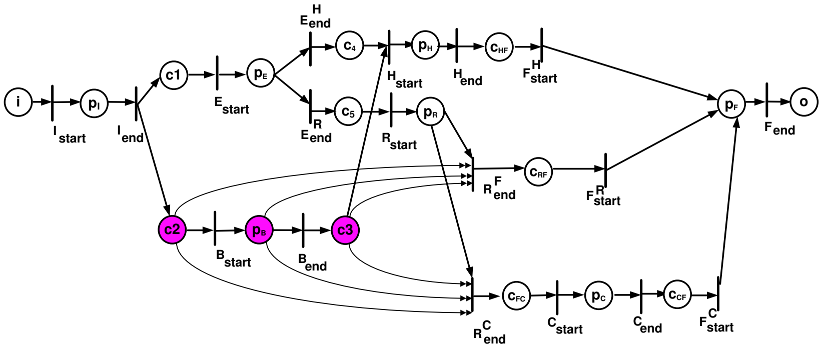


**Fig. 9.** Holiday scenario - RWF-net (note: double-headed reset arcs from $c2$, $c3$ and $p_B$ to transitions $R^F_{end}$ and $R^C_{end}$)

unique input and output places. We use the following abbreviations for the tasks: *Initiate plans* - I, *Take exam* - E, *Book flight* - B, *Resit exam* - R, *Cancel flight* - C, *Finalise plans* - F. Each task in the eYAWL-net has been transformed into the corresponding start and end transitions. For instance, task *Initiate plans* is now represented as $I_{start}$ and $I_{end}$ with the internal place $p_I$. The cancellation region associated with the *Resit exam* task is represented by double-headed reset arcs from the places $c2$, $c3$ and $p_B$ (the internal place for *Book flight*) to the end transitions of *Resit exam* task, $R^F_{end}$ and $R^C_{end}$.

Coverability results from reset nets are used to decide three desirable properties for a YAWL specification: *weak soundness*, *soundness*, and *irreducible cancellation regions*. In this section, as we are considering only nets without OR-joins, the fourth property, *immutable OR-joins* has been omitted from discussion.

We have formulated the three criteria of the weak soundness property for an RWF-net using the notion of coverability. As coverability is decidable for a reset net using backwards firing rule as discussed in [19], the three criteria of the weak soundness property are decidable. The **Coverable** procedure described in [19] is used to determine whether a marking is coverable from the initial marking in a reset net. We exploit these results to propose an algorithmic approach for deciding the weak soundness property of an eYAWL-net without OR-joins.

The weak option to complete criterion is concerned with whether an eYAWL-net can complete (i.e., reach the end condition from the initial marking). Therefore, we need to detect whether a marking that marks the end place (or bigger) can be reached in the corresponding RWF-net.

**Observation 1** *Given an eYAWL-net without OR-joins, the weak option to complete can be decided by testing whether $M_o$ is coverable from $M_i$ in the corresponding RWF-net.*

An eYAWL-net is considered to have finished its processing when the unique end condition ($o$) of the net is marked. Hence, if it is possible to mark $o$ together with any other condition in the net, the net does not satisfy the proper completion criterion. Using the coverable procedure, we test whether it is possible to mark $o$ with any another place in the corresponding RWF-net.

**Observation 2** *Given an eYAWL-net without OR-joins, proper completion can be decided by testing whether $o + p$ is not coverable from $M_i$ in the corresponding RWF-net for all $p \in P$.*

A task in an eYAWL-net is a dead task, if it is not fireable at any reachable markings from the initial marking. We can test the presence of a dead task $t$ in an eYAWL-net by determining whether its internal place $p_t$ is coverable from the initial marking in the corresponding RWF-net.

**Observation 3** *Given an eYAWL-net without OR-joins, no dead tasks criterion can be decided by testing whether $p_t$ is coverable from $M_i$ in the corresponding RWF-net for all $t \in T$.*

As it is possible to decide these three criteria for weak soundness using coverability results for an eYAWL-net without OR-joins, the weak soundness property is decidable.

**Theorem 1.** *Given an eYAWL-net without OR-joins, weak soundness is decidable.*
***Proof:*** *Follows from observations 1, 2 and 3.*

Similarly, we use the notion of coverability as defined on reset nets to determine whether an eYAWL-net without OR-joins satisfies the irreducible cancellation regions property. A condition should not be in a cancellation region of a task, if that condition never contains tokens when the task attempts to cancel it. To determine this, we test whether a marking that marks the internal place of the task as well as the condition is coverable from the initial marking in the corresponding RWF-net.

**Observation 4** *Given an eYAWL-net without OR-joins, whether a condition $c$ is reducible in a cancellation region of $t$ can be decided by testing whether $c + p_t$ is coverable from $M_i$ in the corresponding RWF-net.*

Similarly, a task $tx$ should not be in a cancellation region of another task $t$, if task $tx$ is never active when task $t$ attempts to cancel it. To determine this, we test whether a marking that marks the internal places of both tasks is coverable from the initial marking in the corresponding RWF-net.

**Observation 5** *Given an eYAWL-net without OR-joins, whether a task $tx$ is reducible in a cancellation region of $t$ can be decided by testing whether $p_{tx} + p_t$ is coverable from $M_i$ in the corresponding RWF-net.*

**Theorem 2.** *Given an eYAWL-net without OR-joins, whether there is a reducible element (a condition or a task) in a cancellation region of a task is decidable.*
***Proof:*** *Follows from observations 4 and 5.*

For an eYAWL-net without OR-joins and a finite state space, it is possible to decide the soundness property by generating a reachability graph for the corresponding RWF-net. Figure 10 shows the reachability graph for the reset net in Figure 9. It is easy to see whether an RWF-net satisfies the soundness property by testing the three criteria. If the corresponding RWF-net is sound, then the net without OR-joins is also sound.

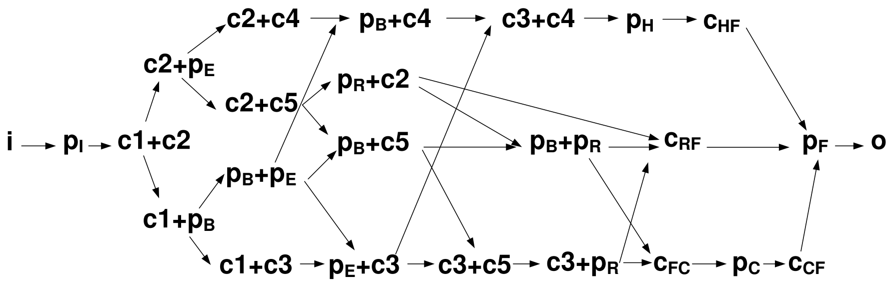


**Fig. 10.** Reachability graph for the RWF-net in Figure 9

**Observation 6** *Given an eYAWL-net without OR-joins and a finite reachability graph, soundness is decidable.*

## 5 YAWL nets with OR-joins

Section 4 shows how a YAWL net without OR-joins can be transformed and analysed using the corresponding RWF-net. In this section, we focus our attention on YAWL nets with OR-joins. Due to the non-local semantics of an OR-join, a net with OR-joins cannot be mapped directly into a reset net. Even though nets with OR-joins are difficult to define formally, it is important to be able to decide the correctness of those models. Therefore, it is important to explore which properties can be detected for YAWL nets with OR-joins. To do this, all OR-joins in a net are first translated into XOR-joins. This idea is based on the optimistic approach for treatment of OR-joins as defined in [19]. After replacing all OR-joins with XOR-joins, it is now possible to transform the net into an RWF-net using the transformation rule.

Even though an XOR-join translation cannot capture the exact semantics of an OR-join, it provides some insights into the three criteria for the weak soundness property. We show here that there is a direct relationship between the sets of reachable markings generated by an eYAWL-net with OR-joins and a corresponding eYAWL-net with XOR-joins and that a reachable marking after firing an XOR-join task is larger than or equal to the reachable marking after firing the corresponding OR-join task.

Figure 11 illustrates the comparative markings for these two nets. Let $N^{OR}$ be an eYAWL-net with an OR-join task, $t^{OR}$, and $N^{XOR}$ the corresponding eYAWL-net with the XOR-join task, $t^{XOR}$. Let $M_j^{OR} = M_j^{XOR}$ be the reachable markings where $t^{OR}$ and $t^{XOR}$ are enabled. When $t^{OR}$ fires at $M_j^{OR}$, a token is removed from all marked conditions in its preset and a marking $M_{j+1}^{OR}$ is reached. When $t^{XOR}$ fires at $M_j^{XOR}$, a token is removed from one of the marked conditions in its preset and a marking $M_{j+1}^{XOR}$ is reached. Hence, the marking reached after firing the XOR-join task $t^{XOR}$ is larger than or equal to the marking reached after firing the OR-join task $t^{OR}$ (i.e., $M_{j+1}^{XOR} \geq M_{j+1}^{OR}$). As a marking with more tokens can enable at least the same transitions as a marking with less tokens, if $M_o^{OR}$ is a reachable marking in $N^{OR}$, $M_o^{XOR}$ is also reachable in $N^{XOR}$ where $M_o^{XOR} \geq M_o^{OR}$. Therefore, we conclude that if $N^{OR}$ has the weak option to complete then $N^{XOR}$ will have the weak option to complete. Similarly, If $N^{OR}$ has no dead transitions then $N^{XOR}$ will have no dead transitions. If $N^{OR}$ does not have proper completion then $N^{XOR}$ will not have proper completion.

$$\begin{tikzcd}
M_i^{OR} \arrow[r, dash, dashed] \arrow[d, <->, dashed] & M_j^{OR} \arrow[r, "{N^{OR}, t^{OR}}"] \arrow[d, <->, dashed, "M_j^{OR} = M_j^{XOR}"'] & M_{j+1}^{OR} \arrow[r, dash, dashed] \arrow[d, <->, dashed, "M_{j+1}^{XOR} \geqslant M_{j+1}^{OR}"'] & M_o^{OR} \arrow[d, <->, dashed, "M_o^{XOR} \geqslant M_o^{OR}"'] \\
M_i^{XOR} \arrow[r, dash, dashed] & M_j^{XOR} \arrow[r, "{N^{XOR}, t^{XOR}}"] & M_{j+1}^{XOR} \arrow[r, dash, dashed] & M_o^{XOR}
\end{tikzcd}$$

**Fig. 11.** Comparative markings between an eYAWL-net with an OR-join and the corresponding eYAWL-net with an XOR-join

**Observation 7** *Given an eYAWL-net with OR-joins, if it satisfies weak option to complete then the corresponding eYAWL-net without OR-joins (where OR-joins are transformed into XOR-joins) satisfies the weak option to complete.*

Note that this observation does not hold in the opposite direction. A counter example is given in Figure 12. The model has an AND-split task A, an OR-join task D and an AND-join task E. The following occurrence sequence is possible: $i \overset{A}{\to} c1 + c2 \overset{B}{\to} c2 + c3 \overset{C}{\to} c3 + c4 \overset{D}{\to} c5$. OR-join task D will wait for both tasks B and C to complete, before firing and the only reachable marking from D is $c5$. To enable E, there should be a reachable marking $c4 + c5$. As it is not possible to reach $c4 + c5$, the model always deadlocks at E. Hence, the model does not satisfy the weak option to complete criterion. Now, consider the translated version where task D is treated as an XOR-join task instead. In this case, the following occurrence sequence is possible: $i \overset{A}{\to} c1 + c2 \overset{B}{\to} c2 + c3 \overset{C}{\to} c3 + c4 \overset{D}{\to} c4 + c5 \overset{E}{\to} o$. Therefore, the translated net has the weak option to complete property when the original net with OR-joins does not.

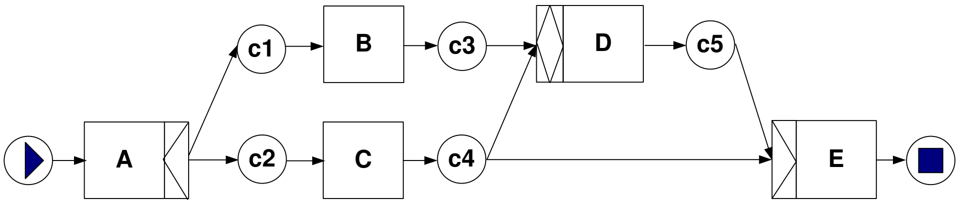


**Fig. 12.** This YAWL net with an OR-join task $D$ always deadlocks at $E$.

**Observation 8** *Given an eYAWL-net with OR-joins, if it has no dead transitions then the corresponding eYAWL-net without OR-joins (where OR-joins are transformed into XOR-joins) has no dead transitions.*

**Observation 9** *Given an eYAWL-net with OR-joins, if it does not have proper completion then the corresponding eYAWL-net without OR-joins (where OR-joins are transformed into XOR-joins) does not have proper completion.*

We now summarise the decidable criteria for a net with OR-joins using XOR-join translations.

**Observation 10** *Given an eYAWL-net $N$ with OR-joins, let $N'$ be the corresponding eYAWL-net where all OR-joins of $N$ have been replaced with XOR-joins and $RN$ be the equivalent RWF-net for $N'$. The following holds:*

1. *if $RN$ does not satisfy the weak option to complete criterion then $N$ does not satisfy the weak option to complete criterion,*
2. *if $RN$ has dead transitions then $N$ has dead transitions, and*
3. *if $RN$ satisfies the proper completion criterion, then $N$ satisfies the proper completion criterion.*

Note that only limited results are available and soundness property is not decidable using this approach. Therefore, we propose an alternative approach for a net with OR-joins that has a finite state space. We propose to create a reachability graph by taking into account OR-join semantics. Such a reachability graph can be constructed using enabling and firing rules as defined in [6] together with the OR-join semantics defined in this thesis. First, we need a way to represent a marking where a task is in the executing state. By following a similar concept used in the function for the reset net mapping, we propose to transform an eYAWL-net before generating the reachability graph. The function **unfoldYNet** introduces an internal condition to represent the executing state of a task. All tasks in a YAWL net are split into two tasks (e.g., a task $t$ is split into a start task, $t_S$, and an end task, $t_E$, with an internal condition $p_t$). If a task is within the cancellation region of another task, its internal condition is put into the cancellation set instead. The cancellation region of a task is associated with the end task as the cancellation is carried out just before completing the task. As multiple instance tasks do not affect the analysis, and hence we abstract from them.

**Definition 21 (unfoldYNet).** *Let $N_1 = (C_1, \mathbf{i}, \mathbf{o}, T_1, F_1, split_1, join_1, rem_1, nofi_1)$ be an eYAWL-net, the function unfoldYNet$(N_1)$ returns*
*$N_2 = (C_2, \mathbf{i}, \mathbf{o}, T_2, F_2, split_2, join_2, rem_2, nofi_2)$ such that*

$C_2 = C_1 \cup \{p_t \mid t \in T_1\}$,
$T_2 = T_{start} \cup T_{end}$ *such that* $T_{start} = \{t_S \mid t \in T_1\}$ *and* $T_{end} = \{t_E \mid t \in T_1\}$,
$F_2 = \{(c, t_S) \mid (c, t) \in F_1\} \cup \{(t_S, p_t) \mid t \in T_1\}$
$\quad \cup \{(p_t, t_E) \mid t \in T_1\} \cup \{(t_E, c) \mid (t, c) \in F_1\}$,
$split_2 = \{(t_S, AND) \mid t \in T_1\} \cup \{(t_E, split_1(t)) \mid t \in T_1\}$,
$join_2 = \{(t_S, join_1(t)) \mid t \in T_1\} \cup \{(t_E, XOR) \mid t \in T_1\}$,
$rem_2 = \{(t_E, (rem_1(t) \cap C_1) \cup \{p_{ct} \mid ct \in (rem_1(t) \cap T_1)\}) \mid t \in T_1\}$,
$nofi_2 = \varnothing$.

For nets with a finite state space, we can generate a reachability graph. From such a graph, it is possible to decide whether a net with OR-joins has the soundness property or not.

**Observation 11 (Soundness)** *Given an eYAWL-net $N$ with OR-joins and a finite reachability graph, the soundness property of $N$ is decidable by testing whether the soundness property holds for the corresponding net unfoldYNet$(N)$.*

Using a finite reachability graph, it is also possible to detect the immutable OR-join property. The enabling markings for an OR-join task can be observed from the reachability graph of a YAWL-net.

**Observation 12 (Immutable OR-joins)** *Given an eYAWL-net $N$ with OR-joins and a finite reachability graph, the immutable OR-joins property of $N$ is decidable by testing whether the immutable OR-joins property holds for the corresponding net unfoldYNet$(N)$.*

Similarly, the following observations can be made regarding the existence of reducible cancellation regions in a net with OR-joins and a finite reachability graph. To determine coverability, we use the reachability set that has been generated using YAWL semantics.

**Observation 13** *Given an eYAWL-net $N$ with OR-joins and a finite reachability graph, whether a condition $c$ is reducible in a cancellation region of $t$ in the net $N$ can be decided by testing whether $c + p_t$ is coverable from $M_i$ in the corresponding net unfoldYNet$(N)$.*

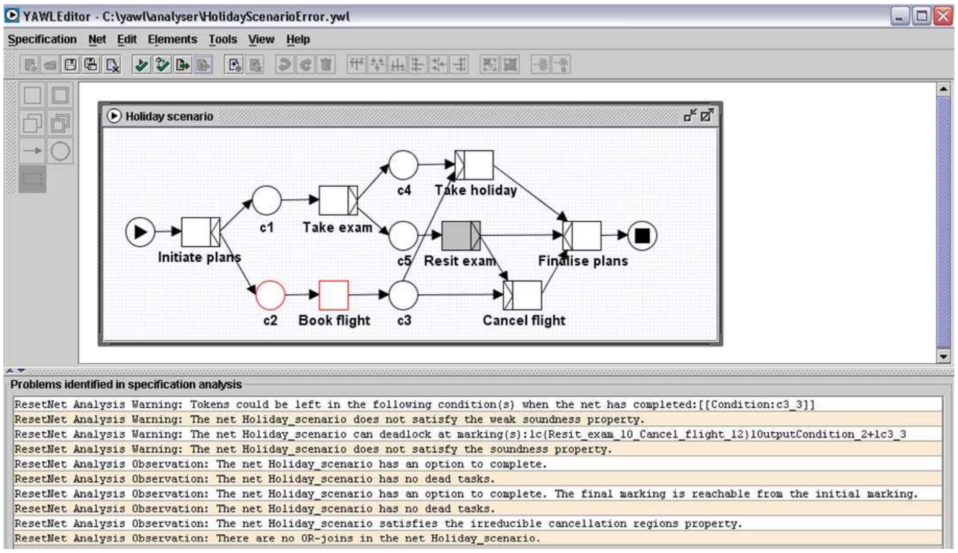
**Observation 14** *Given an eYAWL-net $N$ with OR-joins and a finite reachability graph, whether a task $tx$ is reducible in a cancellation region of $t$ in the net $N$ can be decided by testing whether $p_{tx} + p_t$ is coverable from $M_i$ in the corresponding net unfoldYNet$(N)$.*

**Observation 15 (Irreducible cancellation regions)** *Given an eYAWL-net with OR-joins and a finite reachability graph, whether there is a reducible element (a condition or a task) in a cancellation region of a task is decidable.*

## 6 Verification in YAWL

Although the results of this paper have been presented in the context of YAWL, it is important to realise that the basic ideas can also be applied to other languages supporting cancellation and OR-joins. The reason that we selected YAWL is that it is a compact language with formal semantics that is highly expressive. Moreover, the YAWL language is supported by a YAWL editor to create diagrams and the YAWL engine to enact processes. Both the editor and the engine can be obtained via www.yawl-system.com.

We have extended the editor to support the verification approach presented in this chapter. The verification function can perform checks for all four structural properties: *weak soundness*, *soundness*, *irreducible cancellation regions* and *immutable OR-joins*. If a YAWL specification contains multiple nets, diagnosis is given for each net individually. A typical usage scenario is as follows: a process designer uses the editor to describe a YAWL specification, he/she then performs verification of certain properties, he/she observes the verification messages, and then makes appropriate changes. There are two types of verification messages: *warnings* and *observations*. Warnings are given when the net violates a certain structural property (e.g., the net XYZ does not satisfy the weak soundness property). An additional warning message is provided for each criterion that is violated and it can be used to pinpoint the problem areas in the net. Observations are given for each correct criterion. Figure 13 shows verification messages for the net of Figure 3.



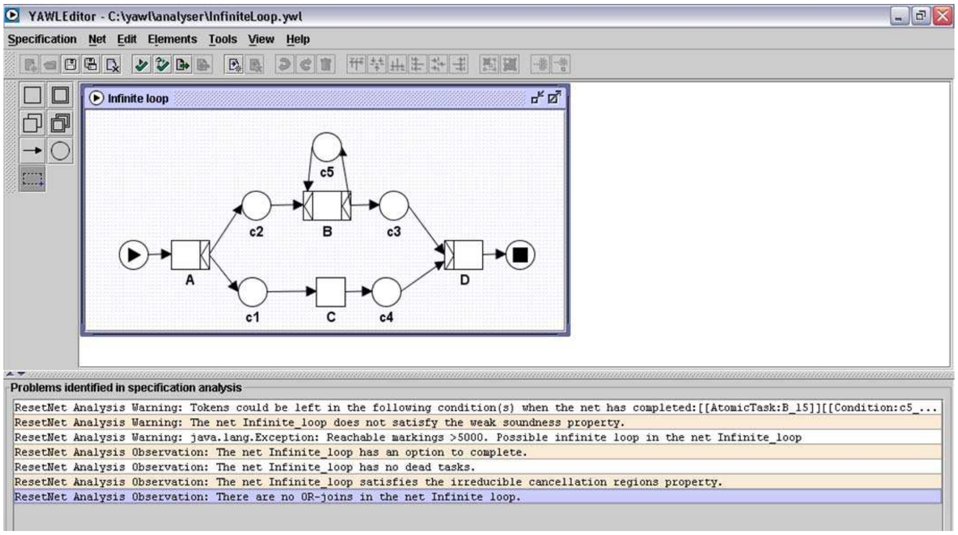
```
ResetNet Analysis Warning: Tokens could be left in the following condition(s) when the net has completed:[[Condition:c3_3]]
ResetNet Analysis Warning: The net Holiday_scenario does not satisfy the weak soundness property.
ResetNet Analysis Warning: The net Holiday_scenario can deadlock at marking(s):1c{Resit_exam_10_Cancel_flight_12}1OutputCondition_2+1c3_3
ResetNet Analysis Warning: The net Holiday_scenario does not satisfy the soundness property.
ResetNet Analysis Observation: The net Holiday_scenario has an option to complete.
ResetNet Analysis Observation: The net Holiday_scenario has no dead tasks.
ResetNet Analysis Observation: The net Holiday_scenario has an option to complete. The final marking is reachable from the initial marking.
ResetNet Analysis Observation: The net Holiday_scenario has no dead tasks.
ResetNet Analysis Observation: The net Holiday_scenario satisfies the irreducible cancellation regions property.
ResetNet Analysis Observation: There are no OR-joins in the net Holiday_scenario.
```

**Fig. 13.** Screenshot of Holiday scenario with errors

Figure 14 displays verification messages for a net with an unbounded place. As stated before, the reachability algorithm can only be used for a specification with a finite state space. We do not have a formal means of detecting whether a net has an infinite state space. A pragmatic approach is taken and the generation of reachable markings is started. The program sets a threshold for a maximum number of markings to consider during the generation of reachable markings. Note that the maximum number of markings is currently set to 5000 for testing purposes. The program also detects whether an *out of memory* error has occurred and this is taken as an indication that the model possibly has infinite state space. Figure 15 displays verification messages from the weak soundness property check for the net with an OR-join task D as shown in Figure 12. The messages state that the weak soundness property cannot be decided in this case using the coverability results. However, it is still possible to decide whether the soundness property holds for the net, as we can construct a reachability graph for the net using the YAWL semantics.

## 7 Related Work and Conclusion

This paper focuses on the verification of YAWL specifications with and without OR-joins. Petri nets based techniques have been applied to workflow verification before [1, 2, 5, 17]. In [3], the author describes how structural properties of a workflow net can be used to determine the soundness property. In [18], the authors present an alternative approach for deciding the relaxed soundness property using invariants. The approach taken results in an approximation of the OR-join semantics and transformation of YAWL nets into Petri nets with inhibitor arcs. However, the use of inhibitor arcs instead of reset arcs means that this approach cannot detect problems in certain YAWL specifications with cancellation features. For example, this approach cannot detect problems in the erroneous holiday scenario described in Figure 3. On the other hand, the approximation of OR-join semantics enables


YAWLEditor - C:\yawl\analyser\InfiniteLoop.ywl
Specification
Net
Edit
Elements
Tools
View
Help
Infinite loop
c5
c2
B
c3
A
D
c1
C
c4
Problems identified in specification analysis

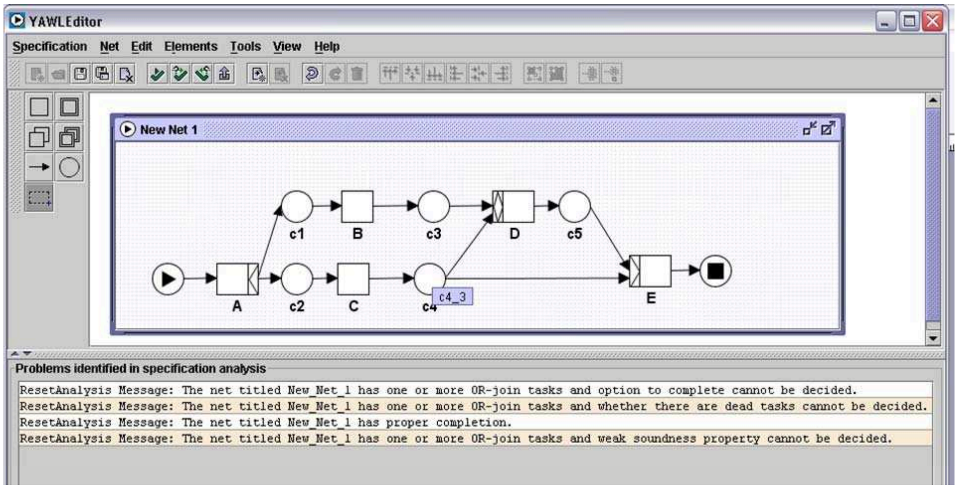
```
ResetNet Analysis Warning: Tokens could be left in the following condition(s) when the net has completed:[[AtomicTask:B_15]][[Condition:c5_...
ResetNet Analysis Warning: The net Infinite_loop does not satisfy the weak soundness property.
ResetNet Analysis Warning: java.lang.Exception: Reachable markings >5000. Possible infinite loop in the net Infinite_loop
ResetNet Analysis Observation: The net Infinite_loop has an option to complete.
ResetNet Analysis Observation: The net Infinite_loop has no dead tasks.
ResetNet Analysis Observation: The net Infinite_loop satisfies the irreducible cancellation regions property.
ResetNet Analysis Observation: There are no OR-joins in the net Infinite loop.
```

**Fig. 14.** Screenshot of a YAWL net with an infinite loop

the verification of YAWL nets with OR-joins using invariants. It overcomes one of the limitations in our approach that reachability analysis has to be performed for a net with OR-joins to decide the soundness property.

The use of reset nets in workflow verification is original and it has been proposed here to deal with cancellation and OR-join features of workflows. Our approach involves translation of YAWL specifications in terms of a subclass of reset nets (RWF-nets) and the use of coverability and reachability results from reset nets for verification [9–13]. We define four desirable properties for YAWL specifications: *weak soundness property*, *soundness property*, *irreducible cancellation regions* and *immutable OR-joins*. We explore how these properties can be detected in YAWL specifications. For YAWL nets without OR-joins, reachability and coverability results on the corresponding RWF-nets are used to detect the soundness property and the weak soundness property. A different approach is needed for YAWL nets with OR-joins. To detect the weak soundness property, the net is first transformed into a corresponding YAWL net with XOR-joins. We then transform the net into an RWF-net to determine the weak option to complete, proper completion and no dead transitions criteria. To detect the soundness property of YAWL nets with OR-joins and finite state space, reachability analysis is performed using YAWL semantics. The main findings in the paper are as follows:

- For YAWL nets without OR-joins, the weak soundness property and the irreducible cancellation regions property are decidable using coverability results for reset nets.
- For YAWL nets without OR-joins and a finite state space, the soundness property is decidable using reachability results for reset nets.
- For YAWL nets with OR-joins, only limited results are available using coverability results for reset nets by first replacing OR-joins with XOR-joins.
- For YAWL nets with OR-joins and a finite state space, the soundness property, the irreducible cancellation regions property and the immutable OR-joins property are decidable using reachability results from the YAWL semantics.



```
ResetAnalysis Message: The net titled New_Net_1 has one or more OR-join tasks and option to complete cannot be decided.
ResetAnalysis Message: The net titled New_Net_1 has one or more OR-join tasks and whether there are dead tasks cannot be decided.
ResetAnalysis Message: The net titled New_Net_1 has proper completion.
ResetAnalysis Message: The net titled New_Net_1 has one or more OR-join tasks and weak soundness property cannot be decided.
```

**Fig. 15.** Screenshot of a YAWL net with an OR-join task D

**Acknowledgements.** We would like to thank Lindsay Bradford and Lachlan Aldred for their assistance during the implementation of verification functionality in the YAWL Editor.

## 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. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
3. W.M.P. van der Aalst. Workflow verification: Finding control-flow errors using peti net-based techniques. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Porcess Management:Models, Techniques and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, page 161. Springer-Verlag, 2000.
4. W.M.P van der Aalst, J. Desel, and E. Kindler. On the Semantics of EPCs: A Vicious Circle. In M. Rump and F.J. Nüttgens, editors, *Proceedings of the EPK 2002: Business Process Management using EPCs*, pages 71–80, Trier, Germany, 2002. Gesellschaft für Informatik, Bonn.
5. W.M.P. van der Aalst, A. Hirnschall, and E. Verbeek. An alternative way to analyze workflow graphs. In C. Woo A. Pidduck, J. Mylopoulos and M. Tamer Özsu, editors, *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (27-31 May)*, volume 2348 of *Lecture Notes in Computer Science*, pages 534–552, Toronto, Canada, 2002. Springer-Verlag.
6. W.M.P. van der Aalst and A.H.M ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, June 2005.
7. 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.
8. P.A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. General decidability theorems for infinite-state systems. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (27 - 30 July)*, pages 313–321, New Brunswick, NJ, July 1996. IEEE Computer Society.

9. 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.
10. C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset Nets Between Decidability and Undecidability. In K. Larsen, S. Skyum, and G. Winskel, editors, *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115, Aalborg, Denmark, July 1998. Springer-Verlag.
11. 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.
12. 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.
13. A. Finkel and Ph. Schnoebelen. Well-structured Transition Systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, April 2001.
14. B. Kiepuszewski. *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows*. PhD thesis, Queensland University of Technology, Brisbane, Australia, 2003.
15. E. Kindler. On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. In J. Desel, B. Pernici, and M. Weske, editors, *Proceedings of 2nd International Conference on Business Process Management*, volume 3080 of *Lecture Notes in Computer Science*, pages 82–97, Potsdam, Germany, 2004. Springer–Verlag.
16. T. Murata. Petri nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, Apr 1989.
17. H.M.W. Verbeek. *Verification of WF-nets*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, June 2004.
18. H.M.W. Verbeek, W.M.P. van der Aalst, and A.H.M ter Hofstede. Verifying Workflows with Cancellation Regions and OR-joins: An Approach Based on Invariants. Technical Report BETA Working Paper Series, WP 156, Eindhoven University of Technology, Eindhoven, The Netherlands, 2006.
19. M.T. Wynn, D. Edmond, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Achieving a General, Formal and Decidable Approach to the OR-join in Workflow using Reset nets. In G. Ciardo and P. Darondeau, editors, *Proceedings of the 26th International conference on Application and Theory of Petri nets and Other Models of Concurrency (20 - 25 June)*, volume 3536 of *Lecture Notes in Computer Science*, pages 423–443, Miami, USA, June 2005. Springer-Verlag.