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]

School of Information Systems, Queensland University of Technology
GPO Box 2434, Brisbane Qld 4001, Australia.
{*m.wynn,d.edmond,a.terhofstede*}*@qut.edu.au*
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, 15, 16].

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, 19]. 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 [11, 12]. 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, 21]. 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 that needs to

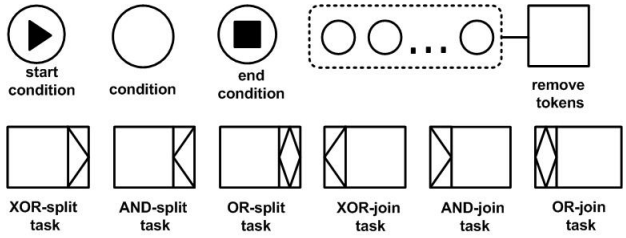

Fig. 1. Symbols in YAWL

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* could be done concurrently after *Initiate plans* task is completed. When a token is present in the 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) and it 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 *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 *Resit exam* task. Regardless of whether *Take holiday*, *Resit exam* or *Cancel flight* completes, *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.

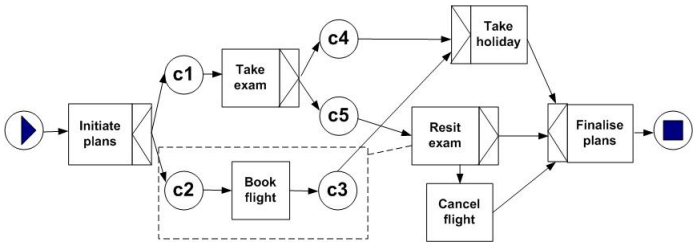

Fig. 2. Holiday scenario

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 [21]. An E2WF-net can be represented by the tuple $(C, \mathbf{i}, \mathbf{o}, T, F, split, join, rem, nofi)$ 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, $split$ and $join$ specifies the split and join

behaviours of each task, rem specifies the cancellation region for a task and $nofi$ 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 $M : C \rightarrow \mathbb{N}$, 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 can 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 $(M \dot{-} M')(c) = M(c) \dot{-} M'(c)$ [1]. 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.

2.2 Properties

We propose to detect the correctness of a YAWL specification by analysing whether it satisfies *the weak soundness property* and *the soundness property*. An eYAWL-net is sound iff it satisfies the following three criteria: option to complete, proper completion and no dead tasks. 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 the weak soundness property that describes the minimal requirements for the soundness property and that can be used for a YAWL specification with infinite state space. We also present two other properties: *reducible cancellation regions* and *convertible OR-joins*. The concepts of reachability and coverability are defined using YAWL semantics as in [6, 21].

Definition 1 (Soundness). *Let N be an eYAWL-net, i, o be the input and output conditions of the net and M_i, M_o be the initial and end markings, i.e., $M_i = i$ is the initial state marking only condition i and $M_o = o$ is the end state marking only condition o. N is sound iff: 1) for every marking M reachable from M_i, there exists a firing sequence leading from M to M_o (Option to complete), 2) the marking M_o is the only marking reachable from M_i with at least one token in place 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 (No dead transitions).*

Definition 2 (Weak soundness). *Let N be an eYAWL-net and i, o be the input and output conditions of the net and $M_i = i$ and $M_o = o$ are the initial and end markings. N has weak soundness property iff: 1) M_o is coverable from M_i (Weak option to complete), 2) there is no marking M coverable from M_i such that $M > M_o$ (Proper completion) and 3) for every transition $t \in T$, there is a marking M coverable from M_i such that t is enabled at M (No dead transitions).*

The holiday scenario as modelled in Figure 2 satisfies both weak soundness and soundness properties. Figure 3 describes a slightly modified version that does not have either the weak soundness or the soundness property. There are two differences: $c3$ is not in the cancellation region of *Resit exam*,

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

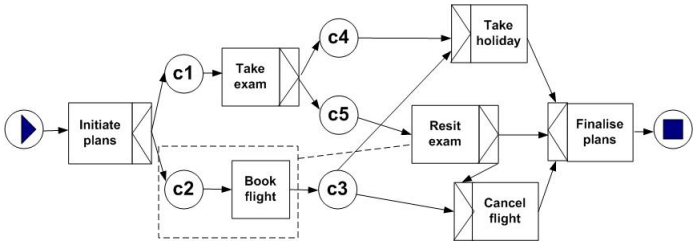

Fig. 3. Holiday scenario - with error

and *Cancel flight* is now an AND-join task. Consider the case where the student has failed the exam and has to resit, after booking the flights. The way this process is now modelled, it is possible for task *Finalise Plans* to be executed, without performing task *Cancel Flight* first. Hence, the following occurrence sequence is possible: [2] $i \xrightarrow{I} c1 + c2 \xrightarrow{B} c1 + c3 \xrightarrow{E} c3 + c5 \xrightarrow{R} c3 + c_{RF} \xrightarrow{F} c3 + o$. 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 satisfies proper completion criterion. This example highlights how subtle differences in modelling business processes can adversely affect the correctness of a YAWL specification.

In addition to the weak soundness property and the soundness property for YAWL nets, we propose two additional properties: *Reducible cancellation regions* and *Convertible OR-joins* for YAWL nets with cancellation regions and OR-joins. Reducible elements in a cancellation region of a task represent elements that can never be cancelled while that task is being executed (e.g. conditions may never contain tokens). We are interested in determining whether cancellation regions modelled in the business processes are really necessary. For instance, in Figure 4, condition $c3$ is modelled 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 (XOR-split). Therefore, it is never possible for condition $c3$ to contain tokens while task CT is executing.

Definition 3 (Reducible cancellation regions). *Let N be an eYAWL-net. N has a reducible element e, if there is a task t such that $e \in rem(t)$ and e can never be cancelled when t is being executed.*

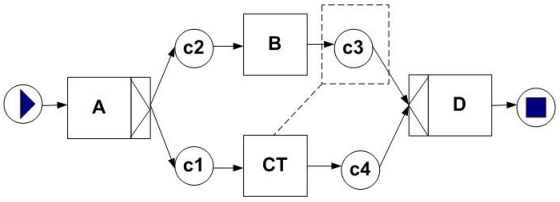

Fig. 4. a YAWL net with a reducible condition $c3$ in cancellation region for CT

Non-local OR-join semantics in YAWL results in expensive runtime analysis [21]. It is therefore desirable to determine in advance whether a more appropriate join structure could be found for a task modelled as an OR-join in a YAWL net. A convertible OR-join task is where it is never possible to mark more than one input condition of an OR-join task (more suitable as an XOR-join) or when all the input conditions can always become marked (more suitable as an AND-join).

[2] 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*.

Definition 4 (A convertible OR-join). *Let N be an eYAWL-net and t be an OR-join task in N. OR-join task t is convertible to an XOR-join if only one condition in $\bullet t$ is always marked in the enabling markings of t or to an AND-join if all conditions in $\bullet t$ are always marked in the enabling markings of t.*

3 Formal Foundation

While inspired by Petri nets [18], 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 [21]. This approach allows us to leverage existing results and techniques in the area of Petri nets and reset nets in particular [8, 10–14]. 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 [11, 12]. 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 5 (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 $\mathbb{M}(N) = P \to \mathbb{N}$.

When a transition t of a reset net N is enabled at a marking M, it can fire and reach another marking M' represented as $M \overset{N,t}{\to} M'$. If there can be no confusion regarding the net, we will abbreviate it as $M \overset{t}{\to} M'$.

Definition 6 (Forward firing). *Let $N = (P, T, F, R)$ be a reset net and $M, M' \in \mathbb{M}(N)$. A transition $t \in T$ is enabled at M, denoted $M[t\rangle$, iff $\bullet t \leq M$.*

$$M \overset{N,t}{\to} M' \Leftrightarrow \bullet t \leq M \wedge$$
$$M'(p) = \begin{cases} M(p) - F(p,t) + F(t,p) & \text{if} \quad p \in P \setminus R(t) \\ F(t,p) & \text{if} \quad p \in R(t). \end{cases}$$

Definition 7 (Occurrence sequence). *Let $N = (P, T, F, R)$ be a reset net and $M \in \mathbb{M}(N)$. If $M \overset{t_1}{\to} M_1 \overset{t_2}{\to} \ldots \overset{t_n}{\to} M_n$ are transition occurrences then $\sigma = t_1 t_2 \ldots t_n$ is an occurrence sequence leading from M to M_n and we write $M \overset{\sigma}{\to} M_n$.*

Definition 8 (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, if there exists an occurrence sequence σ such that $M \overset{\sigma}{\to} M'$.*

Definition 9 (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 $M' \geq M_2$.*

The notation $M[P']$ restricts M to a set of places P', i.e., a projection. Let $M_1 = p1 + p2 + p3$ and $P' = \{p1, p2\}$. $M_1[P'] = p1 + p2$ and dom$(M_1[P']) = \{p1, p2\}$. Let $M_2 = p1 + 2p2$, $M_2[P'] > M_1[P']$ is true as the comparison between M and M' is restricted to a set of places in P' and M_2 has more tokens in $p2$.

Definition 10 (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$(M) \cap P'$ and for all $p \in$ dom$(M[P'])$ $M[P'](p) = M(p)$.*

Here, we define the notion of backward firing that will be used to analyse coverability [11, 13, 17].

Definition 11 (Backward firing [21]). *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'.*

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

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 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 12 (RWF-net). *Let $N = (P, T, F, R)$ be a reset net. The reset net N is an RWF-net iff 1) there exists exactly one $i \in P$ such that $\bullet i = \varnothing$, 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 $(n, o) \in F^*$.*

The soundness definition for an RWF-net is based on the soundness definition from [6] for WF-nets. An RWF-net is sound iff it satisfies the following three criteria: option to complete, proper completion and no dead transitions.

Definition 13 (Soundness). *Let $N = (P, T, F, R)$ be an RWF-net, i, o be the input and output places of the net and M_i, M_o be the initial and end markings. N is sound iff: 1) for every marking M reachable from M_i, there exists a firing sequence leading from M to M_o (Option to complete), 2) the marking M_o is the only marking reachable from M_i with at least one token in place o (Proper completion) and 3) for every transition $t \in T$, there is a marking M reachable from M_i such that $M[t\rangle$ (No dead transitions).*

This definition of soundness is very similar to the notion of soundness defined in Definition 1. The main difference is that Definition 13 refers to RWF-nets rather than eYAWL-nets. Note that reachability is not decidable for reset nets [12] 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 11[11–14]. Note that this even holds for reset nets with an infinite state space. We thus propose a weaker property called weak soundness property which can be decided using coverability results. Hence, the weak soundness property is decidable for an RWF-net. 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. Note that Definition 14 is closely related to Definition 2. The only difference is the type of model considered (RWF-net or eYAWL-net).

Definition 14 (Weak soundness). *Let $N = (P, T, F, R)$ be an RWF-net, i, o be the input and output places of the net and $M_i = i, M_o = o$ be the initial and end markings. N has weak soundness property iff: 1) M_o is coverable from M_i (Weak option to complete), 2) there is no marking M coverable from M_i such that $M > M_o$ (Proper completion) and 3) for every transition $t \in T$, there is a marking M coverable from M_i such that $M[t\rangle$ (No dead transitions).*

4 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 [21]. 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 [21]. 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 $N = (C, \mathbf{i}, \mathbf{o}, T, F, split, join, rem, nofi)$ be an eYAWL-net without OR-joins. $N' =$ transE2WF$(N) = (P, T', F', R)$ is an RWF-net.*

Figure 5 shows the RWF-net corresponding to the YAWL net in Figure 2. Places i and o represent 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}.

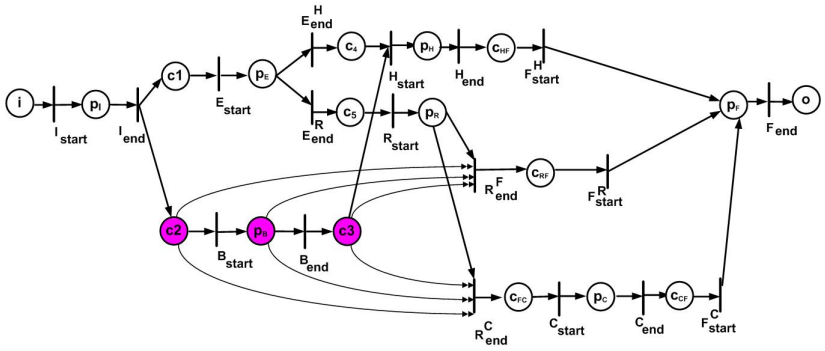

Fig. 5. Holiday scenario - RWF-net (note: double-headed reset arcs from $c2$, $c3$ and p_B to transitions R^F_{end} and R^C_{end})

A series of coverability questions will be asked to determine whether the weak soundness property of an RWF-net holds by using the three criteria defined in Definition 14. The **Coverable** procedure described in [21] is used to determine whether a marking is coverable from another marking of a reset net. The procedure makes use of the backwards firing rule as described in Section 3. We can made the following observations about an eYAWL-net without OR-joins by using coverability results from 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.*

To detect proper completion criterion, the test is whether there is a condition $c \in C$ such that $o + c$ is coverable from M_i. If one of these markings is found to be coverable then the net does not have proper completion.

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

To detect dead transitions, the test is whether a marking p_t is coverable from M_i for all transitions $t \in T$. Note that p_t represents the internal place for a transition in the eYAWL-net. Therefore, for any $t \in T$ if a marking p_t is coverable from M_i in the corresponding RWF-net, then t could be enabled in the eYAWL-net and t is not a dead transition.

Observation 3 *Given an eYAWL-net without OR-joins, no dead transitions 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 (Definition 1) is decidable.*
Proof: *Follows from observations 1, 2 and 3.*

Similarly, we can make the following observations regarding the existence of reducible cancellation regions.

Observation 4 *Given an eYAWL-net without OR-joins, a reducible condition c 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.*

Observation 5 *Given an eYAWL-net without OR-joins, a reducible task tx 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 with a finite state space, it is possible to decide the soundness property by generating a reachability graph for the corresponding RWF-net. Figure 6 shows the reachability graph for the reset net in Figure 5. It is easy to see whether an RWF-net satisfies the soundness property by testing the three criteria of Definition 13. If the corresponding RWF-net is sound, then the eYAWL-net without OR-joins is sound and if the corresponding RWF-net has the weak soundness property, then the eYAWL-net without OR-joins has the weak soundness property.

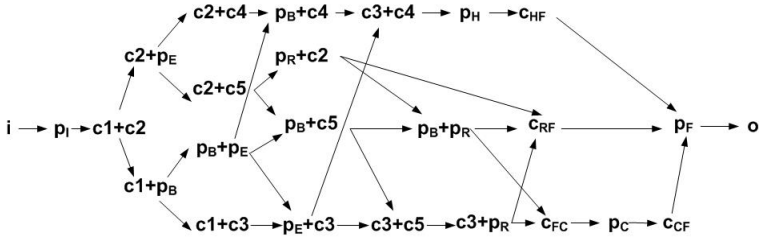

Fig. 6. Reachability graph for the RWF-net in Figure 5

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

To determine whether a net has infinite state space, we do not have a formal means of detecting this. We take a pragmatic approach and start the generation of reachable markings. If a certain duration has passed and the reachable markings generation is not completed, we decide that the model has infinite state space.

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. The informal semantics of OR-joins is that an OR-join task is enabled at a marking iff at least one of its input conditions is marked and it is not possible to reach a marking that still marks all currently marked input conditions (possibly with fewer tokens) and at least one that is currently unmarked. If it is possible to place tokens in the unmarked input conditions of an OR-join in the markings reachable from the current marking, then the OR-join task should not be enabled and wait until either more

input conditions are marked or until it is no longer possible to mark more input conditions [21]. Figure 7 shows a YAWL net with an OR-join task E. Task A is an OR-split task and after firing A, it is possible to enable any combination of tasks B, C and D. The possible reachable markings are $c1+c2+c3$, $c1+c2$, $c2+c3$, $c1+c3$, $c1$, $c2$ and $c3$. Let us consider the following occurrence sequence: $i \xrightarrow{A} c1+c2 \xrightarrow{C} c1+c5$. At the marking $c1+c5$, the OR-join enabling algorithm for E will be applied as there is a token in one of the input conditions for E ($c5$). This algorithm transforms the YAWL net into a reset net and checks whether $c5 + c6$ is coverable from the current marking $c1 + c5$ [21]. As it is not possible to reach a marking that marks both $c5$ and $c6$ from the current marking $c1 + c5$, task E is enabled. In this case, the effect is the same as if task E was an XOR-join task. Now, let us consider the following occurrence sequence instead: $i \xrightarrow{A} c1 + c2 + c3 \xrightarrow{C} c1 + c3 + c5$. At the marking $c1 + c3 + c5$, the OR-join enabling algorithm for E is called as there is a token in $c5$. As it is possible to reach $c1 + c5 + c6$ by firing task D first, E will wait for synchronisation and will not be enabled at the current marking $c1 + c3 + c5$. In this case, the effect is the same as if task E was an AND-join task. Detailed discussion on the formal semantics of OR-joins can be found in [21].

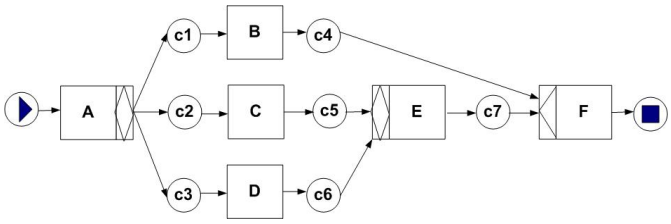

Fig. 7. A YAWL net with an OR-join task E

Due to the non-local semantics of OR-joins, a YAWL net with OR-joins cannot be mapped directly into a reset net [21]. Even though YAWL nets with OR-joins are difficult to define formally, they occur quite frequently in business scenarios and the correctness of these models is crucial. Therefore, it is important that we explore which properties can be detected for YAWL nets with OR-joins. To do this, we propose to translate all OR-joins in a YAWL net into XOR-joins first. This idea is based on the optimistic approach for treatment of OR-joins as shown in [21]. The treatment of OR-joins in the YAWL net as XOR-joins is considered optimistic as it assumes an OR-join can be enabled if there is at least one token in its preset. After replacing all OR-joins with XOR-joins, it is now possible to transform the YAWL net into an RWF-net using the transformation rule mentioned in Section 4.

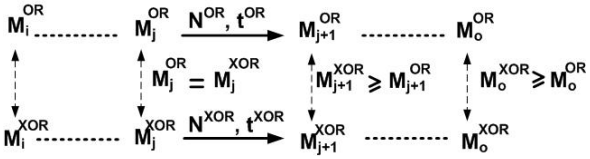

Fig. 8. Comparative markings between an eYAWL-net with an OR-join and the corresponding eYAWL-net with an XOR-join

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 the two eYAWL-nets 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 8 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^{XOR} and t^{OR} are enabled. When t^{OR} fires at M_j^{OR}, a token is removed from all marked conditions in its preset and we can reach M_{j+1}^{OR}. When t^{XOR} fires at M_j^{XOR}, a token is removed from one of the marked conditions in its preset and we can reach M_{j+1}^{XOR}. 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.

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

Note that this observation does not hold in the opposite direction. A counter example is given in Figure 9. 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: $i \xrightarrow{A} c1 + c2 \xrightarrow{B} c2 + c3 \xrightarrow{C} c3 + c4 \xrightarrow{D} c5$. 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: $i \xrightarrow{A} c1 + c2 \xrightarrow{B} c2 + c3 \xrightarrow{C} c3 + c4 \xrightarrow{D} c4 + c5 \xrightarrow{E} o$. Therefore, the translated net has the weak option to complete property when the original net with OR-joins does not.

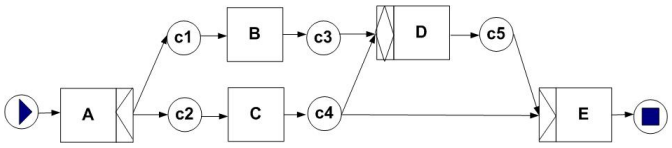

Fig. 9. This YAWL net with an OR-join task D always deadlocks at E.

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.*

Observation 10 *Given an eYAWL-net with OR-joins N, let N' be the corresponding eYAWL-net without OR-joins and RN be the equivalent RWF-net for N'. The following holds: 1) if RN does not have weak option to complete then N does not have weak option to complete, 2) if RN has dead transitions then N has dead transitions and 3) if RN has proper completion, then N has proper completion.*

For a YAWL net with OR-joins that has a finite state space, we propose to create a reachability graph by taking into account OR-join semantics. We can construct such a reachability graph using enabling and firing rules as defined in [6, 21]. Figure 10 shows a reachability graph for the YAWL net with OR-joins in Figure 7. From such a graph, it is possible to decide whether a YAWL net with OR-joins has the soundness property or not. In this example, the net does not have soundness property as a marking greater than o is reachable from i.

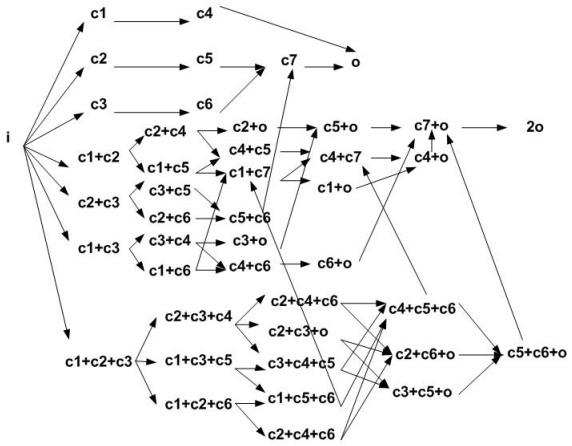

Fig. 10. Reachability graph for the YAWL net in Figure 7

Observation 11 *Given an eYAWL-net with OR-joins and a finite reachability graph, soundness and weak soundness properties are decidable.*

Using a finite reachability graph, we can also detect convertible OR-join tasks. Enabling markings for an OR-join task can be observed from the reachability graph of a YAWL-net. An OR-join task can be converted to an XOR-join task, if all enabling markings for that OR-join only mark one input condition and not more than one input conditions. On the other hand, an OR-join task can be converted to an AND-join task, if all enabling markings for that OR-join always mark all input conditions of the OR-join.

Observation 12 (XOR-join) *Given an eYAWL-net with OR-joins and a finite reachability graph, whether an OR-join could be an XOR-join can be decided by testing whether a reachable marking that marks exactly one input condition of the OR-join task enables the OR-join and a marking that marks more than one input conditions to the OR-join task is not in the set of reachable markings.*

Observation 13 (AND-join) *Given an eYAWL-net with OR-joins and a finite reachability graph, whether an OR-join could be an AND-join can be decided by testing whether a reachable marking that marks all input conditions to the OR-join task enables the OR-join and a reachable marking that marks less input conditions to the OR-join task is not an enabling marking for that OR-join.*

Theorem 3. *Given an eYAWL-net with OR-joins and a finite reachability graph, whether an OR-join task can be replaced by an XOR-join or an AND-join is decidable.*
Proof: *Follows from observations 12 and 13.*

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 have been downloaded by more than 9000 times. Both can be obtained via www.yawl-system.com.

We have extended the YAWL editor to support the verification approach presented in this paper. The verification function can perform four checks: the weak soundness property, the soundness property, reducible cancellation regions and convertible OR-joins. The designer can select appropriate options for a given specification. If a YAWL specification has hierarchical structure (i.e., contains multiple YAWL nets), diagnosis is given for every YAWL net individually. A typical usage scenario is as follows: a process designer uses the YAWL editor to describe a YAWL specification, performs verification, observes the outcomes, and then makes appropriate changes if necessary. Verification messages are shown either as warnings or observations. Figure 11 shows verification messages for the YAWL net of Figure 3. As reachability algorithm can only be used for specifications with finite state spaces, we allow for the setting of a maximal number of markings. We also detect *out of memory* error and it is taken as indication that the model possibly has infinite state space.

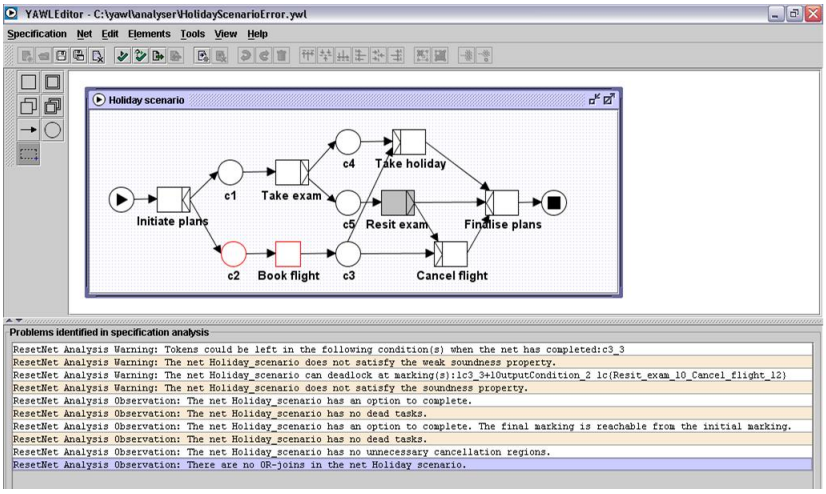

Fig. 11. Holiday Scenario with errors

7 Related Work and Conclusion

This paper focuses on the verification of YAWL specifications with and without OR-joins. The use of Petri nets for workflow verification have been studied before [1, 2, 5, 19]. In [3], the author describes how structural properties of a workflow net can be used to detect the soundness property. In [9], the authors discuss an approach for cyclic workflows using hierarchical decomposition. In [20], the authors present an alternative approach for deciding relaxed soundness property using invariants. The approach taken results in an approximation of 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.

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 [10–14]. We define four desirable properties for YAWL specifications: *weak soundness property*, *soundness property*, *reducible cancellation regions* and *convertible 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, weak soundness and reducible cancellation regions are decidable using coverability results for reset nets.
- For YAWL nets without OR-joins and a finite state space, weak soundness and soundness properties are 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 and convertible OR-joins are decidable using reachability results from YAWL without translating the net into reset nets.

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

References

1. W.M.P van der Aalst. Verification of workflow nets. In *Proceedings of Application and Theory of Petri Nets*, volume 1248 of *LNCS*, 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 *Business Porcess Management:Models, Techniques and Empirical Studies*, volume 1806 of *LNCS*, 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 *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 *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (27-31 May)*, volume 2348 of *LNCS*, 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. Y. Choi and J. Zhao. Decomposition-based Verification of Cyclic workflows. In *Proceedings of Automated Technology for Verification and Analysis(ATVA 2005) 4-7 October*, volume 3707 of *LNCS*, pages 84–98, Taipei,Taiwan, 2005. Springer-Verlag.
10. P. Darondeau. Unbounded Petri net Synthesis. In *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, volume 3098 of *LNCS*, pages 413–428, Eichstätt,Germany, 2003. Springer-Verlag.
11. C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset Nets Between Decidability and Undecidability. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, volume 1443 of *LNCS*, pages 103–115, Aalborg, Denmark, July 1998. Springer-Verlag.
12. C. Dufourd, P. Jančar, and Ph. Schnoebelen. Boundedness of Reset P/T Nets. In *Lectures on Concurrency and Petri Nets*, volume 1644 of *LNCS*, pages 301–310, Prague, Czech Republic, July 1999. Springer-Verlag.
13. 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.
14. A. Finkel and Ph. Schnoebelen. Well-structured Transition Systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, April 2001.
15. B. Kiepuszewski. *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows*. PhD thesis, Queensland University of Technology, Brisbane, Australia, 2003.
16. E. Kindler. On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. In *Proceedings of 2nd International Conference on Business Process Management*, volume 3080 of *LNCS*, pages 82–97, Potsdam, Germany, 2004. Springer–Verlag.
17. M. Leuschel and H. Lehmann. Coverability of Reset Petri Nets and other Well-Structured Transition Systems by Partial Deduction. In *Proceedings of Computational Logic 2000*, volume 1861 of *LNAI*, pages 101–115, London, UK, 2000. Springer-Verlag.
18. T. Murata. Petri nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, Apr 1989.
19. H.M.W. Verbeek. *Verification of WF-nets*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, June 2004.
20. H.M.W. Verbeek, A.H.M ter Hofstede, and W.M.P. van der Aalst. Verifying Workflows with Cancellation Regions and OR-joins: An Approach Based on Invariants. Technical Report BPM centre report, Eindhoven University of Technology, Eindhoven, The Netherlands, December 2005.
21. 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 *Proceedings of the 26th International conference on Application and Theory of Petri nets and Other Models of Concurrency (20 - 25 June)*, volume 3536 of *LNCS*, pages 423–443, Miami, USA, June 2005. Springer-Verlag.