AN APPROACH OF OPTIMISATION AND FORMAL VERIFICATION OF WORKFLOW PETRI NETS

OUTMAN EL HICHAMI, MOHAMMED AL ACHHAB, ISMAIL BERRADA,
RACHID OUCHEIKH, BADR EDDINE EL MOHAJIR

1, 3 University of Abdelmalek Essaâdi, Faculty of Sciences, Tetouan, Morocco
2 University of Abdelmalek Essaâdi, National School of Applied Sciences, Tetouan, Morocco
3, 4 University of Sidi Mohamed Ben Abdellah, Faculty of Sciences and Technology, Fez, Morocco

E-mail: 1 el.hichami.outman@taalim.ma, 2 alachhab@ieee.ma, 3 iberrada@univ-lr.fr,
4 cheikh.rachid09@gmail.com, 5 elmohajir@ieee.ma

ABSTRACT

In this paper, we are interested in the verification of Workflows (WF). WF are increasingly used for modeling and improving the quality of business processes (BP). In fact, the BP are becoming more complex because of the process nature which can be decomposed into sub-processes that run in parallel and/or sequentially. In this paper, we are particularly interested in verifying a class of properties known as response properties. Our second aim is to optimise the verification process of complex WF. Therefore, we first propose a method to compute and extract WFφ which is a part of WF concerned with the property φ to verify, and second we apply an abstraction method which optimises the size of WFφ. This optimisation is interesting so that the verification process can be done just on the optimise WFφ, and we deduce the preservation of the satisfaction of φ in WF.

Keywords: Workflow Petri nets, response properties, abstraction, formal methods

1. INTRODUCTION

In order to achieve a quality approach, the process approach is recommended for companies. In fact, business processes (BP) are typically associated with operational objectives and business relationships such as an engineering development process, insurance claims process, and web services [21]. The representation of a BP in a form which supports automated manipulation like modeling or enactment by a workflow system.

Workflows (WF) [7] have been established to manage the automatic execution of BP. A WF process is a set of activities which contains executable tasks that consume and produce well defined data. In this study, we are interested in using the formal methods of WF process validation. We distinguish two main existing verification interests. On the one hand, the WF analysis axes that consist of verifying general properties like absence of deadlocks, livelocks or no dead activities [5][9]. On the other hand, the use of verification framework to validate the dynamic behaviors of BP [11][18][19][20]. The last approach needs a translation framework from the WF schema towards a format and content that are recognized by the verification tool. The specification of the properties should also be expressed within a logic that is understood and supported by the verification tool.

In this work, we focus on an automatic and direct verification of a class of properties related to a WF processes. We are particularly interested in responses properties. This type of properties can be written as follows:

- Task A will always be followed by task B;
- Task A and task B will be executed in parallel and after task C;
- Task C will be executed after tasks A and B.

This paper discusses the use of Workflow Petri nets (WF-net) introduced by van der Aalst [6]. WF-net is an established tool based on Petri nets for modeling and analyzing the correctness of WF.

In the specification phase, we aim that the designer can use a graphical user interface based on the same concepts as established in WF-net to specify the properties to be verified. An optimal verification approach is proposed.

Our aim is the integration of the formal verification techniques in the design phase. First, the designer starts with modeling the Workflow, then he specifies the properties to verify, in this way constraint specifications can be automatically and
intuitively validated by the designer at this early stage. The famous drawback of this approach is the verification processes complexity which is still a problem when we use formal methods. In order to optimise the verification process, we propose to use the abstraction notion defined in [13]. This approach allows to group the tasks/places that run in parallel and/or sequentially. In [13] the authors state that this abstraction process preserves the liveness and boundedness properties in WF-net.

Our verification process based on the abstraction and optimisation of WF-net is defined as follows: first we specify the property $\varphi$ to be verified with the graphical user interface. The verification algorithm starts with an optimisation of WF-net model in order to keep just the part $WF-net_\varphi$ concerned with $\varphi$. Then, the abstraction process will be done only in this part $(WF-net_\varphi)$. Our approach aims to verify the property $\varphi$ on the optimised submodel of $WF-net_\varphi$, and conclude the satisfaction (or not) of $\varphi$ in the global WF-net.

The organization of this paper is as follows: Section 2 discusses the related work. Section 3 and 4 provide formal definitions of basics WF-net that are required in the rest of this paper. Section 5 describes a set of reduction for WF-net. We develop our approach of verification in Section 6. We also show the efficiency of our approach with experiments and analysis in Section 7. Section 8 concludes the paper, and draws perspectives.

2. RELATED WORK

Several techniques and methods are proposed to validate WF-net. Verbeek et al. [1] have developed a tool, called Woflan\(^1\), to verify soundness. However, the verification of large WF-net can be intractable due to the space explosion problem. The current initiative intends to remedy to this situation by proposing an abstraction and optimisation method of WF-net. YAMAGUCHI et al. [11] have also developed a tool which consists of WfNetEditor and WfNetAnalyzer that generates a WF-net as a PNML\(^2\) file which is an intermediate representation of WF-net. The check process converts the PNML file into PROMELA and uses LTL formulas to specify the properties. Finally, the method uses the SPIN model checker. This approach drastically reduces the complexity of verification processes. However, its use does not deal with the dynamic behavior of WF-net, and its limited only by the acyclic WF-net.

Some works have contributed to the verification of dynamic behavior of BP [15] [12]. This verification process is always based on existing verification tool, which implies the transformation of WF-net in the language of verification supported by the verification tool and specification of properties to be verified by temporal logic.

In [16] the authors have implemented a concept proof of their approach with existing software namely the open modeling platform Oryx [14] and the BPMN-Q query language [10]. This approach is based on the decomposition of BPMN-Q. However, these approaches fail because they do not satisfy all properties of a query and the decomposition of a BPMN-Q query is complicated.

The above works show that the verification phase comes after the design phase, and the knowledge of the logic used for the specification of properties is needed.

Our approach has two advantages: (1) integrate the verification process in the design stage allowing a gradual validation of BP. This phase can be avoided by implementing a graphical specification interface. (2) exploit the abstraction and the optimisation of WF-net to remedy the problem of the complexity of the verification process.

3. BASICS OF PETRI NETS

Petri nets [2] are largely used as tool for representation, validation and verification of Workflows [5][6][9]. In this section, we give the basic definitions, notations of Petri nets and main results of their structure theory used in this work.

3.1 Petri nets

A Petri net is a tuple $N = (P,T,F)$ where:

1. $P \neq \emptyset$ is a finite set of places;
2. $T \neq \emptyset$ is a finite set of transitions with $P \cap T \neq \emptyset$;
3. $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation.

A place can contain zero or more tokens. A token is represented by a black dot. The global state of a Petri net, also called a marking, is the distribution of tokens over places. Formally, a marking of a Petri net $N$ is a function $M : P \rightarrow \mathbb{N}$. The initial marking of $N$ is denoted by $M_0$.

3.2 Place and Transition

Let $N = (P,T,F)$ be a Petri net, we define a function $f : F \rightarrow \{0,1\}$ such that:

$$f(x,y) = \begin{cases} 1 & \text{if } (x,y) \in F \\ 0 & \text{otherwise} \end{cases}$$
For each transition or place $x$ we call a set $\bullet x = \{ y \in P \cup T \mid f(x, y) = 1 \}$ the preset of $x$, and a set $\bullet x' = \{ y \in T \mid f(x, y) = 1 \}$ the postset of $x$.

We now introduce some additional notations used in the present paper. A transition $t \in T$ is enabled at a marking $M$ if $\forall p \in \bullet t : M(p) \geq 1$. If a transition $t$ is enabled in marking $M$, it can be fired leading to a new marking $M'$ written as $M \xrightarrow{t} M'$ such that:

$$M'(p) = \begin{cases} M(p) + 1 & \text{if } p \in \bullet t \\ M(p) - 1 & \text{if } p \in \bullet t' \\ M(p) & \text{otherwise} \end{cases}$$

Given $w = t_1 \ldots t_n \in T^n$, we write $M_0 \xrightarrow{w} M$ if $M = M_0$ or there exist markings $M_1, \ldots, M_n$ such that $M_{n-1} \xrightarrow{t_{n-1}} M_n$. Then we say that $M$ is reachable. The set of reachable markings of the Petri net $N$ is denoted by $[M]$ and defined by $[M] = \{ M : \exists w \in T \mid M_0 \xrightarrow{w} M \}$.

A Petri net $N$ is live iff for every reachable marking $M$, and every transition $t$, there is a marking $M_t$ reachable from $M_t$, which enables $t$.

A Petri net $N$ is bounded iff for each place $p$ there is a natural number $n$ such that for every reachable marking, the number of tokens in $p$ is less than $n$.

### 3.3 Routing constructs

The Petri net process definition defines three routing constructs used to specify the relationships between tasks during the process execution: sequential, parallel, and conditional [6], (see Figure 1).

![Routing constructs](image)

### 4. WORKFLOW PETRI NETS (WF-NET)

WF-net is a restriction of Petri net for modeling, verifying and performance evaluation of workflows.

#### 4.1 Definition

A Petri net is called a WF-net [6] if it has one input place $i$ and one output place $o$ without input and output transitions. For every transition or place $x \in P \cup T$, there exists a path from $i$ to $x$ and a path from $x$ to $o$.

Formally, a Petri net $N = (P, T, F)$ is a WF-net if:

- $3i, o \in P$ such that $\bullet i = o = \emptyset, M_0(i) = 1$ and $\forall p \in P - \{i\} : M_0(p) = 0$ and $\forall x \in P \cup T, (i, x) \in F$ and $(x, o) \in F$. The resulting Petri net is strongly connected.

Normally, a WF-net should have a soundness property which guarantees a logical correctness of the modeled workflow. A WF-net is intuitively said to be sound iff, for any case, the initial place is transformed to the final place and there are no dead transitions [17].

#### 4.2 Definition

For further clarification, we give a simple example of the handling of a questionnaire [17]:

![Example of WF-net: handling of a questionnaire](image)

In this example, the WF-net starts with a registration step after which two parallel branches are started. The top branch is concerned with the handling of a questionnaire. After sending the questionnaire to the customer who submitted the complaints, there are two possibilities. The customer may return the questionnaire on time and subsequently it is processed. Otherwise, a timeout occurs and this step is skipped. In the lower branch of the WF-net, the complaint is first processed. After this, the result is evaluated. Based on this evaluation, the complaint is either checked or not. If it is checked, the result may be OK or not. If it is not OK, the complaint is processed again. This is repeated until no check is needed or the check is...
OK. Finally, after completing both parallel branches, the complaint is archived.

5. ABSTRACTION OF WF-NET

In this section, we present the rules which we use for the abstraction of some routing constructs of WF-net [13]. Our goal is to apply these reduction rules in order to group some transitions and places with preserving the essential properties of WF-net. As a result, the complexity of the verification process is reduced and can be performed more efficiently.

In the following, we denote \( AbsN = (P_a, T_a, F_a) \) the abstract WF-net of \( N = (P, T, F) \), and we use the following four operations of abstraction:

5.1 Abstraction of Series Transitions

The rule allows for the merging of two sequential transitions \( t \) and \( u \) with one place \( p \) in between these two transitions into only one transition \( v \) (see Figure 3).

Formally, let \( T \subseteq \Gamma \) be the transitions where \( |\Gamma| \geq 2 \), and \( F_a = (F \cap ((P_a \times T_a) \cup (T_a \times P_a))) \cup (\bullet t \times \{v\}) \cup (\{v\} \times \bullet u) \cup (\{v\} \times \bullet p) \).

5.2 Abstraction of Series Places

The rule allows for the merging of two sequential places \( p \) and \( q \) with one transition \( t \) in between them into a single place \( r \) (see Figure 4).

Formally, let \( t \in T \) be a transition and \( p, q \in P \) be two places, and \( r \in P_a \setminus P \) be a place such that:

5.3 Abstraction of Parallel Transitions

The rule allows for the merging of multiple transitions (at least two) that have the same inputs and outputs into a single transition (see Figure 5).

Formally, let \( \Gamma \subseteq T \) be the transitions where \( |\Gamma| \geq 2 \), and \( v \in T_a \setminus T \) be a transition such that:

5.4 Abstraction of Parallel Places

The rule allows for the merging of two parallel places \( p \) and \( q \) into a single place \( r \) (see Figure 6).
5.4 Abstraction of Parallel Places

The rule allows for the merging of multiple places (at least two) with the same inputs and outputs into a single place \( q \) (see Figure 6).

Formally, let \( Q \subseteq P \) be the places where \( |Q| \geq 2 \), and \( q \in P \setminus P \) be a place such that:

1. \( \forall p_i, p_j \in Q : p_i = p_j \) (input transitions for all places in \( Q \) are identical);
2. \( \forall p_i, p_j \in Q : p_i = p_j \) (output transitions for all places in \( Q \) are identical).

Construction of \( \text{AbsN} \):

1. \( P_a = (P \setminus Q) \cup \{q\} \);
2. \( T_a = T \);
3. \( F_a = (F \cap (P_a \times T_a) \cup (T_a \times P_a)) \cup (\bullet P \setminus \{q\}) \)

It is not difficult to see that the previous abstractions preserve the properties of liveness and boundedness [13]. That is, let \( N = (P, T, F) \) and \( \text{AbsN} = (P_a, T_a, F_a) \) be two \( WF\)-net before and after the previous reductions. Then \( \text{AbsN} \) is live (respectively bounded) iff \( N \) is live (respectively bounded).

6. VERIFICATION PROCESS

In this section, we present our verification process of \( WF\)-net. After the modelisation of \( BP \) to \( WF\)-net which will be verified, the designer can use a graphical interface to specify the property \( \varphi \) to validate. First we apply our algorithm (see Algorithm 1) to keep just the part of \( WF\)-net, concerned with this property. Second, we apply the abstraction rules in order to optimise \( WF\)-net, After that, the existing verification algorithms can be applied. Finally, we conclude that the satisfaction (or not) of \( \varphi \) in the original \( WF\)-net. This approach is presented in Figure 6:

6.1 Response properties models

In this section, we first give three models of response properties to specify the dynamic behavior of the \( BP \). Then we will also discuss some possible semantics of these models.

The first property (\( \Phi_1 \)) states that task \( t_i \) will always be followed by task \( t_j \). The second property (\( \Phi_2 \)) states that task \( t_j \) and task \( t_k \) will be executed in parallel and after task \( t_i \). The third property (\( \Phi_3 \)) states that task \( t_k \) will be done after the end of task \( t_i \) and task \( t_j \).

The semantics\(^3\) of \( \Phi_1, \Phi_2, \) and \( \Phi_3 \) can be interpreted in different ways. In this paper we focus on the implementation and the verification of LTL formulas [3]. This type of properties can be written as follows:

\[
- (\Phi_1) \quad \{ t_i \text{ is reachable from } t_j \} \quad \text{([LTL formula])}
\]

\[
- (\Phi_2) \quad \{ t_j \text{ and } t_k \text{ will be reachable from } t_i \} \quad \text{([LTL formula])}
\]

\[
- (\Phi_3) \quad \{ t_k \text{ will be reachable from } t_i \text{ and } t_j \} \quad \text{([LTL formula])}
\]

\(^3\) In temporal logic, \( \square \): Eventually in the future, \( [] \): Now and forever in the future, \( \square n \): Eventually in the future after \( n \) steps, and \( \square m \geq n \): Eventually in the future after \( m \) steps (\( m \geq n \)).
6.2 Optimisation and abstraction of WF-net

In this section, we present the process proposed to integrate the optimisation and abstraction in the formal verification of WF-net.

After the modelisation of the WF-net, the designer can specify the property $\Phi$ to be verified. The verification process starts with the optimisation of WF-net in order to keep just the part $WF$-net$_{\Phi}$ concerned with $\Phi$. Then, the abstraction process will be done only in this part ($WF$-net$_{\Phi}$). Figure 8 schematizes this approach.

![Figure 8: Our approach of optimisation and abstraction](image)

6.2.1 Construction algorithm of the part concerned with $\Phi$1

Algorithm 1

Data Original $WF$-net $N = (P, T, F)$, $t_i, t_j$: the extremities tasks in $\Phi$1

Result Part of $WF$-net: $N_{\Phi1} = (P_{\Phi1}, T_{\Phi1}, F_{\Phi1})$ related to $\Phi$1

begin
for $t_i \in T$ do
Mark($t_i$) = false /*Mark $t_i$ as not visited*/
endfor

Function CreatePart($WF$-net $N$, Task $t$)
begin
Mark($t$) /*Mark the $t$ as explored */
/ * while: select only the $p$ places adjacents to $t$ */
while $p$ such as $f(t, p) = 1$ do
/* for: select only the transitions adjacent to $p$ */
for $t'$ (postset $p$) do
if NotMark($t'$) then
Mark($t'$)
if $t' = t$ then
Add to $P_{\Phi1}$ all the places between ($t_i, t_j$)
Add to $T_{\Phi1}$ all the transitions between ($t_i, t_j$)
Add to $F_{\Phi1}$ all the relations between ($t_i, t_j$)
endif
CreatePart($N, t'$)
endif
endfor
endwhile
end

end

After the construction of $WF$-net $N_{\Phi1}$, and in order to keep the same behavior of $WF$-net $N$, it is necessary to add some transitions and/or places to $WF$-net $N_{\Phi1}$. There are four cases to study: places (respectively transitions) which have the links with outside transitions (respectively places) and not in $WF$-net $N_{\Phi1}$ (see bold arcs in Figure 9):

1. Or-split (arc 1 in Figure 9) which connects a place $p$ in $P_{\Phi1}$ to a transition $t$ in $T$ and not in $T_{\Phi1}$. In this case, we must add $t$ in $T_{\Phi1}$ in order to keep the alternative tasks in $WF$-net $N_{\Phi1}$.
2. Or-join (arc 2 in Figure 9) which connects a transition $t$ in $T$ and not in $T_{\Phi1}$ to a place $p$ in $P_{\Phi1}$. In this case, it is not necessary to add $t$ in $T_{\Phi1}$ because in our case, we are only interested in paths that start from $t_i$ and which have $t_j$ as target termination.
3. And-split (arc 3 in Figure 9) which connects a transition $t$ in $T_{\Phi1}$ to a place $p$ in $P$ and not in $P_{\Phi1}$. In this case, it is not necessary to add $p$ in $P_{\Phi1}$ because (1) in our case, we are only interested in paths $w$ between $t_i$ and $t_j$ and (2) we are not concerned with parallel routing which doesn't influence the $w$.
4. And-join (arc 4 in Figure 9) which connects a place $p$ in $P$ and not in $P_{\Phi1}$ to a transition $t$ in $T_{\Phi1}$. In this case, it is not necessary to add $p$ in $P_{\Phi1}$. In fact, the transition will be executed because in our work we consider that $WF$-net $N$ is sound.

In the following, we formally show how to add these transitions to $N_{\Phi1}$: Let $WF$-net $N = (P, T, F)$, $\Phi$1: $t_i \rightsquigarrow t_j$ be the property to be verified, and $N_{\Phi1} = (P_{\Phi1}, T_{\Phi1}, F_{\Phi1})$ be the $WF$-net related to $WF$-
net $N$ after the optimisation related to $\Phi_1$.

Formal definition of construction of WF-net $N_{\Phi_1}$:
1. Add the output transitions for all the Or-split places:
   $$\forall p \in P_{\Phi_1} \mid p^* = \{t_1, \ldots, t_m\} \in \left( T \setminus T_{\Phi_1} \right)$$
   $$T_{\Phi_1} = T_{\Phi_1} \cup \{t_j \mid j \leq m\};$$
   $$F_{\Phi_1} = F_{\Phi_1} \cup \{\{(p) \mid t_j \}\mid j \leq m\}.$$  

2. Add the input place $p_i$ and the output place $p_o$:
   $$P_{\Phi_1} = P_{\Phi_1} \cup \{p_i, p_o\};$$
   $$F_{\Phi_1} = F_{\Phi_1} \cup \{(p_i \times t_j) \cup (t_j \times p_o) \cup \{(p) \mid t_j \} \mid j \leq m\}.$$  

Remark: The construction of the WF-net concerned with both $\Phi_2$ and $\Phi_3$

To build the part concerned with the property $\Phi_2$, we apply the Algorithm 1 twice by changing the target extremities of $\Phi_2$. The same principle to create the part related to $\Phi_3$, but this time by modifying the source extremities of $\Phi_3$.

6.2.2 Example of $\Phi_1$

Let $\phi$ be a property of type $\Phi_1$ such that:
- $\Phi_1$: After “register”, “OK” must be reachable.

We now present how to create $N_{\phi_1} = (P_{\phi_1}, T_{\phi_1}, F_{\phi_1})$ related to WF-net $N = (P, T, F)$ of Figure 2 (the handling of a questionnaire). For this, we apply the Algorithm 1 to build the part concerned with $\phi_1$. After that, we add the transition $t_8$ to the Or-split place $p_3$. Finally, we add the input place $p_i$ and the output place $p_o$. Therefore, the WF-net $N_{\phi_1}$ is as follows:

![Figure 10: The WF-net $N_{\phi_1}$ related to $\phi_1$ after the optimisation of WF-net $N$](image1)

After the construction of WF-net $N_{\phi_1}$, and in order to optimise more the size of WF-net $N_{\phi_1}$, we perform the abstraction process in the part concerned with $\phi_1$. The result is the WF-net $\text{Abs}N_{\phi_1}$ (presented in Figure 11).

![Figure 11: The WF-net $\text{Abs}N_{\phi_1}$ related to Figure 10 after the abstraction](image2)

6.2.3 Example of $\Phi_2$ and $\Phi_3$

Let $\phi_2$ (respectively $\phi_3$) be a property of type $\Phi_2$ (respectively $\Phi_3$)

- $\Phi_2$: After “register”, “returned questionnaire” and “evaluate” must be executed in parallel.
- $\Phi_3$: “archive” is done after the end of “time-out” and “returned questionnaire”.

Figure 12 shows the parts concerned with $\phi_2$ and $\phi_3$:

![Figure 12: Parts concerned with $\phi_2$ and $\phi_3$](image3)

After the construction of WF-net $N_{\phi_2}$ and WF-net $N_{\phi_3}$, the abstraction process can be done on the parts concerned with $\phi_2$ and $\phi_3$. The result is the WF-net $\text{Abs}N_{\phi_2}$ and the WF-net $\text{Abs}N_{\phi_3}$ (see Figure 13).

![Figure 13: The WF-net $\text{Abs}N_{\phi_2}$ and $\text{Abs}N_{\phi_3}$ after the abstraction parts concerned with $\phi_2$ and $\phi_3$](image4)
6.3 Proof of satisfaction of $\Phi_1$, $\Phi_2$ and $\Phi_3$ in The WF-net

In this section, we prove that if each of the three response properties models are satisfied in the WF-net $\text{Abs}N$ after the optimisation and the abstraction of the original WF-net $N$ related to these properties, then we conclude that the satisfaction in the original one.

In our context we consider that WF-net $N$ is sound. So, every transition in $T$ will be enabled (because no deadlock transitions in WF-net $N$ and every transition is reachable from $M_0$, the initial marking of WF-net $N$) which means that the source extremities of the transitions $t_i$, $t_j$ and $t_k$ in $\Phi_1$, $\Phi_2$ and $\Phi_3$ will be enabled.

6.3.1 Proof of $\Phi_1$

Let $N = (P, T, F)$ be a WF-net, $\Phi_1$: $t_i \leadsto t_j$ be a property to verify, and $\text{Abs}N_{\Phi_1} = (P_{\Phi_1}, T_{\Phi_1}, F_{\Phi_1})$ be the WF-net after the optimisation and the abstraction of WF-net $N$ related to $\Phi_1$.

We prove that if the property $\Phi_1$ is satisfied in WF-net $\text{Abs}N_{\Phi_1}$, then we conclude that $\Phi_1$ is satisfied in WF-net $N$.

**Lemma 1.** WF-net $\text{Abs}N_{\Phi_1} \models \Phi_1 \Rightarrow WF-net\ N \models \Phi_1$

**Proof.** According to the semantic of $\Phi_1$ in LTL context, WF-net $\text{Abs}N_{\Phi_1} \models \Phi_1 \Rightarrow \forall$ paths $w = t_1 \ldots t_j \in T_{\Phi_1}$ i.e. $M(t_i, \bullet)^w \rightarrow M(t_j, \bullet)$. Because the optimisation and the abstraction keep all the possible paths between $t_i$ and $t_j$ in WF-net $\text{Abs}N_{\Phi_1}$. Furthermore, WF-net $N$ is sound then no deadlock transitions $\Rightarrow \forall w ' = t_1 \ldots t_j \in T$ and $w \subseteq w'$. Hence, WF-net $N \models \Phi_1$.

6.3.2 Proof of $\Phi_2$

Let $N = (P, T, F)$ be a WF-net, $\Phi_2$: $t_i \leadsto \leq_{t_k}$ be a property to verify, and $\text{Abs}N_{\Phi_2} = (P_{\Phi_2}, T_{\Phi_2}, F_{\Phi_2})$ be the WF-net after the optimisation and the abstraction of WF-net $N$ related to $\Phi_2$.

We prove that if the property $\Phi_2$ is satisfied in WF-net $\text{Abs}N_{\Phi_2}$, then we conclude that $\Phi_2$ is satisfied in WF-net $N$.

**Lemma 2.** WF-net $\text{Abs}N_{\Phi_2} \models \Phi_2 \Rightarrow WF-net\ N \models \Phi_2$

**Proof.** According to the semantic of $\Phi_2$ in LTL context, $\text{Abs}N_{\Phi_2} \models \Phi_2 \Rightarrow \forall$ paths $w = t_1 \ldots t_j$ and $\forall$ paths $w_2 = t_{i_1} \ldots t_{i_k} \in T_{\Phi_2}$ i.e. $M(t_i, \bullet)^{w_2} \rightarrow M(t_j, \bullet)$ and $M(t_i, \bullet)^{w_2} \rightarrow M(t_k, \bullet)$.

Because the optimisation and the abstraction keep all the possible paths between $t_i$ and $t_k$ between $t_i \ldots t_k$ in WF-net AbsN$_{\Phi_2}$. Furthermore, WF-net $N$ is sound then no deadlock transitions $\Rightarrow \forall w = t_1 \ldots t_j$ and $\forall w '' = t_{i_1} \ldots t_{i_k}$ $\in T^2$ and $w \subseteq w'', w_2 \subseteq w''$. Hence, WF-net $N \models \Phi_2$.

6.3.3 Proof of $\Phi_3$

Let $N = (P, T, F)$ be a WF-net, $\Phi_3$: $t_i \leadsto \leq_{t_k}$ be a property to verify, and $\text{Abs}N_{\Phi_3} = (P_{\Phi_3}, T_{\Phi_3}, F_{\Phi_3})$ be the WF-net after the optimisation and the abstraction of $N$ related to $\Phi_3$.

We prove that if the property $\Phi_3$ is satisfied in $\text{Abs}N_{\Phi_3}$, then we conclude that $\Phi_3$ is satisfied in $N$.

**Lemma 3.** WF-net $\text{Abs}N_{\Phi_3} \models \Phi_3 \Rightarrow WF-net\ N \models \Phi_3$

**Proof.** According to the semantic of $\Phi_3$ in LTL context, WF-net $\text{Abs}N_{\Phi_3} \models \Phi_3 \Rightarrow \forall$ paths $w_1 = t_1 \ldots t_i$ and $\forall$ paths $w_2 = t_j \ldots t_k \in T_{\Phi_3}$ i.e. $M(t_i, \bullet)^{w_1} \rightarrow M(t_j, \bullet)$ and $M(t_j, \bullet)^{w_2} \rightarrow M(t_k, \bullet)$.

Because the optimisation and the abstraction keep all the possible paths between $t_i$ and $t_k$ between $t_i \ldots t_k$ in WF-net $\text{Abs}N_{\Phi_3}$. Furthermore, WF-net $N$ is sound then no deadlock transitions $\Rightarrow \forall w = t_1 \ldots t_k$ and $\forall w '' = t_{i_1} \ldots t_{i_k} \in T^2$ and $w \subseteq w'', w_1 \subseteq w''$. Hence, WF-net $N \models \Phi_3$.

7. EXPERIMENTS AND ANALYSIS

In this paper, we use the SPIN\textsuperscript{4} tools to validate our proposal and to assure the theoretical results. In these experiments, we discuss the verification of the three models of response properties $\Phi_1$, $\Phi_2$ and $\Phi_3$. The results of this analysis show the performance of our approach.

7.1 Transforming WF-net to PROMELA

In [8], the authors have proposed a method to describe a WF-net into PROMELA\textsuperscript{5} that can be simulated and verified with the SPIN model checker. In this method, a WF-net system is represented as a single process. The process describes each firing of its transitions.

Program 1 presents an outline of the PROMELA program for WF-net (handling of a questionnaire).

7.2 LTL formulas

The properties to be verified in SPIN have to be expressed as LTL formulas. LTL formulas correspond to the response properties $\phi_1$, $\phi_2$ and $\phi_3$ to be verified and can be rewritten as follows:

$\phi_1$: After "register", "OK" must be reachable. 
$\lbrack (M[1]>=1) \Rightarrow \lbrack (M[6]>=1) \rbrack \rbrack$.

$\phi_2$: After "register", "returned questionnaire" and "evaluate" must be executed in parallel.
$\lbrack (M[0]>=1) \Rightarrow ((M[2]>=1 \& \& (M[8]>=1)) \rbrack \rbrack$.

$\phi_3$: "archive" is done after the end of "time-out" or "returned questionnaire".
$\lbrack (M[9]>=1) \Rightarrow ((M[11]>=1)) \rbrack \rbrack$.

\textsuperscript{4} http://spinroot.com

\textsuperscript{5} spinroot.com/spin/Man/promela.html
7.3 Experimental results

In this section we give some statistics in order to show the performance of our approach. We compare the size, the memory and the verification time between the original WF-net related to Figure 2 and the WF-net: AbsNφ1, AbsNφ2 and AbsNφ3.

We first give the results without optimisation and abstraction of WF-net related to Figure 2.

Table 1: Experiments without optimisation and abstraction.

<table>
<thead>
<tr>
<th>LTL formula</th>
<th>States stored</th>
<th>Transition s</th>
<th>Memory (Mb)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>φ1</td>
<td>15</td>
<td>61</td>
<td>11.876</td>
<td>0.18</td>
</tr>
<tr>
<td>φ2</td>
<td>16</td>
<td>52</td>
<td>12.364</td>
<td>0.21</td>
</tr>
<tr>
<td>φ3</td>
<td>12</td>
<td>40</td>
<td>11.485</td>
<td>0.18</td>
</tr>
</tbody>
</table>

To evaluate the effect of our approach, we perform the verification of properties φ1, φ2 and φ3 in the AbsNφ1, the AbsNφ2 and the AbsNφ3. These experiments are given in Table 2.

Table 2: Experiments with optimisation and abstraction.

<table>
<thead>
<tr>
<th>LTL formula</th>
<th>States stored</th>
<th>Transition s</th>
<th>Memory (Mb)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>φ1</td>
<td>8</td>
<td>12</td>
<td>3.770</td>
<td>0.02</td>
</tr>
<tr>
<td>φ2</td>
<td>10</td>
<td>23</td>
<td>3.672</td>
<td>0.03</td>
</tr>
<tr>
<td>φ3</td>
<td>5</td>
<td>8</td>
<td>3.522</td>
<td>0.02</td>
</tr>
</tbody>
</table>

For more clarification, Figure 14 illustrates this comparison between the original WF-net related to Figure 2 and the WF-net: AbsNφ1, AbsNφ2 and AbsNφ3.

8. CONCLUSION

This paper proposes a new approach to verify the WF-net in a BP context. We exploited the optimisation and the abstraction methods to remedy the verification processes complexity which is still a problem when we use a formal method. We are particularly interested in the response properties, and we present three class of these properties.

We propose an algorithm for extracting a part from WF-net in order to build an abstract WF-net. After this abstraction, we performed an optimisation in abstract WF-net to optimise more the size of WF-net. This method has the advantage of preserving the dynamic behaviors of WF-net.

Our approach is based on the verification of the response property on a part of the WF-net which is concerned by this property, and we proved that if the optimised WF-net satisfies this property, then we deduced the validation in the global WF-net. This proof is enriched by several practical experiences with the SPIN tools in order to show the performance of our approach.

---

Theorem 1:

Let $\phi$ be a LTL formula and $\mathcal{N}$ be a WF-net. If $\mathcal{N}$ satisfies $\phi$, then the abstracted WF-net $\mathcal{N}'$ also satisfies $\phi$.

Proof:

By induction on the structure of $\phi$. The theorem is trivial for atomic propositions, and we assume it holds for $\phi_1$ and $\phi_2$. Let $\phi = \phi_1 \land \phi_2$. By induction, $\mathcal{N}$ satisfies $\phi_1$ and $\mathcal{N}$ satisfies $\phi_2$. Since $\mathcal{N}$ satisfies $\phi_1$ and $\phi_2$, it follows that $\mathcal{N}$ satisfies $\phi$. 

---

Verifiable with disable dead-variable elimination
For now, we study the possibility to facilitate the conditions of abstraction in order to better optimise the size of abstract WF-net, and we will investigate the semantic expansions of the response properties.

REFERENCES: