Process Tree-based Analysis Method of DECLARE Relation Constraints in Acyclic Bridge-less Well-structured Workflow Nets

Shingo Yamaguchi[1] and **Mohd Anuaruddin Bin Ahmadon**[2]

ABSTRACT

In this paper, we proposed a method to analyze workflows' constraints whose templates are defined in a declarative language called DECLARE. Checking such constraints is important but known to be intractable in general. Our results show three things. First, utilizing a tree representation of workflow process called *process tree*, we provided necessary and sufficient conditions on the constraints. Second, those conditions enable us to not only check a given constraint in polynomial time but also find a clue for improving the net if it violates the constraint. Third, we revealed the relationship among the constraint templates.

Keywords: Petri Net, Workflow Net, DECLARE, Linear Temporal Logic (LTL), Process Tree, Constraint

1. INTRODUCTION

Workflow nets [1] (WF-nets for short) are a subclass of Petri nets [2] and are widely used for modeling and analysis of workflows. A WF-net precisely specifies how to execute every task. Due to taking all possibilities into account, the WF-net tends to become large and complicated.

Pesic *et al.* [3] proposed a different approach called *DECLARE* [4] to allow users to specify constraints for only tasks in which they are interested. DECLARE is a declarative language for modeling loosely-structured processes. It provides templates of constraints based on Linear Temporal Logic (LTL for short) semantics. The templates are classified into four groups: existence, relation, negative relation, and choice [5]. DECLARE can export constraints to LTL checkers. This enables us to check whether a given process model satisfies the constraints by means of model checking.

Ab Malek *et al.* [6] defined a property called *response* and investigated its decision problem. This property has almost the same definition as one of the DECLARE relation templates. They proved that the problem is intractable for a subclass of WF-nets called *acyclic asymmetric choice WF-nets*. They also proposed the necessary and sufficient condition to solve the problem for its subclass called *acyclic bridge-less well-structured WF-nets*. The advantage of this condition is verifiable in polynomial time by utilizing a tree representation of a process called *process tree*. But unfortunately, the other DECLARE relation templates have not been investigated at all.

In this paper, we propose a method to analyze workflows' constraints of the DECLARE relation templates. We develop Ab Malek *et al.*'s process tree-based approach in order to provide necessary and sufficient conditions on the constraints. Those conditions enable us to not only check a given constraint in polynomial time but also find a clue for improving any net violating the constraint. We also reveal the relationship among the templates. The rest of this paper is organized as follows: Section 2 introduces WF-nets, process trees, and DECLARE. Section 3 describes the analysis method. Section 4 illustrates the proposed analysis method with an application example and shows the usefulness of the method. Section 5 gives the conclusion and suggests future work.

2. PRELIMINARY

2.1 Workflow Nets

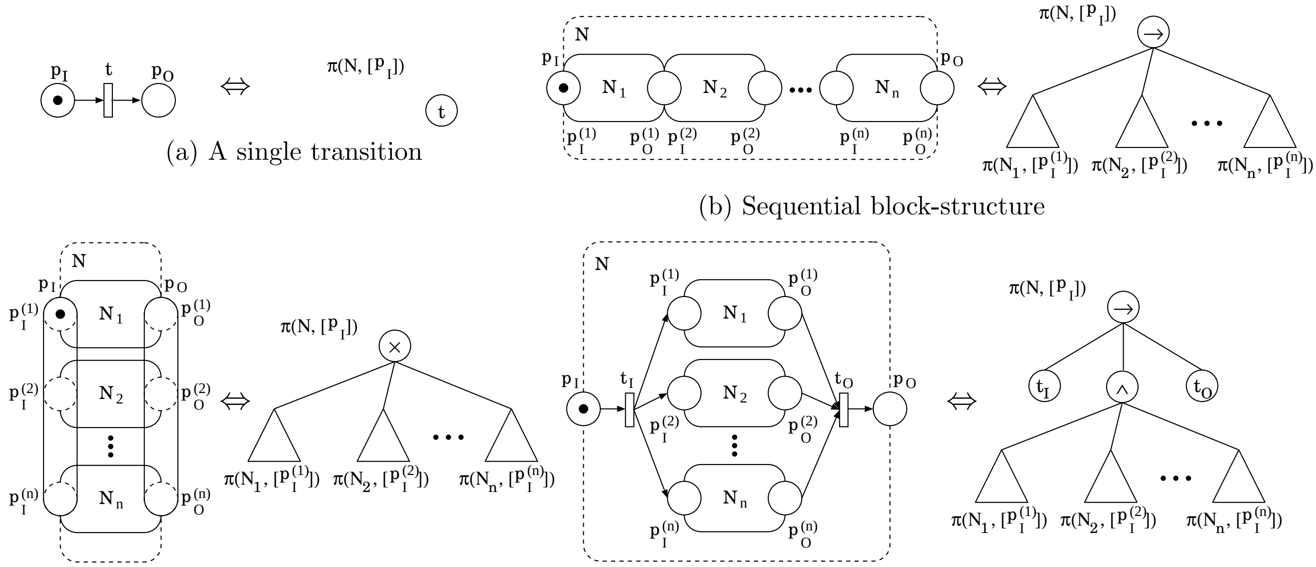
Petri nets are a mathematical tool applicable to various systems [7]. An ordinary Petri net PN is a three tuple (P, T, A), where P, T $(\cap P{=}\emptyset)$, and A $(\subseteq(P{\times}T)\cup(T{\times}P))$ are respectively finite sets of places, transitions, and arcs. For a node (a place or a transition) x, $\bullet x$ and $x\bullet$ respectively denote $\{y|(y,x){\in}A\}$ and $\{y|(x,y){\in}A\}$. A WF-net is a Petri net which represents a workflow process. PN is said to be a WF-net if (i) N has a single source place p_I and a single sink place p_O and (ii) every node is on a path from p_I to p_O. $\overline{N}$ denotes the Petri net obtained by connecting from p_O to p_I via an additional transition t^* and is called the short-circuited net of N. A marking of N is a mapping $M\colon P{\to}\mathbb{N}$. We represent M as a bag over P, i.e. $M{=}[p^{M(p)}|p{\in}P, M(p){>}0]$. $(N, [p_I])$ denotes N with the initial marking $[p_I]$. A transition t is said to be firable in M if $M{\geq}\bullet t$. Firing t in M results in a new marking M' $({=}M{\cup}t\bullet{\setminus}\bullet t)$.

We say a path from a node x to a node y is an

Manuscript received on August 5, 2019 ; revised on October 6, 2019.

Final manuscript received on October 7, 2019.

[1,2] The authors are with the Graduate School of Sciences and Technology for Innovation, Yamaguchi University and 2-16-1 Tokiwadai, Ube-shi, Yamaguchi 755-8611, Japan, E-mail: shingo@yamaguchi-u.ac.jp and anuar@yamaguchi-u.ac.jp



(a) A single transition

(b) Sequential block-structure

(c) Exclusive-choice block-structure

(d) Parallel block-structure

Fig.1: *Definition of the process tree for an acyclic bridge-less WS WF-net.*

XY-path, where if $x \in P$ then X is P, otherwise X is T; if $y \in P$ then Y is P, otherwise Y is T. Let c be a circuit. A path h is said to be a handle of c if h and a part of c are disjoint. A path b is said to be a bridge between c and h if b shares only its start node or its end node with each of c and h. A WF-net N is said to be:

· Marked graph (MG for short) if $\forall p \in P - \{p_I, p_O\}$: $|\bullet p| = |p \bullet| = 1$, $|p_I \bullet| = 1$, and $|\bullet p_O| = 1$.

· Free choice (FC for short) if $\forall p, q \in P$: $p\bullet \cap q\bullet \neq \emptyset \Rightarrow |p\bullet| = |q\bullet| = 1$.

· Well-structured (WS for short) if $\overline{N}$ has neither TP-handles nor PT-handles.

· Bridge-less [8] if $\overline{N}$ has no bridge.

2.2 Process Tree

A process tree is a tree representation of a WF-net with block-structure [9, 10]. Bin Ahmadon *et al.* [11] proved that any acyclic bridge-less WS WF-net $(N, [p_I])$ can be represented as a process tree with three operators: *sequence* ($\rightarrow$), *exclusive-choice* ($\times$), and *parallel* ($\wedge$). The process tree of $(N, [p_I])$ is denoted by $\pi_{(N,[p_I])}$ and is recursively defined as follows (See Fig. 1). If N has a single transition t then $\pi_{(N,[p_I])}$ is a tree composed of a single node labeled as t (See Fig. 1 (a)). For the node labeled as t, we simply write t. Let $N_1, N_2, \cdots, N_n$ be subnets represented as $\pi_{(N_1,[p_I^{(1)}])}$, $\pi_{(N_2,[p_I^{(2)}])}$, $\cdots$, $\pi_{(N_n,[p_I^{(n)}])}$.

· If N is a sequential block-structure of $N_1, N_2, \cdots, N_n$ then $\pi_{(N,[p_I])}$ is an ordered rooted tree whose root is labeled as symbol '$\rightarrow$' and immediately precedes all the roots of $\pi_{(N_1,[p_I^{(1)}])}$, $\pi_{(N_2,[p_I^{(2)}])}$, $\cdots$, $\pi_{(N_n,[p_I^{(n)}])}$ (See Fig. 1 (b)).

· If N is an exclusive-choice block-structure of $N_1, N_2, \cdots, N_n$ then $\pi_{(N,[p_I])}$ is an ordered rooted tree whose root is labeled as symbol '$\times$' and immediately precedes all the roots of $\pi_{(N_1,[p_I^{(1)}])}$, $\pi_{(N_2,[p_I^{(2)}])}$, $\cdots$, $\pi_{(N_n,[p_I^{(n)}])}$ (See Fig. 1 (c)).

· If N is a parallel block-structure of $N_1, N_2, \cdots, N_n$ then, let π be an ordered rooted tree whose root is labeled as symbol '$\wedge$' and immediately precedes all the roots of $\pi_{(N_1,[p_I^{(1)}])}$, $\pi_{(N_2,[p_I^{(2)}])}$, $\cdots$, $\pi_{(N_n,[p_I^{(n)}])}$, $\pi_{(N,[p_I])}$ is an ordered rooted tree whose root is labeled as symbol '$\rightarrow$' and immediately precedes t_I, the root of π, and t_O (See Fig. 1 (d)).

2.3 DECLARE

DECLARE is a modeling language to express a process in the form of constraints. Users can use templates to describe constraints easily. The templates are classified into four groups: existence, relation, negative relation, and choice [5]. A relation template defines a dependency between two actions α and β. Typical relation templates are as follows:

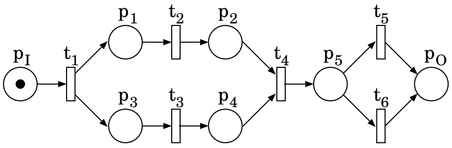
· The *responded existence*(α, β) template specifies that if α occurs, β also has to occur (either before or after α occurs). It is represented as the LTL formula $\diamond\alpha \Rightarrow \diamond\beta$.

· The *co-existence*(α, β) template specifies that if one of α and β occurs, the other one has to occur. It is represented as the LTL formula $\diamond\alpha \Leftrightarrow \diamond\beta$.

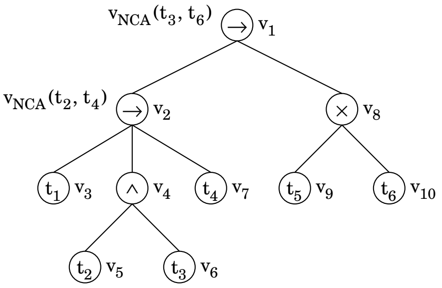
· The *precedence*(α, β) template specifies that β can occur only if α occurs before. It is represented as the LTL formula $(\neg\beta U \alpha) \vee \Box(\neg\beta)$.

· The *response*(α, β) template specifies that every time α occurs, β has to occur after α occurred. It is represented as the LTL formula $\Box(\alpha \Rightarrow \diamond\beta)$.

·The *succession*(α, β) template specifies that both the precedence and the response relations hold be-



(a) An acyclic bridge-less WS WF-net $(N_0, [p_I])$.



(b) The process tree $\pi_{(N_0,[p_I])}$.

Fig.2: *An example.*

tween α and β. It is represented as the LTL formula $((\neg\beta U\alpha) \vee \Box(\neg\beta)) \wedge \Box(\alpha \Rightarrow \diamond\beta)$.

3. ANALYSIS METHOD OF DECLARE RELATION CONSTRAINTS

We propose a method to analyze constraints of the DECLARE relation templates. Checking such constraints is important but intractable in general. Focusing our analysis on acyclic bridge-less WS WF-nets, which are convertible to process trees, we provide a necessary and sufficient condition on each of the DECLARE relation templates. We show that those conditions can be checked in polynomial time. Then we reveal the relationship among the templates.

3.1 Necessary and Sufficient Conditions

Let N be an acyclic bridge-less WS WF-net including two transitions t and u. In its process tree $\pi_{(N,[p_I])}$, nodes t and u have common ancestors. Let $v_{NCA}(t, u)$ denote the nearest common ancestor of t and u. As an example, let us consider the acyclic bridge-less WS WF-net N_0 shown in Fig. 2 (a). Figure 2 (b) shows the process tree $\pi_{(N_0,[p_I])}$. In $\pi_{(N_0,[p_I])}$, we found that $v_{NCA}(t_2, t_4)$ is v_2 and $v_{NCA}(t_3, t_6)$ is v_1.

3.1.1 Responded Existence

In terms of WF-nets, the *responded existence*(t, u) template specifies that in $(N, [p_I])$, if transition t fires then transition u also has to fire (either before or after t fires).

Theorem 1: For an acyclic bridge-less WS WF-net N including transitions t and u, the constraint *responded existence*(t, u) is satisfied in $(N, [p_I])$ if and only if there is no exclusive-choice ($\times$) operator node on the path between $v_{NCA}(t, u)$ and u in $\pi_{(N,[p_I])}$ (See Fig. 3). ■

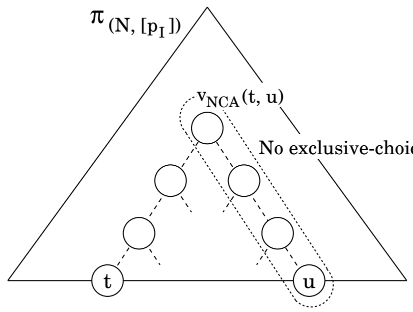


Fig.3: *Illustration of the process tree-based condition on the constraint responded existence(t, u) in an acyclic bridge-less WS WF-net* $(N, [p_I])$.

Definition 1: Let N be an acyclic WF-net. A subnet N' of N is called a MGWF-component if $\overline{N'}$ is a strongly-connected MG-component† of $\overline{N}$. ■

Property 1: Any acyclic WS WF-net N is covered by MGWF-components. ■

Proof: $\overline{N}$ is a FC Petri net from Property 4 of Ref. [12]. $(\overline{N}, [p_I])$ is live and safe from Property 5 of Ref. [12]. By Theorem 14 of Ref. [2], any live and safe FC Petri net is covered by strongly-connected MG-components. Since N is acyclic, each MG-component includes transition t^*. Removing t^* from the MG-component, we can obtain a MGWF-component. Thus N is covered by the MGWF-components that correspond one-to-one to the MG-components. **Q.E.D.**

Property 2: Let N be an acyclic bridge-less WS WF-net, and let MG be a MGWF-component of N. $\pi_{(MG,[p_I])}$ is a subtree of $\pi_{(N,[p_I])}$ which is obtained, for each exclusive-choice operator node, by leaving one subtree and removing the others. ■

Proof: MG is a subnet of N which has no exclusive-choice block structure. It is obtained from N by the following operation: For each exclusive-choice block structure, leave one subnet and remove the others. In terms of process tree, this operation means, for each exclusive-choice operator node, leaving one subtree and removing the others. **Q.E.D.**

For an acyclic WS WF-net N, $\mathcal{MG}(N)$ denotes the set of MGWF-components by which N is covered. For each $MG \in \mathcal{MG}(N)$, let $T(MG)$ denote the set of all the transitions in MG.

Proof of Theorem 1: Assume that there exists an exclusive-choice operator node on the path between $v_{NCA}(t, u)$ and u in $\pi_{(N,[p_I])}$. From Property 1, N is covered by $\mathcal{MG}(N)$. From Property 2, this means

$$\exists MG \in \mathcal{MG}(N) : (t \in \pi_{(MG,[p_I])} \wedge u \notin \pi_{(MG,[p_I])}) \quad (1)$$

On the other hand, since the constraint *responded existence*(t, u) is satisfied in $(N, [p_I])$, we have

†Let N $(=(P, T, A))$ be a Petri net, and $X \subseteq P \cup T$ such that $X \neq \emptyset$. $N \restriction X$ $(=(P \cap X, T \cap X, A \cap (X \times X)))$ denotes the subnet generated by X. $N \restriction X$ is called a MG-component of N if $N \restriction X$ is MG and $\forall t \in X \cap T$: $\bullet t \cup t \bullet \subseteq X$.

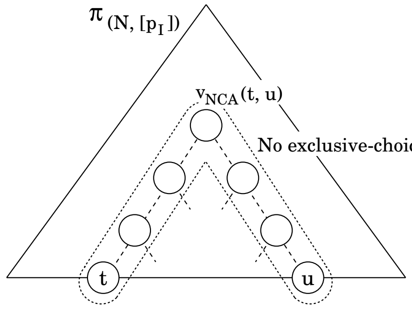


Fig.4: *Illustration of the process tree-based condition on the constraint co-existence(t, u) in an acyclic bridge-less WS WF-net* $(N, [p_I])$.

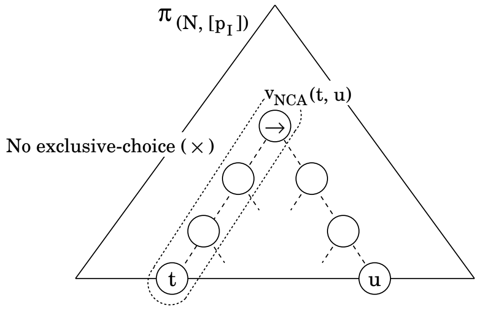


Fig.5: *Illustration of the process tree-based condition on the constraint precedence(t, u) in an acyclic bridge-less WS WF-net* $(N, [p_I])$.

In $(N, [p_I])$, if t fires then u also has to fire.
$\Leftrightarrow \forall MG{\in}\mathcal{MG}(N) : (t{\in}T(MG) \Rightarrow u{\in}T(MG))$
$\Leftrightarrow \forall MG{\in}\mathcal{MG}(N) : (t{\in}\pi_{(MG,[p_I])} \Rightarrow u{\in}\pi_{(MG,[p_I])})$
$\Leftrightarrow \neg\exists MG{\in}\mathcal{MG}(N) : (\alpha{\in}\pi_{(MG,[p_I])} \wedge \beta{\notin}\pi_{(MG,[p_I])})$
(De Morgan's laws)

This is inconsistent with Eq. (1). Thus there is no exclusive-choice operator node on the path between $v_{NCA}(t,u)$ and u in $\pi_{(N,[p_I])}$. **Q.E.D.**

3.1.2 Co-Existence

In terms of WF-nets, the *co-existence*$(t,\ u)$ template specifies that in $(N, [p_I])$, if one of transitions t and u fires then the other one has to fire.

Corollary 1: For an acyclic bridge-less WS WF-net N including transitions t and u, the constraint *co-existence*$(t,\ u)$ is satisfied in $(N, [p_I])$ if and only if there is no exclusive-choice $(\times)$ operator node on the path between $v_{NCA}(t,u)$ and t, and the path between $v_{NCA}(t,u)$ and u in $\pi_{(N,[p_I])}$ (See Fig. 4). ■
Proof: Similar to the proof of Theorem 1 since we have $\forall MG{\in}\mathcal{MG} : (t{\in}T(MG) \Leftrightarrow u{\in}T(MG))$.
Q.E.D.

3.1.3 Precedence

In terms of WF-nets, the *precedence*$(t,\ u)$ template specifies that in $(N, [p_I])$, transition u can fire only if transition t fires before.

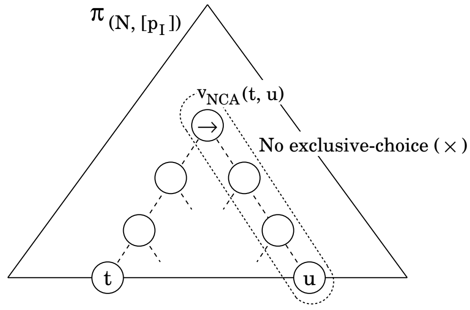


Fig.6: *Illustration of the process tree-based condition on the constraint response(t, u) in an acyclic bridge-less WS WF-net* $(N, [p_I])$.

Corollary 2: For an acyclic bridge-less WS WF-net N including transitions t and u, the constraint *precedence*$(t,\ u)$ is satisfied in $(N, [p_I])$ if and only if in $\pi_{(N,[p_I])}$,

(*i*) $v_{NCA}(t,u)$ is a sequence $(\rightarrow)$ operator node;
(*ii*) The position of t is on the left side of u; and
(*iii*) There is no exclusive-choice $(\times)$ operator node on the path between $v_{NCA}(t,u)$ and t. (See Fig. 5). ■

Proof: The proof of "if" part: From Conditions (i) and (ii), the sequence operator node has the subtree including t on the left side of the subtree including u. This means that in $(N, [p_I])$, a firing of t always precedes that of u. In addition, from Condition (iii) and Property 2, we have

$\forall MG{\in}\mathcal{MG}(N) : (u{\in}\pi_{(MG,[p_I])} \Rightarrow t{\in}\pi_{(MG,[p_I])})$
$\Leftrightarrow \forall MG{\in}\mathcal{MG}(N) : (u{\in}T(MG) \Rightarrow t{\in}T(MG))$
$\Leftrightarrow$ In $(N, [p_I])$, if u fires then t also has to fire.
$\Leftrightarrow$ In $(N, [p_I])$, u can fire only if t fires.

Thus, u can fire only if t fires before.

The proof of "only-if" part: We shall prove the contraposition. Condition (i): If $v_{NCA}(t,u)$ is an exclusive-choice $(\times)$ operator node, u cannot fire if t fires. If $v_{NCA}(t,u)$ is a parallel $(\wedge)$ operator node, a firing of t does not always precedes that of u in $(N, [p_I])$. Condition (ii): Assume that $v_{NCA}(t,u)$ is a sequence $(\rightarrow)$ operator node. If the position of t is on the right side of u, a firing of t does not precede that of u in $(N, [p_I])$. Condition (iii): If there exists an exclusive-choice $(\times)$ operator node on the path between $v_{NCA}(t,u)$ and t, if u fires then t does not always fire in $(N, [p_I])$. This means that u does not always fire only if t fires. **Q.E.D.**

3.1.4 Response

In terms of WF-nets, the *response*$(t,\ u)$ template specifies that in $(N, [p_I])$, every time transition t fires, transition u has to fire after t.

Corollary 3: For an acyclic bridge-less WS WF-net N including transitions t and u, the constraint *response*$(t,\ u)$ is satisfied in $(N, [p_I])$ if and only if in $\pi_{(N,[p_I])}$,

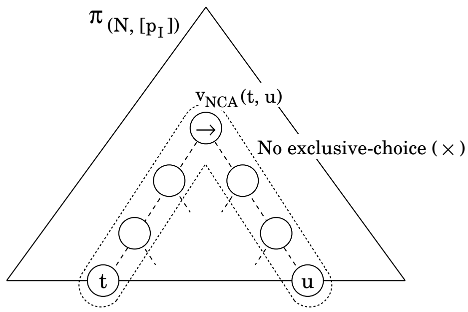


Fig. 7: *Illustration of the process tree-based condition on the constraint succession(t, u) in an acylic bridge-less WS WF-net* $(N, [p_I])$.

(i) $v_{NCA}(t, u)$ is a sequence ($\rightarrow$) operator node;
(ii) The position of t is on the left side of u; and
(iii) There is no exclusive-choice ($\times$) operator node on the path between $v_{NCA}(t, u)$ and u. (See Fig. 6) ■

Proof: Similar to the proof of Corollary 2. **Q.E.D.**

3.1.5 Succession

In terms of WF-nets, the *succession*(t, u) template specifies that in $(N, [p_I])$, both the *precedence*(t, u) and the *response*(t, u) hold between transitions t and u.

Corollary 4: For an acyclic bridge-less WS WF-net N including transitions t and u, the constraint *succession*(t, u) is satisified in $(N, [p_I])$ if and only if in $\pi_{(N,[p_I])}$,

(i) $v_{NCA}(t, u)$ is a sequence ($\rightarrow$) operator node;
(ii) The position of t is on the left side of u;
(iii) There is no exclusive-choice ($\times$) operator node on the path between $v_{NCA}(t, u)$ and t; and
(iv) There is no exclusive-choice ($\times$) operator node on the path between $v_{NCA}(t, u)$ and u. (See Fig. 7). ■

Proof: Follows immediately from Corollaries 2 and 3. **Q.E.D.**

3.2 Computation Complexity and Relationship

The proposed necessary and sufficient conditions enable us to check a constraint of the DECLARE relation templates in polynomial time.

Theorem 2: The following problem can be solved in polynomial time: Given an acyclic bridge-less WS WF-net N including two transitions t and u, to decide if constraints *responded existence*, *co-existence*, *precedence*, *response*, and *succession*(t, u) are respectively satisfied in $(N, [p_I])$. ■

Proof: We have only to construct $\pi_{(N,[p_I])}$ and to traverse it. The tree construction with ≪Process Tree Conversion Algorithm≫ of Ref. [13] and the tree traversal with Bread-First Search take $O(|P| + |T| + |A|)$. **Q.E.D.**

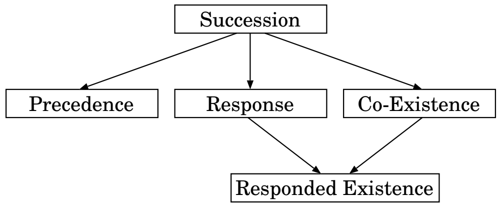


Fig. 8: *Illustration of the relationship among the DECLARE relation templates.*

On the other hand, it is intractable to check a constraint of the DECLARE relation templates in acyclic FC WF-nets.

Corollary 5: The following problem is NP-complete: Given an acyclic FC WF-net N including two transitions t and u, to decide if constraints *responded existence*, *co-existence*, *precedence*, *response*, and *succession*(t, u) are respectively satisfied in $(N, [p_I])$. ■

Proof: This problem can be regarded as a kind of liveness. In a similar way to the proof of Theorem 1 of Ref. [14], we can prove the NP-completeness by reducing the 3-conjunctive normal form boolean satisfiability problem to the problem. **Q.E.D.**

This intractability enhances the value of our necessary and sufficient conditions checkable in polynomial time, because acyclic WS WF-nets form the largest class in the well-known subclasses of acyclic FC WF-nets.

In addition, we also reveal the following relationship among the DECLARE relation templates.

Property 3: Let N be an acyclic bridge-less WS WF-net N including transitions t and u.

(i) The *succession*(t, u) $\Rightarrow$ the *precedence*(t, u)
(ii) The *succession*(t, u) $\Rightarrow$ the *response*(t, u) $\Rightarrow$ the *responded existence*(t, u)
(iii) The *succession*(t, u) $\Rightarrow$ the *co-existence*(t, u) $\Rightarrow$ the *responded existence*(t, u) ■

Proof: Follows from Theorem 1 and Corollaries 1–4. **Q.E.D.**

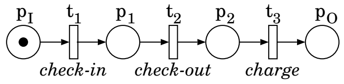
This relationship is illustrated in Fig. 8. An arrow from p to q means "p implies q."

4. APPLICATION EXAMPLE AND PERFORMANCE EVALUATION

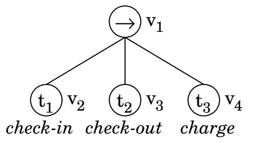
4.1 Application Example

We illustrate the proposed analysis method with an application example. In this example, we analyzed two workflow definitions for lending rooms in a hotel. One is a WF-net N_1 of Fig. 9 (a), the other is a WF-net N_2 of Fig. 10 (a). These WF-nets include three tasks: "*check-in*", "*check-out*" and "*charge*". These tasks should satisfy the following constraints:

(i) Every time task "*check-in*" is executed, task "*check-out*" has to be executed after "*check-in*" is executed.



(a) The WF-net model $(N_1, [p_I])$.



(b) The process tree $\pi_{(N_1,[p_I])}$.

Fig.9: *Example 1: A classical hotel process.*

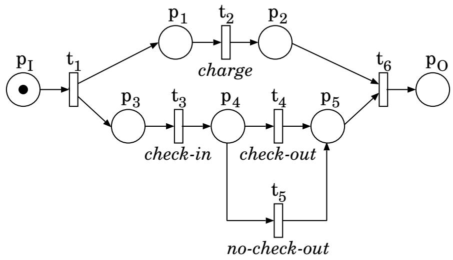
(*ii*) Every time task "*check-out*" is executed, task "*check-in*" has to be executed before "*check-out*" is executed.

(*iii*) If task "*check-out*" is executed, task "*charge*" also must be executed (either before or after "*check-out*" is executed).

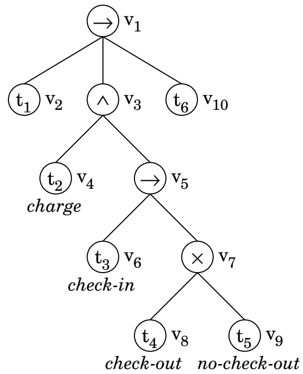
Constraints (i)–(iii) are respectively represented as the *precedence*("*check-in*", "*check-out*"), the *response*("*check-in*", "*check-out*"), and the *responded existence*("*check-out*", "*charge*").

Let us first analyze N_1. Figure 9 (b) shows $\pi_{(N_1,[p_I])}$. N_1 satisfies Rule (i), i.e. the *precedence*(t_1 (labeled as "*check-in*"), t_2 (labeled as "*check-out*")), because in $\pi_{(N_1,[p_I])}$ $v_{NCA}(t_1, t_2)$ (=v_1) is a sequence operator node, the position of t_1 is on the left side of t_2, and there is no exclusive-choice operator node in the tree. N_1 also satisfies Rule (ii), i.e. the *response*(t_1, t_2), for the same reason as Rule (i). Property 3 implies that N_1 satisfies the succession (t_1, t_2). In addition, N_1 satisfies Rule (iii), i.e. the *responded existence*(t_2 (labeled as "*check-out*"), t_3 (labeled as "*charge*")), because there is no exclusive-choice operator node in $\pi_{(N_1,[p_I])}$.

Next, let us analyze N_2. In this net, task "*charge*" is not restricted after task "*check-out*". Task "*check-out*" is not necessary. Figure 10 (b) shows $\pi_{(N_2,[p_I])}$. N_2 satisfies Rule (i), i.e. the *precedence*(t_3 (labeled as "*check-in*"), t_4 (labeled as "*check-out*")), because in $\pi_{(N_2,[p_I])}$ $v_{NCA}(t_3, t_4)$ (=v_5) is a sequence operator node, the position of t_3 is on the left side of t_4, and there is no exclusive-choice operator node between v_5 and t_3. On the other hand, N_2 does not satisfy Rule (ii), i.e. the *response*(t_3, t_4), because there exists an exclusive-choice operator node v_7 between v_5 and t_4. From the condition on the response template, we get a clue to improve N_2 to satisfy the *response*(t_3, t_4). We have only to clear the exclusive-choice operator node v_7 from the path between v_5 and t_4. To do so, one way is to simply remove task "*no-check-out*". Another way is to put tasks "*check-in*" and "*check-out*" together within the exclusive choice block-structure. N_2 satisfies Rule (iii), i.e. the *responded existence*(t_4 (labeled as "*check-out*"), t_2 (labeled as "*charge*")), because there is no exclusive-choice operator node between $v_{NCA}(t_4, t_2)$ (=v_3) and t_2 in $\pi_{(N_2,[p_I])}$.



(a) The WF-net model $(N_2, [p_I])$.



(b) The process tree $\pi_{(N_2,[p_I])}$.

Fig.10: *Example 2: An ultra-modern hotel process.*

4.2 Performance Evaluation

We conducted an experiment to evaluate the proposed method quantitatively. Van der Aalst stated that in most cases, the number of tasks is less than 100 [1]. In this experiment, we constructed and used a WF-net N_3 with 100 transitions as a worst case. For the modeling, we used the tool called *Workflow Petri Net Designer* (WoPed) [15] which has been developed at the Cooperative State University Karlsruhe. Figure 11 is a snapshot of WoPeD which shows $(N_3, [p_I])$. Next we converted N_3 to the process tree $\pi(N_3, [p_I])$ with our tool named *Process Tree Analysis Tool* (ProTAT) [11]. Figure 12 is a snapshot of ProTAT which shows $\pi(N_3, [p_I])$. We extended the ProTAT to include the proposed method. Then we computed the following DECLARE properties with the extended ProTAT.

(*i*) The responded existence (t_8, t_{86})
(*ii*) The co-existence (t_8, t_{86})
(*iii*) The precedence (t_8, t_{86})
(*iv*) The response (t_8, t_{86})
(*v*) The succession (t_8, t_{86})
(*vi*) The responded existence (t_8, t_{19})
(*vii*) The co-existence (t_8, t_{19})
(*viii*) The precedence (t_8, t_{19})
(*ix*) The response (t_8, t_{19})
(*x*) The succession (t_8, t_{19})

We show the computation results in Table 1. The evaluation was conducted on Windows 10 computer with an Intel Core i7 2.4GHz (Quad-Core) processor and 8 GB RAM memory. Every computation time is

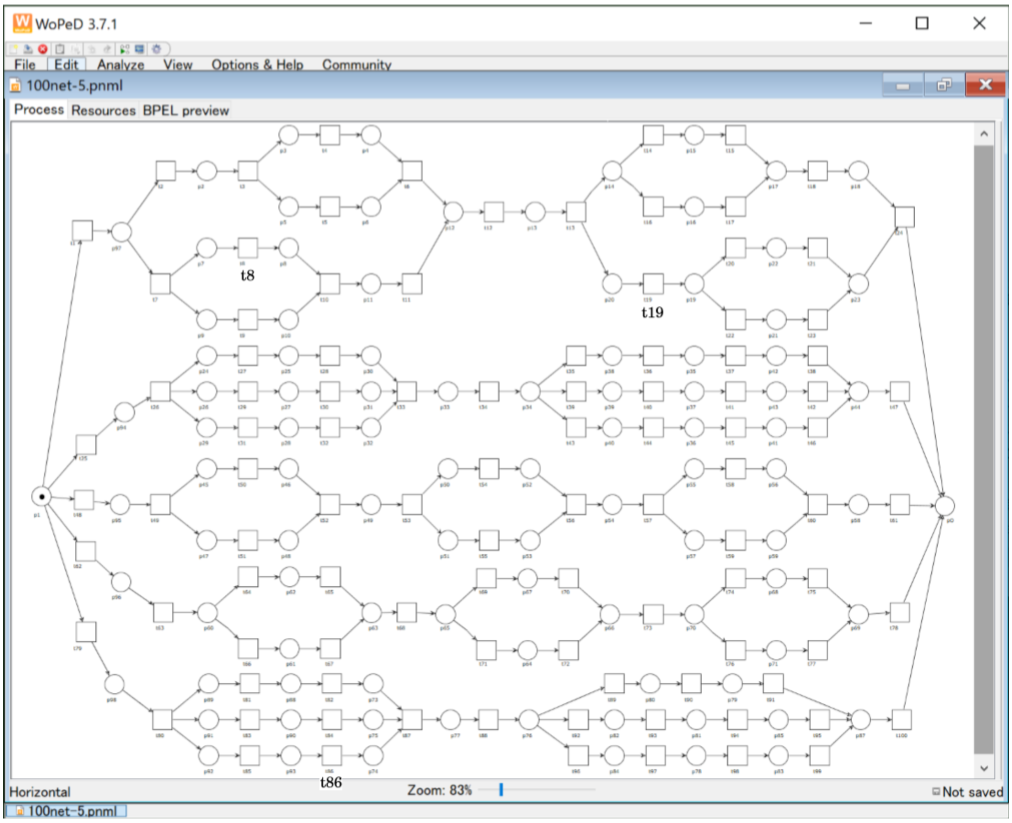


Fig.11: *A snapshot of WoPeD. It shows a WF-net* $(N_3, [p_I])$ *with 100 transitions.*

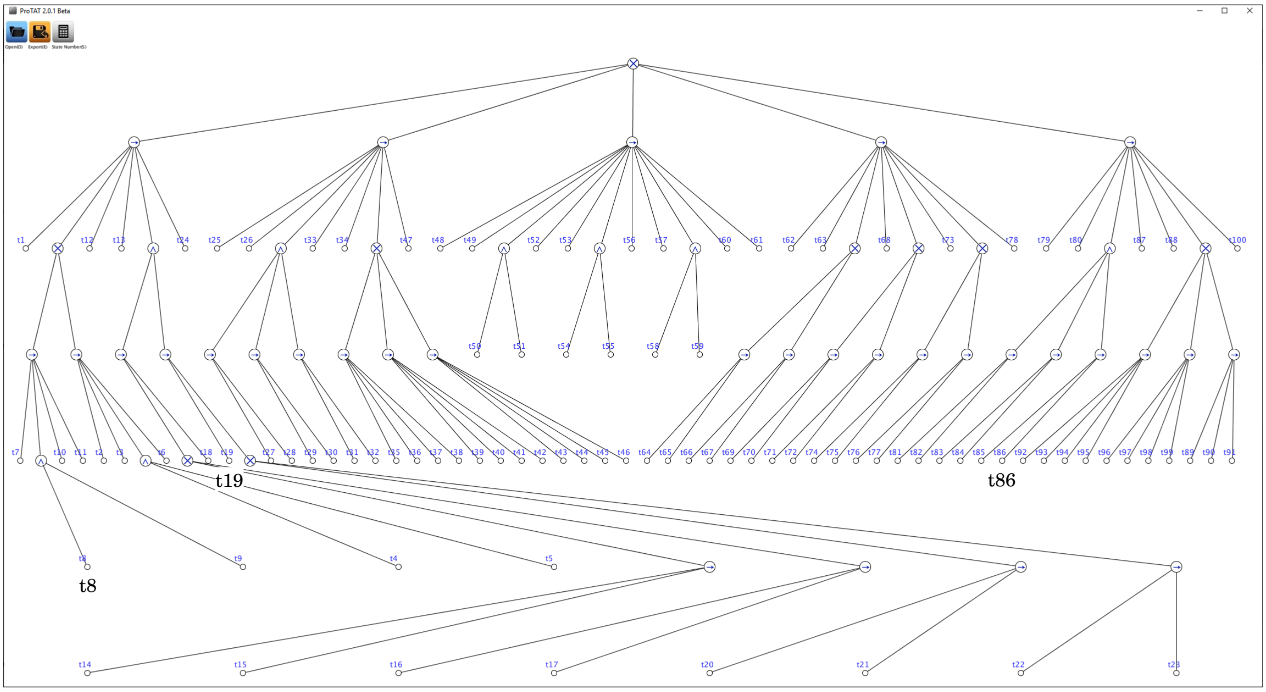


Fig.12: *A snapshot of ProTAT. It shows the process tree* $\pi(N_3, [p_I])$.

less than one millisecond. This means that with the proposed method, it is feasible to perform the verification for practical-sized business processes. From these results, we conclude the proposed method useful.

5. CONCLUSION

In this paper, we proposed a method to analyze constraints of the DECLARE relation templates. Checking such constraints is important but known to

Table 1: *Computation results of the extended ProTAT with the proposed method.*

Property	Result	Computation time
Responded existence (t_8, t_{86})	false	0.205ms
Co-existence (t_8, t_{86})	false	0.137ms
Precedence (t_8, t_{86})	false	0.158ms
Response (t_8, t_{86})	false	0.178ms
Succession (t_8, t_{86})	false	0.192ms
Responded existence (t_8, t_{19})	true	0.193ms
Co-existence (t_8, t_{19})	false	0.131ms
Precedence (t_8, t_{19})	false	0.144ms
Response (t_8, t_{19})	true	0.168ms
Succession (t_8, t_{19})	false	0.178ms

be intractable in general. Focusing our analysis on acyclic bridge-less WS WF-nets, we provided necessary and sufficient conditions on each template on the basis of a process tree. Those conditions enable us to not only check a given constraint in polynomial time, but also find a clue for improving any net violating the constraints. In addition, we revealed the relationship among the templates. We also illustrated the proposed analysis method with an application example and showed the usefulness of the method.

In our future work, we will propose a systematic way to apply our method to business process management.

ACKNOWLEDGMENT

This work was partially supported by Interface Corporation.

References

[1] W.M.P. van der Aalst, "The application of Petri nets to workflow management," *J. Circuits, Systems, Computers*, vol.8, no.1, pp.21–65, 1998.

[2] T. Murata, "Petri nets: properties, analysis and applications," *Proc. of IEEE*, vol.77, no.4, pp.541–580, 1989.

[3] M. Pesic, H. Schonenberg, W.M.P. van der Aalst, "DECLARE: Full support for loosely-structured processes," *Proc. of IEEE EDOC 2007*, pp.287–300, 2007.

[4] DECLARE, `http://www.win.tue.nl/declare/`

[5] F.M. Maggi, A.J. Mooij, W.M.P. van der Aalst, "User-guided discovery of declarative process models," *Proc. of IEEE CIDM 2011*, pp.192–199, 2011.

[6] M.S.B. Ab Malek, M.A. Bin Ahmadon, S. Yamaguchi, "Computational complexity and polynomial time procedure of response property problem in workflow nets," *IEICE Trans. Fundamentals*, vol.E101-D, no.6, pp.1503–1510, 2018.

[7] S. Yamaguchi, M.A. Bin Ahmadon, Q.W. Ge, "Introduction of Petri nets: Its applications and security challenges," in *Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security*, Hershey, PA, USA, chapter 7, pp.145–179, 2016.

[8] S. Yamaguchi, M.A. Bin Ahmadon, "Properties and decision procedure for bridge-less workflow nets," *IEICE Trans. Fundamentals*, vol.E99-A, no.2, pp.509–512, 2016.

[9] D. Nikovski, A. Baba, "Workflow trees for representation and mining of implicitly concurrent business processes," *Mitsubishi Electric Research Laboratories Technical Report*, TR2007-072, 2007.

[10] W.M.P. van der Aalst, J.C.A.M. Buijs, and B.F. van Dongen, "Towards improving the representational bias of process mining," *Lecture Notes in Business Information Processing*, vol.116, pp.39–54, Springer-Verlag, Berlin, 2012.

[11] M.A. Bin Ahmadon, S. Yamaguchi, "State number calculation problem of workflow nets," *IEICE Trans. Information and Systems*, vol.E98-D, no.6, pp.1128–1136, 2015.

[12] S. Yamaguchi, "Polynomial time verification of reachability in sound extended free-choice workflow nets," *IEICE Trans. Fundamentals*, vol.E97-A, no.2, pp.468–475, 2014.

[13] M.A. Bin Ahmadon, S. Yamaguchi, "Convertibility and conversion algorithm of well-structured workflow net to process tree," *Proc. of CANDAR 2013*, pp.122–127, 2013.

[14] S. Yamaguchi, "Analysis of option to complete, proper completion and no dead tasks for acyclic free choice workflow nets," *IEICE Trans. Fundamentals*, vol.E102A, no.2, pp.336–342, 2019.

[15] WoPeD, `https://woped.dhbw-karlsruhe.de/`

Shingo Yamaguchi received the B.E., M.E. and D.E. degrees from Yamaguchi University, Japan, in 1992, 1994 and 2002, respectively. He was a Visiting Scholar in the Department of Computer Science at University of Illinois at Chicago, United States, in 2007. He is currently a Professor in the Graduate School of Sciences and Technology for Innovation, Yamaguchi University, Japan. His research interests are in the area of net theory and its applications. He is a Senior Member of IEEE and an elected member of Board of Governors of IEEE Consumer Electronics Society.

Mohd Anuaruddin Bin Ahmadon received his B.E., M.E. and D.E. degrees from Yamaguchi University, Japan in 2014, 2015 and 2017, respectively. He is currently an Assistant Professor in the Graduate School of Sciences and Technology for Innovation, Yamaguchi University, Japan. His research interests include Petri net and its application to system analysis. He is a member of IEEE.