2013 First International Symposium on Computing and Networking

Convertibility and Conversion Algorithm of Well-Structured Workflow Net to Process Tree

†Mohd Anuaruddin Bin Ahmadon, ‡Shingo Yamaguchi
†Faculty of Engineering, Yamaguchi University
‡Graduate School of Science and Engineering, Yamaguchi University
2-16-1 Tokiwadai, Ube, 755-8611, Japan
Telephone.: +81-836-85-9510, Fax.:+81-836-85-9501
Email: {s903ff, shingo}@yamaguchi-u.ac.jp

***Abstract*—A workflow net is a Petri net for modeling and analyzing a workflow. A process tree has been proposed as a representational bias of a workflow net and enabled us to grasp the structure of actions routing such as sequence, choice and parallel in the workflow net. The advantage of process trees enables us to grasp the structure of action routing such as sequence, choice, and parallel in a given workflow net. Unfortunately, workflow nets are not always convertible to process trees. In this paper, we discuss the convertibility and conversion of workflow nets to process trees. We first proposed a necessary and sufficient condition on convertibility, i.e. a workflow net is convertible to a process tree iff it is acyclic, bridge-less, and well-structured. Next, based on the condition we constructed an algorithm to convert the workflow net to a process tree. Then we proposed an application of the conversion algorithm, which allows us to calculate the state number of the workflow net.**

I. Introduction

Petri nets [1] are a mathematical and graphical modeling tool applicable to many systems. Workflow net (WF-net for short) is a subclass of Petri nets for modeling and analyzing a workflow. WfMC, an international standardization orgatnization on workflows [2] had distinguished four types of routing construct known as sequence, parallel, choice (selection) and iteration. It is important to understand the routings in a workflow. WF-net can represent those routing construction. One routing of a series of actions can be easily understood but it is not easy to understand the overall relations between routings only by looking at WF-net alone. This leads to representational bias of WF-net.

There are various representational biases for WF-net such as footprint [3] and process tree [4]. These representational biases have been originally developed for process mining. Footprint is a matrix representation of WF-net that represents order relations between transitions in a WF-net. Yamaguchi et al. [3] applied footprint in superclass extraction from existing WF-nets. Process tree is a tree representation that represents relation between routings for mining sound WF-nets. Process tree is very useful to understand the overall routings of WF-nets. Susaki et al. [5] applied process tree for calculating the state number of WF-nets. Van der Aalst et al. [4] implemented a genetic algorithm to discover process tree in process mining techniques. However, not all WF-nets can be converted to process tree, e.g., non-sound WF-nets cannot be converted to process tree. Therefore it is necessary to decide a given WF-net can be converted to a process tree.

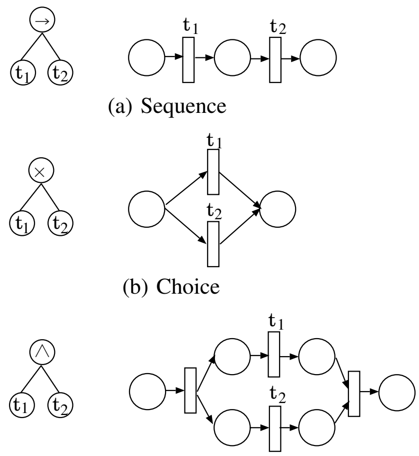
In this paper, we discuss the convertibility and conversion of WF-net to process trees. We first propose a necessary and sufficient condition on convertibility. Next, we construct a conversion algorithm based on the condition. Then we propose an application of the conversion algorithm, which allows us to apply state number calculation for WF-nets as proposed by Susaki et al. [5].

II. Preliminary

a) Petri Nets and workflow nets: Petri net is a three tuple $N{=}(P, T, A)$, where P, T, and A ($\subseteq(P{\times}T)\cup(T{\times}P)$) are finite sets of places, transitions, and arcs, respectively. Let x be a node of N. $\overset{N}{\bullet}x$ and $x\overset{N}{\bullet}$ respectively denote $\{y|(y,x){\in}A\}$ and $\{y|(x,y){\in}A\}$. A marking (or a state) is a mapping $M{:}\,P{\rightarrow}\mathbb{N}$. We represent M as a bag over P: $M{=}[p^{M(p)}|p{\in}P, M(p){>}0]$. A transition t is said to be firable in M if $M{\geq}\overset{N}{\bullet}t$. Firing t in M results in a new marking M' ($=M{\cup}t\overset{N}{\bullet} - \overset{N}{\bullet}t$). This is denoted by $M[N,t\rangle M'$. A marking M_n is said to be reachable from a marking M_0 if there exists a transition sequence $t_1t_2\cdots t_n$ such that $M_0[N,t_1\rangle M_1[N,t_2\rangle M_2\cdots[N,t_n\rangle M_n$. The set of all markings reachable from M_0 in (N,M_0) is denoted by $R(N,M_0)$. The tree representation of the markings in $R(N,M_0)$ is called the reachability tree.

N 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; and (iii) there is no dead transition in N. There is a particular subclass of WF-nets: *well-structured* (WS for short). A structural characterization of good workflows is that two paths initiated by a transition (a place) should not be joined by a place (a transition). WS is derived from this structural characterization. To give the formal definition of WS, we introduce some notations. We make N strongly connected by connecting p_O to p_I via an additional transition t^*. The resulting Petri net is called the *short-circuited net* of N, and is denoted by $\overline{N}$ ($=(P,\, T{\cup}\{t^*\}, A{\cup}\{(p_O,t^*),(t^*,p_I)\})$). Let c be a circuit in $\overline{N}$. A path $\rho = x_1x_2\cdots x_n$ ($n{\geq}2$) is called a *handle* [6] of c if ρ shares exactly two nodes, x_1 and x_n, with c. A path ρ is called a bridge between c and its handle h if each of c and h shares exactly one node, x_1 or x_n, with b. A handle (a bridge) from a place to another place is called a PP-handle (a PP-bridge). A handle (a bridge) from a place to a transition is called a PT-handle (a PT-bridge). A handle (a bridge) from a transition to a place is called a TP-handle (a TP-bridge). A handle (a bridge) from a transition to another

978-1-4799-2795-1/13 $31.00 © 2013 IEEE
DOI 10.1109/CANDAR.2013.24



(a) Sequence

(b) Choice

(c) Parallel

Fig. 1. Translation of process tree operators to Petri net constructs

transition is called a TT-handle (a TT-bridge). A WF-net N is said to be WS if there are neither TP-handles nor PT-handles of any circuit in $\overline{N}$. We can decide in polynomial time whether a given WF-net is WS by applying a modified version of the max-flow min-cut technique [2].

b) Process Tree: A process tree is a tree representation of a process [4]. Each leaf node and each internal node respectively represent an action and an operator in the process. In this paper, we use three operators: sequence ($\rightarrow$), exclusive choice ($\times$) and parallel ($\wedge$).

Definition 1: The set Π of process trees π is as follows:

(i) If ℓ is an action label, then $\ell \in \Pi$.
(ii) If $\oplus$ is an operator and $\ell_1, \ell_2, \cdots, \ell_n$ are action labels, then $\oplus(\ell_1, \ell_2, \cdots, \ell_n) \in \Pi$.
(iii) If $\oplus$ is an operator and $\pi_1, \pi_2, \cdots, \pi_n \in \Pi$, then $\oplus(\pi_1, \pi_2, \cdots, \pi_n) \in \Pi$. ■

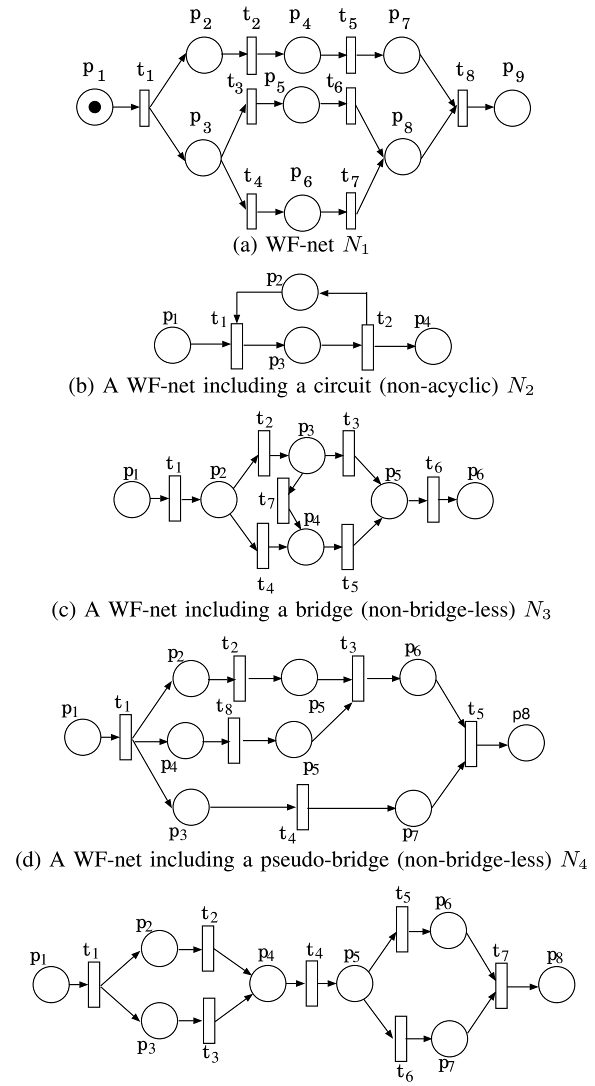
Each operator can be translated to a part of a Petri net (see Fig. 1). Let us consider a WF-net N_1, which is shown in Fig. 2 (a). The process tree of N_1 is shown in Fig. 3. It is also represented as $\wedge(\rightarrow(t_2, t_5), \times(\rightarrow(t_3, t_6), \rightarrow(t_4, t_7)))$.

III. Convertibility of Workflow Net to Process Tree

Van der Aalst [4] has shown that process trees can represent some WF-nets. Unfortunately, all WF-nets are not always convertible to proccess trees. For example, non-sound WF-nets are not convertible because process trees cannot represent non-sound WF-nets. It is necessary to decide whether a given WF-net is convertible to a process tree. We call this problem as convertibility problem. In this section, we first give the formal definition of convertibility problem. Then we give a necessary and sufficient condition on the problem.

A. Convertibility problem

We formalize convertibility problem as follows:



(a) WF-net N_1

(b) A WF-net including a circuit (non-acyclic) N_2

(c) A WF-net including a bridge (non-bridge-less) N_3

(d) A WF-net including a pseudo-bridge (non-bridge-less) N_4

(e) A WF-net with a TP-handle and a PT-handle N_5

Fig. 2. Example of WF-net instances

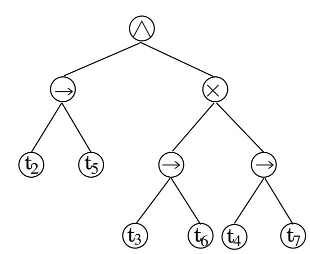
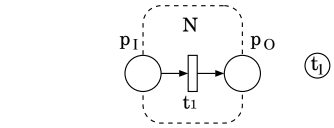


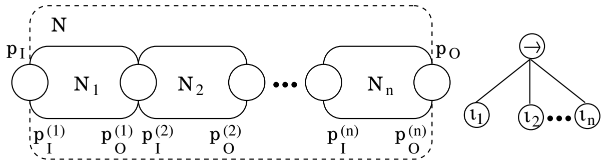
Fig. 3. Process tree of N_1

Definition 2 (Convertibility problem): Instance: WF-net N, Question: Is N convertible to a process tree? ■

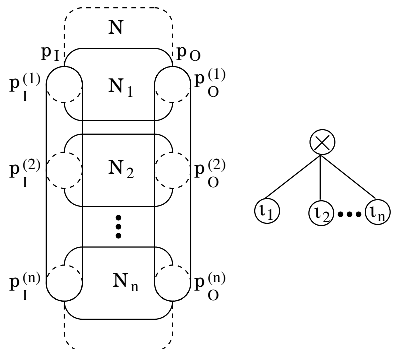
We consider five instances of convertibility problem for example. The first instance is a WF-net N_1 shown in Fig. 2 (a). This WF-net can be represented as process tree as shown



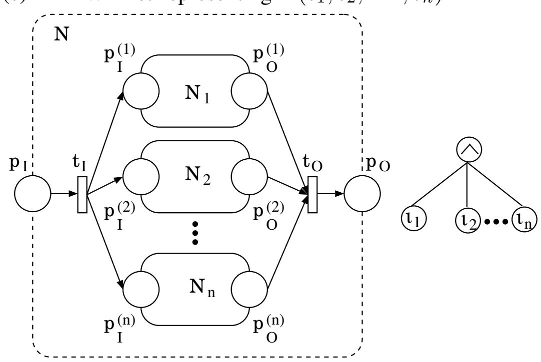
(a) PTB WF-net representing an action label ℓ



(b) PTB WF-net representing $\rightarrow(\ell_1, \ell_2, \cdots, \ell_n)$



(c) PTB WF-net representing $\times(\ell_1, \ell_2, \cdots, \ell_n)$



(d) PTB WF-net representing $\wedge(\ell_1, \ell_2, \cdots, \ell_n)$

Fig. 4. Illustration of PTB WF-net and its equivalent process tree

in Fig. 3. By looking at Fig. 2 (a) we found that N_1 is an acyclic WS WF-net and has no bridge. The second instance is a WF-net N_2 shown in Fig. 2 (b) has a circuit $t_1p_3t_2p_2t_1$. In this paper, we use no operator representing such circuit. Therefore, we assume that N_2 is not convertible to a process tree. The third instance is a WF-net N_3 shown in Fig. 2 (c) has a bridge $p_3t_7p_4$. Originally without bridge (the path of $p_3t_7p_4$), paths $p_2t_2p_3t_3p_5$ and $p_2t_4p_4t_5p_5$ are a choice but since the bridge $p_3t_7p_4$ exists, $t_2p_3t_7p_4t_5$ forms a new sequence relation connecting the path. So action t_2 and t_5 has two relations, a choice and a sequence. It is not convertible because process tree operator can only represent one routing relation between actions. The forth instance is a WF-net N_4 shown in Fig. 2 (d). N_4 has 3 paths $t_1p_2t_2p_5t_3p_6t_5$, $t_1p_4t_8p_5t_3$ and $t_1p_3t_4p_7t_5$ which are respectively regarded as a circuit and its handle, there is a path $t_1p_4t_8p_5t_3$ between them. It is similar to a bridge but is not exactly a bridge. We call it "pseudo-bridge". It is not convertible because if without the pseudo-bridge, t_1 has a parallel relation with t_3, but since the pseudo-bridge $t_1p_4t_8p_5t_3$ exists, a new relation exists between t_1 and t_5. Since the action t_1 has more than one relation it cannot be represented with process tree operator. A WF-net N is said to be bridge-less if the short-circuited net of N includes neither bridges nor pseudo-bridges. The fifth instance is a WF-net N_5 shown in Fig. 2 (e). N_5 has a TP-handle $t_1p_2t_2p_4$ and a PT-handle $p_5t_5p_6t_7$. Since N_5 is not WS and there are no operator to represent TP-handle and PT-handle, it is not convertible. By generalizing the analysis result, we deduced that acyclic, bridge-less and WS structure plays a core role in the convertibility problem.

B. Necessary and Sufficient Condition

We propose a necessary and sufficient condition on the convertibility problem. For this we (i) define a subclass of WF-nets called as Process Tree Based (PTB for short) WF-net which can be represented as a process tree and (ii) show the PTB WF-net is acyclic, bridge-less and WS and (iii) show that a WF-net is PTB , i.e. convertible to a process tree iff it is acyclic, bridge-less and WS.

Definition 3 (PTB WF-net): For any process tree π,

(i) If π is an action label, a WF-net N which consists of a transition representing the action label and its input and output places is PTB (See Fig. 4(a)).

(ii) If π is $\oplus(\ell_1, \ell_2, \cdots, \ell_n)$ then let $N_1, N_2, \cdots, N_n$ be respectively PTB WF-nets representing action labels $\ell_1, \ell_2, \cdots, \ell_n$.

   a) If $\oplus$ is sequence ($\rightarrow$) then a WF-net which is constructed by concatenating $N_1, N_2, \cdots, N_n$ to link the sink place of N_i with the source place of $N_{i+1} (1 \leq i < n)$ is PTB. The WF-net is illustrated in Fig. 4 (b).

   b) If $\oplus$ is choice ($\times$) then a WF-net which is constructed by bundling $N_1, N_2, \cdots, N_n$ to unite all their source places and sink places is PTB. The WF-net is illustrated in Fig. 4 (c).

   c) If $\oplus$ is parallel ($\wedge$) then a WF-net which is constructed by joining respectively all source places and sink places of $N_1, N_2, \cdots, N_n$ is PTB. The WF-net is illustrated in Fig. 4 (d).

(iii) If π is $\oplus(\pi_1, \pi_2, \cdots, \pi_n)$ then let $N_1, N_2, \cdots, N_n$ be respectively PTB WF-nets representing sub-process trees $\pi_1, \pi_2, \cdots, \pi_n$.

   a) If $\oplus$ is sequence then a WF-net which is constructed by concatenating $N_1, N_2, \cdots, N_n$ to link the sink place of N_i with the source place of $N_{i+1} (1 \leq i < n)$ is PTB.

   b) If $\oplus$ is choice then a WF-net constructed by bundling PTB WF-nets $N_1, N_2, \cdots, N_n$ to unite all their source places and sink places is PTB.

   c) If $\oplus$ is parallel then a WF-net constructed by joining respectively all source places and sink places of PTB WF-net $N_1, N_2, \cdots, N_n$ is PTB. ■

N_1 shown in Fig. 2 (a) is PTB. Let us construct N_1 from the process tree shown in Fig. 3. For $\rightarrow(t_2, t_5)$ we construct a WF-net $p_2t_2p_4t_5p_7$ based on Item (ii)-a) of Def. 3. For $\times(\rightarrow(t_3, t_6), \rightarrow(t_4, t_7))$ we constructed a WF-net by bundling paths $p_3t_3p_5t_6p_8$ and $p_3t_4p_6t_7p_8$ based on Item (iii)-b). We obtain N_1 by bundling those WF-nets.

Lemma 1: A WF-net is PTB iff N is acyclic, bridge-less and WS. ■

Proof: The proof of "if" part: We make use of van Hee et al. [7]'s ST-net[1]. We show the following: (i) An acylic bridge-less WS WF-net N is a ST-net. (ii) An acyclic, bridge-less ST-net is PTB.

We first show that an acylic bridge-less WS WF-net N is a ST-net. Intuitively, ST-nets are constructed from SMs and MGs by means of refinement [2]. The dual nets [6] of WF-nets are called tWF-nets. From the definition of WS, there are neither TP-handles nor PT-handles of any circuit in $\overline{N}$. This implies that $\overline{N}$ consists of a circuit c, PP-handles of c, and TT-handles of c. Any PP-handle includes both terminal nodes of a TT-handle, or includes none. We can look for an SM WF-net N as a subnet of N, which consists of PP-handles not including terminal nodes of any TT-handle. This implies $N = \mathcal{N} \otimes_p \mathcal{M}$ for some place p of a WF-net $\mathcal{N}$. Similarly, any TT-handle includes both terminal nodes of a PP-handle, or includes none. We can look for an acyclic MG tWF-net $\mathcal{M}$ as a subnet of N, which consists of TT-handles not including terminal nodes of any PP-handle. This implies $N = \mathcal{N} \otimes_t \mathcal{M}$ for some transition t of a WF-net $\mathcal{N}$. Repeating these refinements, we can show that N is a ST-net.

Next we show that an acylic bridge-less ST-net N is PTB. Any acyclic bridge-less SM or MG WF-net is obviously PTB. Let $\mathcal{N}$ be a PTB WF-net, t a transition in $\mathcal{N}$ and $\mathcal{M}$ a acyclic bridge-less MG tWF-net. Let $\mathcal{M}'$ be a WF-net obtained by extending a place to each source transition and sink transition in $\mathcal{M}$. Since $\mathcal{N}$ and $\mathcal{M}'$ are PTB they have process trees $\pi_{\mathcal{N}}$ and $\pi_{\mathcal{M}'}$. $\mathcal{N} \otimes_t \mathcal{M}$ has a process tree by replacing transition t in $\pi_{\mathcal{N}}$ with $\pi_{\mathcal{N}}$. Therefore $\mathcal{N} \otimes_t \mathcal{M}$ is PTB. In the similar way, $\mathcal{N} \otimes_p \mathcal{M}$ is PTB.

The proof of "only if" part: If a WF-net is PTB, then it is acyclic bridge-less WS. From Item (ii)-a) of Def. 3 $\rightarrow(\ell_1, \ell_2, \cdots, \ell_n)$ constructs an WF-net which is a path. It is acylic, bridge-less and WS. From Item (ii)-b) of Def. 3 $\times(\ell_1, \ell_2, \cdots, \ell_n)$ constructs an acyclic bridge-less SM WF-net. It is WS. From Item (ii)-c) of Def. 3 $\wedge(\ell_1, \ell_2, \cdots, \ell_n)$ constructs an acyclic bridge-less MG WF-net. It is WS. From Item (iii) of Def. 3, for each operator $\oplus$, $\oplus(\pi_1, \pi_2, \cdots, \pi_n)$ constructs a WF-net obtained by combining acyclic bridge-less WS WF-nets. Therefore the obtained WF-net is also acyclic, bridge-less and well-structured. ■

Theorem 1: A WF-net N is convertible to a process tree iff N is acyclic, bridge-less and WS. ■

This theorem means the necessary and sufficient condition on the convertibility problem. By using the necessary and sufficient condition, let us decide whether N_1 shown in Fig. 2(a) is PTB. N_1 is acyclic bridge-less WS. So N_1 is PTB i.e. convertible to a process tree. However, the process tree is not known, so we need to convert N_1 into the process tree.

IV. Conversion Algorithm of Workflow Net to Process Tree

A. Algorithm

We propose an algorithm to convert a WF-net N to a process tree based on Depth-First Search (DFS for short). Let f_π denote a formula of a process tree π. Set $f_\pi \leftarrow \varepsilon$ (the empty string). Let n $(\in P \cup T)$ be the most recently finished node[3] in the DFS.

- If n is a transition
  - If $|n\overset{N}{\bullet}| \geq 2$, then a parallel routing starts from this node. So we add '$\wedge(\rightarrow($' to f_π.
  - If $|\overset{N}{\bullet}n| \geq 2$, then a parallel routing ends at this node. So we add ')' to f_π.
  - Otherwise, $f_\pi \leftarrow f_\pi + n$.
- If n is a place
  - If $|n\overset{N}{\bullet}| \geq 2$, then a choice routing starts from this node. So we add '$\times(\rightarrow($' to f_π.
  - If $|\overset{N}{\bullet}n| \geq 2$, then a choice routing ends at this node. So we add ')' to f_π.

For nodes we have once visited we set its color as *gray*. If all of its input nodes $\overset{N}{\bullet}n$ has been visited and finished we set the color to *black*. The algorithm is given as follows:

≪Process Tree Conversion Algorithm≫

Input: PTB WF-net, $(N, [p_I])$
Output: Process tree formula, f_π

MakeProcessTree(N, π)
  $f_\pi \leftarrow \varepsilon$
  for each $n \in P \cup T$
    color$(n) \leftarrow white$
  VisitNode(p_I)
  Output f_π, and stop

VisitNode(n)
  if $n \in T$ **then**
    if $|n \bullet| \geq 2$ **then**
      $f_\pi \leftarrow f_\pi$ + '$\wedge(\rightarrow($'
    else
      $f_\pi \leftarrow f_\pi + n$
    if $\forall u \in \bullet n$: color(u) is $black$ **then**
      color$(n) \leftarrow black$
    else

[1]The set S of ST-net is the smallest set of nets N defined as follows: (i) If N is a WF-net then $N \in S$; (ii) If N is an acyclic MG WF-net then $N \in S$; (iii) If $N \in S$, p is a place in N, and $M \in S$ is a tWF-net then $N \otimes_p M \in S$; (iv) If $N \in S$, t is a transition in N, and $M \in S$ is a tWF-net then $N \otimes_t M \in S$.

[2]Let $\mathcal{N}$ be a WF-net. Refinement of a place p in $\mathcal{N}$ with a WF-net $\mathcal{M}$ yields a WF-net, denoted by $\mathcal{N} \otimes_p \mathcal{M}$, built as follows: p is replaced in $\mathcal{N}$ by $\mathcal{M}$; transitions in $\overset{\mathcal{N}}{\bullet}p$ become input transitions of the source place of $\mathcal{M}$, and transitions in $p\overset{\mathcal{N}}{\bullet}$ become output transitions of the sink place of $\mathcal{M}$. Refinement of a transition t in $\mathcal{N}$ with a tWF-net $\mathcal{M}$ yields a WF-net, denoted by $\mathcal{N} \otimes_t \mathcal{M}$, built as follows: t is replaced in $\mathcal{N}$ by $\mathcal{M}$; places in $\overset{\mathcal{N}}{\bullet}t$ become input places of the source transition of $\mathcal{M}$, and places in $t\overset{\mathcal{N}}{\bullet}$ become output places of the sink transition of $\mathcal{M}$.

[3]A node is said to be finished if all of its input nodes have been explored.

TABLE I. ALGORITHM EXECUTION AT EACH NODES OF N_1

P	T	$\|V_I\|$	$\|V_O\|$	Extend to f_π
p_1		0	1	
	t_1	1	2	$\wedge(\rightarrow($
p_2		1	1	
	t_2	1	1	t_2
p_4		1	1	
	t_5	1	1	t_5
p_7		1	1	
	t_8	2	1	)
p_3		1	2	$\times(\rightarrow($
	t_3	1	1	t_3
p_5		1	1	
	t_6	1	1	t_6
p_8		2	1	$)\rightarrow($
	t_4	1	1	t_4
p_6		1	1	
	t_7	1	1	t_7
p_8		2	1	)
	t_8	2	1	))
p_9		1	0	

    if color(n) is not $gray$
      color(n) $\leftarrow gray$
      $f_\pi \leftarrow f_\pi$ + ')'
if $n \in P$ **then**
  if $|n\bullet| \geq 2$ **then**
    $f_\pi \leftarrow \pi$ + '×(→('
  if $\forall u \in \bullet n$: color(u) is $black$ **then**
    color(n) $\leftarrow black$
  else
    if color(n) is not $gray$
      color(n) $\leftarrow gray$
      $f_\pi \leftarrow f_\pi$ + ')'
if color(n) is $black$ **then**
  if $n \in T$ and $|\bullet n| \geq 2$ **then**
    $f_\pi \leftarrow f_\pi$ + ')'
    for each $t \in n\bullet$ **then**
      VISITNODE(t)

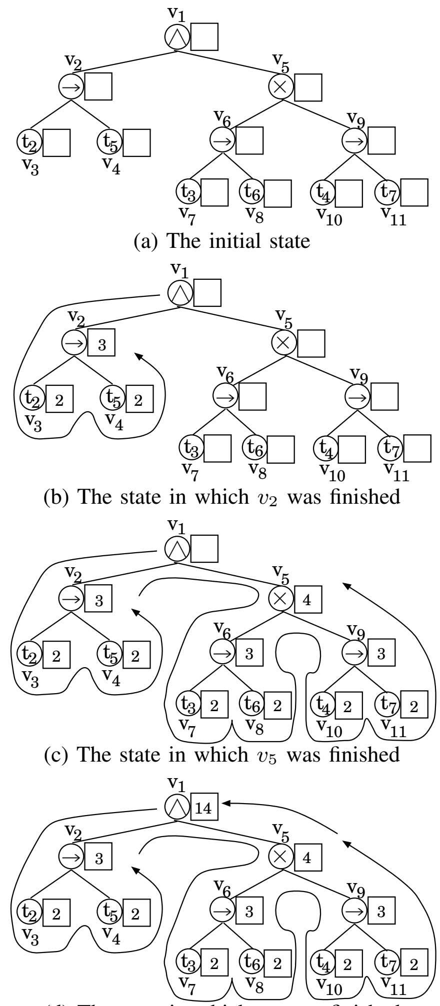
This algorithm can run in polynomial time because it is based on DFS.

B. Example

We illustrate the proposed algorithm with an application example to PTB WF-net N_1 shown in Fig. 2(a). Table I shows the execution of the proposed algorithm. Table I shows the table for each node visited and the process tree formula f_π at certain node progress. All nodes are recursively searched and only transition nodes is stacked into f_π. The procedure is recursively repeated until $n\bullet = \emptyset$. The process tree for PTB WF-net in Fig. 2(a) can be represented as $\wedge(\rightarrow(t_2, t_5), \times(\rightarrow(t_3, t_6), \rightarrow(t_4, t_7)))$. The process tree diagram is shown in Fig. 3.

V. APPLICATION

A tree representation of WF-net has a lot of usage especially in process mining and to ease model analysis such as state number calculation. Susaki et al. [5] proposed a process tree based state number calculation algorithm. The proposed algorithm is based on DFS. Let v be the most recently finished node[4] in the DFS, $s(v)$ is the number of state at v which is calculated according to Property 4 of Ref. [5].



(a) The initial state
(b) The state in which v_2 was finished
(c) The state in which v_5 was finished
(d) The state in which v_1 was finished

Fig. 5. The execution of the proposed algorithm for the example shown in Fig. 2 (a)

≪State number calculation of PTB WF-net≫

Input: PTB WF-net $(N, [p_I])$, its process tree π
Output: $|R(N, [p_I])|$
CALCULATESTATENUMBERPTBWF-NET($(N, [p_I]), \pi$)
$v \leftarrow$ the root of π
CALCULATESTATENUMBER(v)
Output $s(v)$ as $|R(N, [p_I])|$, and stop

CALCULATESTATENUMBER(v)
if v is a leaf node
  $s(v) \leftarrow 2$
if v is '→'
  for each child u of v
    CALCULATESTATENUMBER(u)
  $s(v) \leftarrow \sum_{\text{child } u \text{ of } v}(s(u) - 1) + 1$

[4] A node is said to be finished if all of its children nodes have been explored.

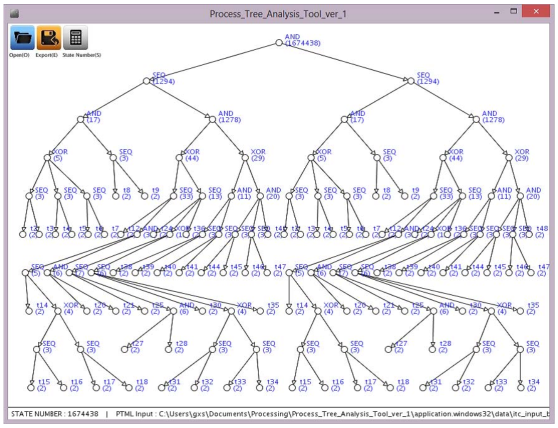


Fig. 6. Process Tree Analysis Tool version 1.0

if v is '$\times$'
  for each child u of v
    CALCULATESTATENUMBER(u)
  $s(v) \leftarrow \sum_{\text{child } u \text{ of } v}(s(u)-2)+2$
if v is '$\wedge$'
  for each child u of v
    CALCULATESTATENUMBER(u)
  $s(v) \leftarrow \prod_{\text{child } u \text{ of } v} s(u)+2$

This algorithm can run in polynomial time because it is based on DFS. Ref. [5] illustrated the proposed algorithm with an application example to PTB WF-net $(N_1, [p_1])$ shown in Fig. 2 (a). Figure 5 shows the execution. (a) The initial state. For each node v, the rectangle of its rightside represents $s(v)$. (b) The state in which v_2 was finished in the DFS. From equation $s(v) = \sum_{i=1}^{n}(s(v_i)-1)+1$ which v is sequence ($\rightarrow$), then we have $s(v_2) = (s(v_3)-1)+(s(v_4)-1)+1 = 3$. (c) The state in which v_5 was finished. From equation $s(v) = \sum_{i=1}^{n}(s(v_i)-2)+2$ which v is choice ($\times$), then we have $s(v_5) = (s(v_6)-2)+(s(v_9)-2)+2 = 4$. (d) The state in which v_1 was finished. From equation $s(v) = \prod_{i=1}^{n} s(v_i)+2$ which v is parallel ($\wedge$), then we have $s(v_1) = s(v_2) \times s(v_5)+2 = 14$. Thus the algorithm outputs 14 as $|R(N_1, [p_1])|$. In fact, $R(N_1, [p_1]) = \{[p_1], [p_2, p_3], [p_2, p_5], [p_2, p_6], [p_2, p_8], [p_4, p_3], [p_4, p_5], [p_4, p_6], [p_4, p_8], [p_7, p_3], [p_7, p_5], [p_7, p_6], [p_7, p_8], [p_9]\}$. This means $|R(N_1, [p_1])|$=14.

We developed a tool to analyse and visualize a process tree called the Process Tree Analysis Tool (ProTAT for short). We can visualize a process tree described in a XML based format that we called as Process Tree Markup Language (PTML for short) within ProTAT. The tool was developed with Processing 2.0 and Java. It also enabled us to calculate the state number of the process tree using the ≪State number calculation of PTB WF-net≫. Figure 6 shows an example of state number calculation. The process tree formula in Fig. 6 is as follows: $\rightarrow(\wedge(\times(\rightarrow(t_{12}, \wedge(\rightarrow(t_{14}, \times(\rightarrow(t_{15}, t_{16}), \rightarrow(t_{17}, t_{18}))), \wedge(t_{20}, t_{21}))), \rightarrow(t_{24}, \times(\rightarrow(t_{25}, \wedge(t_{27}, t_{28})), t_{30}, \times(\rightarrow(t_{31}, t_{32}), \rightarrow(t_{33}, t_{34})), t_{35}), t_{36})), \times(\wedge(\rightarrow(t_{38}, t_{39}), \rightarrow(t_{40}, t_{41})), \wedge(\rightarrow(t_{44}, t_{45}), \rightarrow(t_{46}, t_{47}), t_{48}))), \wedge(\times(\rightarrow(t_2, t_3), \rightarrow(t_4, t_5), \rightarrow(t_6, t_7)), \rightarrow(t_8, t_9)))$. Using ProTAT, we could obtain the state number as 1,674,438. The calculation took at most one second on a personal computer with Intel Core i7 2.4 Ghz and 8.0 GB memory.

VI. CONCLUSION

In this paper, we discussed the convertibility and conversion of WF-nets to process trees. We first proposed a necessary and sufficient condition on convertibility, i.e. a WF-net is convertible to a process tree iff it is acyclic, bridge-less and WS. Next, based on the condition we constructed an algorithm to convert the WF-net to a process tree. Then we proposed an application of the conversion algorithm, which allows us to calculate the state number of the WF-net.

In the future work, we planned to show the computation complexity of the acyclic, bridge-less and well-structuredness of WF-nets.

ACKNOWLEDGMENT

This research was partially supported by JSPS KAKENHI Grant Number 23560532.

REFERENCES

[1] T. Murata, "Petri nets: Properties, analysis and applications," Proceedings of the IEEE, vol.77, no.4, pp.541–580, 1989.

[2] W.M.P. van der Aalst, K. M. van Hee, *Workflow Management: Models, Methods, and Systems*, The MIT Press, 2002.

[3] S. Yamaguchi, S. Nishi, "Extract Superclass Problem of Workflow Nets and a Solution Method," Proc. ITC-CSCC 2013, pp.80–83, 2013.

[4] 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.

[5] T. Susaki, S. Yamaguchi, "On Process Tree Based Calculation of the Number of States in Petri Nets," Proceedings, ITC-CSCC 2013, pp.84–87, 2013.7.

[6] J. Esparza and M. Silva, "Circuits, handles, bridges and nets," Lecture Notes in Computer Science, vol.483, pp.210–242, 1990.

[7] K. M. van Hee, N. Sidorova, M. Voorhoeve, "Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach," 24th International Conference, ICATPN 2003 Eindhoven, The Netherlands, Proceedings, vol.2679, pp.337–356, Springer-Verlag, Berlin, 2003.