State Number Calculation Problem of Workflow Nets

Article *in* IEICE Transactions on Information and Systems · June 2015

DOI: 10.1587/transinf.2014FOP0009

Shingo Yamaguchi
Yamaguchi University
125 PUBLICATIONS **384** CITATIONS

PAPER *Special Section on Formal Approach*

State Number Calculation Problem of Workflow Nets

Mohd Anuaruddin BIN AHMADON†, ***Nonmember and*** **Shingo YAMAGUCHI**†a), ***Senior Member***

SUMMARY The number of states is a very important matter for model checking approach in Petri net's analysis. We first gave a formal definition of state number calculation problem: For a Petri net with an initial state (marking), how many states does it have? Next we showed the problem cannot be solved in polynomial time for a popular subclass of Petri nets, known as free choice workflow nets, if $P \neq NP$. Then we proposed a polynomial time algorithm to solve the problem by utilizing a representational bias called as process tree. We also showed effectiveness of the algorithm through an application example.

key words: *Petri net, state number calculation problem, process tree, solvability, computational complexity, model checking*

1. Introduction

Petri nets [1] are a mathematical and graphical modeling tool applicable to many systems. Once we model a system as a Petri net, we can simulate the behaviour of the system by using tokens on the Petri net. We can also analyse the behaviour of the system exhaustively by enumerating all possible token distributions (states). Unfortunately, the number of all the possible states is of exponential order in the size of the Petri net. This is called the state space explosion. For example, Fig. 1 shows a Marked Graph (MG for short) with i parallel paths. The state number is $2^i + 2$ which increases exponentially with the number of parallel paths. The problem seems to be unsolvable in polynomial time if we try to enumerate all the possible states.

Petri net's state number is useful for analysis method that involves behavioural analysis such as in model checking approach. Model checking has been attracting attention as a promising approach to analysis of Petri nets. SPIN [2], a popular model checking tool, is available only to Petri nets with less than 1 million states, because SPIN basically enumerates all possible states in the Petri net. We need a polynomial time solution to accurately calculate the state number of the given Petri net before using SPIN.

In 2011, Chao et al. [3] proposed a method to calculate the number of all the possible states. They first transformed a given Petri net to an algebraic expression, and then calculated the number of all the possible states by utilizing the algebraic expression. This method is, however, available only to a simple subclass i.e. MG and State Machine (SM for short). In addition, the computational complexity has not been discussed.

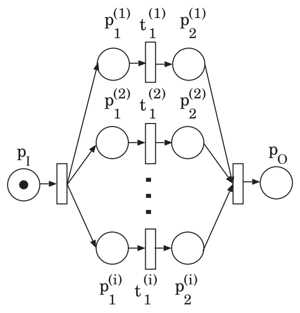


Fig. 1 Illustration of a MG$_i$ with $(i+1)$ parallel paths. The state number is 2^i+2.

In this paper, we first give a formal definition of state number calculation problem and show the computation complexity of the problem for a popular subclass of Petri nets, i.e. free choice workflow nets (FC WF-net for short). Then we propose a polynomial time algorithm to solve the problem for a Petri net which can be represented as a process tree. This paper is organized as follows: After the introduction in Sect. 1, Sect. 2 gives the definition and properties of Petri net, workflow net [4] and process tree [5]. In Sect. 3, we formalize the state number calculation problem and reveal the solvability and computational complexity of the problem. In Sect. 4, we formalize the convertibility problem of WF-net to process tree. In Sect. 5, we define a subclass of WF-net which can be represented as process tree, then we propose a polynomial time algorithm for the state number calculation. In Sect. 6, we show evaluation for our solution and an application example of our method in model checking approach. Finally, we give the conclusion and the future work.

2. Preliminary

(1) Petri Nets and Workflow Nets

A 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 $M\colon P\to\mathbb{N}$. 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 $M[N,t\rangle M'$.

Manuscript received August 27, 2014.
Manuscript revised December 19, 2014.
Manuscript publicized February 13, 2015.
†The authors are with the Graduate School of Science and Engineering, Yamaguchi University, Ube-shi, 755–8611 Japan.
a) E-mail: shingo@yamaguchi-u.ac.jp
DOI: 10.1587/transinf.2014FOP0009

A marking M_n is said to be reachable from a marking M_0 if there exists a transition sequence $t_1 t_2 \cdots t_n$ 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. Each transition represents an action. There is a particular subclass of WF-nets: *well-structured* (WS for short). 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 $h=x_1 x_2 \cdots x_n$ $(n\geq 2)$ is called a *handle* [6] of c if h shares exactly two nodes, x_1 and x_n, with c. A path b is called a bridge between c and h if each of c and h shares exactly one node, x_1 or x_n, with b. A handle (a bridge) from a node x to another node y is called a XY-handle (a XY-bridge), 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. A WF-net N is said to be WS if there are neither TP-handles nor PT-handles of any circuit in $\overline{N}$.

(2) Soundness

Soundness is a criterion of correctness for WF-nets. A WF-net N $(=(P, T, A))$ is said to be *sound* iff (i) $\forall M\in R(N, [p_I])$: $\exists M'\in R(N, M)$: $M'\geq[p_O]$; (ii) $\forall M\in R(N, [p_I])$: $M\geq[p_O]\Rightarrow M=[p_O]$; and (iii) There is no dead transition in $(N, [p_I])$. The soundness problem for EFC WF-nets or WS WF-nets can be solved in polynomial time (Corollaries 1 and 2 of Ref. [4]).

(3) Process Tree

A process tree [5] is a tree representation of a process in WF-nets. Each leaf node represents an action (transition) and each internal node represents a routing operator in the process respectively. Process tree was originally proposed for process mining. The major purpose is to ensure soundness of the WF-net discovered from the mining. Reference [5] proposed a genetic algorithm to discover a WF-net from an event log.

This paper uses three routing operators [7]: sequence SEQ ($\rightarrow$), exclusive choice XOR ($\times$) and parallel AND ($\wedge$). Each operator can be translated to a part of a WF-net as shown in Fig. 2. Any process tree can also be represented as a formula. For example, the process tree shown in Fig. 2 (a) is represented as $\rightarrow(\alpha, \beta)$. The order of the child nodes in the formula must follow the sequence from left to right as represented in the tree.

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

(i) If ι is an action label, then $\iota \in \Pi$.
(ii) If $\oplus$ is an operator and $\iota_1, \iota_2, \cdots, \iota_n$ are action labels, then $\oplus(\iota_1, \iota_2, \cdots, \iota_n) \in \Pi$.
(iii) If $\oplus$ is an operator and $\pi_1, \pi_2, \cdots, \pi_n \in \Pi$, then $\oplus(\pi_1, \pi_2, \cdots, \pi_n) \in \Pi$. □

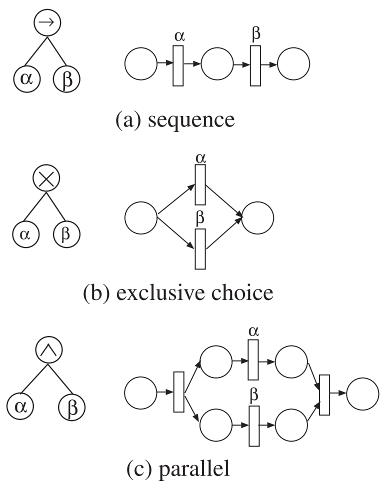


Fig. 2 Translation of process tree operators to Petri net constructs.

3. State Number Calculation Problem and Its Properties

In this section, we formalize a problem, named state number calculation problem [8], that calculates the number of all possible states in a given Petri net. Then we reveal the solvability and the computational complexity of the problem. The formal definition of the problem is given as follows:

Definition 2 (State number calculation problem):
Instance: Petri net (N, M_0)
Question: How many states are there in $R(N, M_0)$? □

As an example, in the case of WF-net of MG$_i$ shown in Fig. 1, the problem is how many states there are in $R(MG_i, [p_I])$.

Let us consider the solvability of the state number calculation problem.

Property 1: The state number calculation problem is solvable. □

Proof: Let (N, M_0) be any Petri net. The state number calculation problem can be divided into two cases by the boundedness of (N, M_0). The boundedness problem is known to be decidable [10]. If (N, M_0) is bounded, we have only to count the nodes in the reachability tree of (N, M_0). Otherwise, (if (N, M_0) is unbounded), we can regard $|R(N, M_0)|$ as ∞, where $\forall n\in\mathbb{N} : \infty > n$, $\infty \pm n = \infty$ and $\infty\geq\infty$. **Q.E.D.**

Let us consider the state number calculation problem of $(MG_i, [p_I])$ shown in Fig. 1. Since $(MG_i, [p_I])$ is bounded, we can solve the problem by using its reachability tree [1]. Unfortunately, $(MG_i, [p_I])$ has 2^i+2 markings. For example, to calculate the state number for the MG with i=20, we have to count 1,048,578 markings. In general, we cannot solve the problem by enumerating all the states in practical time.

Then, let us consider the computation complexity of the state number calculation problem. In this paper, we assume that P and NP are not equivalent, i.e. $P{\neq}NP$. An NP-hard problem cannot be solved in polynomial time. We call the problem as intractable. We show that the state number calculation problem is intractable for FC WF-nets with initial marking $[p_I]$. To prove the intractability, we tackle the following decision version of the state number calculation problem: Given a Petri net (N, M_0), to decide whether $|R(N, M_0)|{\geq}\infty$. This decision problem is the boundedness problem. We have only to show that the boundedness problem is intractable for FC WF-nets with initial marking $[p_I]$.

To do so, we show that an NP-complete problem, called 3-conjunctive normal form boolean satisfiability problem [11] (3-CNF-SAT for short), can be transformed to the complement of the boundedness problem, i.e. the unboundedness problem of FC WF-nets with initial marking $[p_I]$.

Definition 3 (3-CNF-SAT [11]):
Instance: Expression $\mathcal{E}$ of 3-conjunctive normal form that has n boolean variables and m clauses.
Question: Is there an assignment of variables satisfying $\mathcal{E}{=}true$? □

Lemma 1: The boundedness problem is co-NP-hard for FC WF-nets with initial marking $[p_I]$. □

Proof: We prove the co-NP-hardness by a reduction from 3-CNF-SAT in a way similar to Ref. [13]. Let $\mathcal{E}$ be an expression of 3-CNF-SAT which has n boolean variables $x_1, x_2, \cdots, x_n$ and m clauses $c_1, c_2, \cdots, c_m$. A literal ℓ_i is either a variable x_i or its negation $\overline{x_i}$. Without loss of generality, it can be assumed that $\mathcal{E}$ has all of x_i's and $\overline{x_i}$'s $(i{=}1, 2, \cdots, n)$, and $m{\geq}3$ [12]. We first construct the following Petri net $N_\mathcal{E}{=}(P_\mathcal{E}, T_\mathcal{E}, A_\mathcal{E})$.

$$\begin{aligned}
P_\mathcal{E} &= \{p_I, p_1, p_O\} \cup \textstyle\bigcup_{i=1}^{n}\{q_i\} \cup \bigcup_{j=1}^{m}\{c_j\}\\
T_\mathcal{E} &= \{t_1, t_2, t_3\} \cup \textstyle\bigcup_{i=1}^{n}\{x_i, \overline{x_i}\}\\
A_\mathcal{E} &= \{(p_I, t_1), (t_2, p_1), (p_1, t_3), (t_3, p_1), (t_3, p_O)\}\\
&\quad \cup \textstyle\bigcup_{i=1}^{n}\{(t_1, q_i), (q_i, x_i), (q_i, \overline{x_i})\}\\
&\quad \cup \textstyle\bigcup_{k=1}^{3}\bigcup_{j=1}^{m}\{(\ell_k, c_j) | \ell_k \text{ is the } k\text{-th literal of clause } c_j\}\\
&\quad \cup \textstyle\bigcup_{j=1}^{m}\{(c_j, t_2)\}
\end{aligned}$$

$N_\mathcal{E}$ is an FC WF-net because its short-circuited net $\overline{N_\mathcal{E}}$ is strongly connected; Places $c_1, c_2, \cdots, c_m$ share only one output transition t_2, and the other places share no output transition. $N_\mathcal{E}$ can be constructed in polynomial time, because it consists of $(n{+}m{+}3)$ places, $(2n{+}3)$ transitions, and $(3n{+}4m{+}5)$ arcs.

Let us prove that $(N_\mathcal{E}, [p_I])$ is unbounded iff there is an assignment of variables satisfying $\mathcal{E}{=}true$.

The proof of "if" part: Let α denote an assignment of variables satisfying $\mathcal{E}{=}true$, and let $\ell_1, \ell_2, \cdots, \ell_n$ be the literals mapped to *true* by α. By the construction of $N_\mathcal{E}$, we have

$$\begin{aligned}
&[p_I]\ [N_\mathcal{E}, t_1\rangle\, [q_1, q_2, \cdots, q_n]\\
&\qquad [N_\mathcal{E}, \ell_1\ell_2\cdots\ell_n\rangle\, M\ ({\geq}\ [c_1, c_2, \cdots, c_m]).
\end{aligned}$$

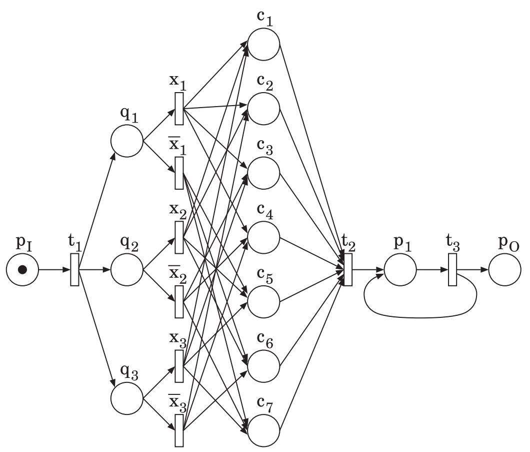


Fig. 3 The FC WF-net $(N_{\mathcal{E}_1}, [p_I])$ corresponding to a 3-CNF-SAT expression $\mathcal{E}_1 = (x_1{\vee}x_2{\vee}x_3) \wedge (x_1{\vee}x_2{\vee}\overline{x_3}) \wedge (x_1{\vee}\overline{x_2}{\vee}x_3) \wedge (x_1{\vee}\overline{x_2}{\vee}\overline{x_3}) \wedge (\overline{x_1}{\vee}x_2{\vee}x_3) \wedge (\overline{x_1}{\vee}x_2{\vee}\overline{x_3}) \wedge (\overline{x_1}{\vee}\overline{x_2}{\vee}x_3)$. $(N_{\mathcal{E}_1}, [p_I])$ is unbounded.

We are to show that $M{\geq}[c_1, c_2, \cdots, c_m]$. Since $N_\mathcal{E}$ is FC, we can freely choose, at every place q_i, between letting transition x_i or $\overline{x_i}$ fire. Since α satisfies $\mathcal{E}$, for each clause c_j $(1{\leq}j{\leq}m)$, there exists a literal ℓ_i $(1{\leq}i{\leq}n)$ in c_j. Therefore place c_j is marked by firing ℓ_i. As a result, we have

$$\begin{aligned}
&M\ \ [N_\mathcal{E}, t_2\rangle M'\ (= M\backslash[c_1, c_2, \cdots, c_m]\cup[p_1])\\
&\qquad [N_\mathcal{E}, t_3\rangle M'\cup[p_O].
\end{aligned}$$

Since $M'\cup[p_O]$ covers M', $(N_\mathcal{E}, [p_I])$ is unbounded.

The proof of "only if" part: Let α denote any assignment of variables satisfying $\mathcal{E}{=}false$. Since α does not satisfy $\mathcal{E}$, there exists a clause c_j $({\in}\{c_1, c_2, \cdots, c_m\})$ mapped to *false* by α. Let $\ell_1^j, \ell_2^j, \ell_3^j$ denote the literals in c_j. Since the corresponding transitions $\ell_1^j, \ell_2^j, \ell_3^j$ do not fire, their common output place, i.e. place c_j, is never marked. c_j is an input place of transition t_2, so t_2 is dead. This enables us to ignore the part following t_2 in $N_\mathcal{E}$. The remaining part is acyclic. Since any acyclic Petri net is bounded, $(N_\mathcal{E}, [p_I])$ is bounded.

Q.E.D.

For example, let us consider the following boolean expression:

$$\begin{aligned}
\mathcal{E}_1 = &(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3)\\
&\wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})\\
&\wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)
\end{aligned}$$

$\mathcal{E}_1$ is satisfiable by choosing $x_1{=}true$, $x_2{=}true$, $x_3{=}true$. Figure 3 shows the Petri net $N_{\mathcal{E}_1}$ constructed from $\mathcal{E}_1$. $(N_{\mathcal{E}_1}, [p_I])$ is unbounded, because

$$\begin{aligned}
&[p_I]\, [N_{\mathcal{E}_1}, t_1\rangle\, [q_1, q_2, q_3]\\
&\qquad [N_{\mathcal{E}_1}, x_1x_2x_3\rangle\, [{c_1}^3, {c_2}^2, {c_3}^2, c_4, {c_5}^2, c_6, c_7]\\
&\qquad [N_{\mathcal{E}_1}, t_2\rangle\, [{c_1}^2, c_2, c_3, c_5, p_1]\\
&\qquad [N_{\mathcal{E}_1}, t_3\rangle\, [{c_1}^2, c_2, c_3, c_5, p_1, p_O].
\end{aligned}$$

From Lemma 1, we can obtain the following theorem.

Theorem 1: The state number calculation problem cannot be solved in polynomial time for FC WF-nets with initial marking $[p_I]$ if $P \neq NP$. □

Proof: The decision problem related to this problem, i.e. the boundedness problem, is co-NP-hard. This means that the original problem is intractable. **Q.E.D.**

4. Convertibility of Workflow Net to Process Tree

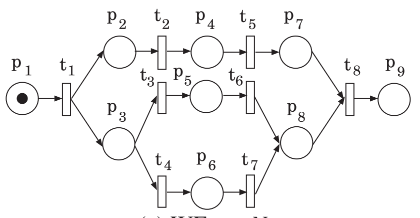
We showed that the state number calculation problem is intractable but we cannot give up solving the problem because the problem is important for analysing workflows. We try to utilize process trees to solve the state number calculation problem. The structure of process tree allows us to calculate state number without enumerating all states. Unfortunately, not all WF-nets are always convertible to process trees. For example, non-sound WF-nets are not convertible because the process tree itself is the representational bias of sound WF-net as described in [5]. Soundness is a necessary condition but is not sufficient. It is necessary to decide whether a given WF-net is convertible to a process tree or not. We call this problem as convertibility problem. In this section, we first give a formal definition of convertibility problem. Then we give a necessary and sufficient condition on the problem.

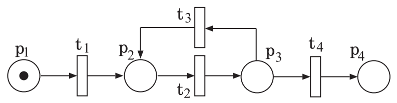
4.1 Convertibility Problem

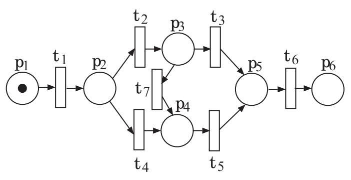
We formalize convertibility problem as follows:

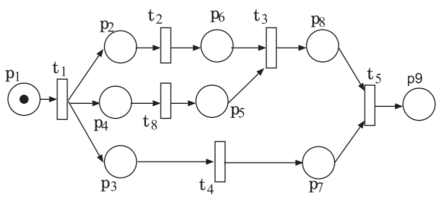
Definition 4 (Convertibility problem):
Instance : WF-net N
Question : Is N convertible to a process tree? □

Let us consider five instances of convertibility problem for example. The first instance is a WF-net N_1 shown in Fig. 4 (a). This WF-net can be represented as a process tree as shown in Fig. 5. By looking at Fig. 4 (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. 4 (b). N_2 has a circuit $p_2t_2p_3t_3p_2$. In this paper, we use no operator representing circuits. 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. 4 (c). N_3 has a bridge $p_3t_7p_4$. Originally without the bridge (path $p_3t_7p_4$), paths $p_2t_2p_3t_3p_5$ and $p_2t_4p_4t_5p_5$ construct an exclusive choice but since bridge $p_3t_7p_4$ exists, $t_2p_3t_7p_4t_5$ forms a new sequence relation connecting the path. So actions t_2 and t_5 have two relations, an exclusive choice and a sequence. It is not convertible because one process tree operator can only represent one routing relation between actions. The forth instance is a WF-net N_4 shown in Fig. 4 (d). N_4 has a path $t_1p_2t_2p_6t_3p_8t_5$ with a handle $t_1p_3t_4p_7t_5$. There exists a path $t_1p_4t_8p_5t_3$ between the path and its handle. Path $t_1p_4t_8p_5t_3$ is similar to a bridge but is not exactly a bridge. We call it "pseudo-bridge". It is not convertible because 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. In this paper, we call a WF-net N as "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. 4 (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.


(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

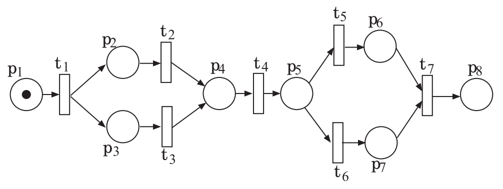

(e) A WF-net with a TP-handle and a PT-handle N_5

Fig. 4 Example of WF-net instances.

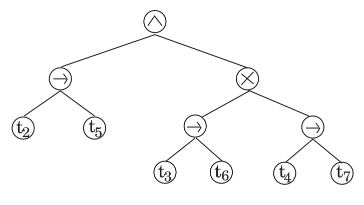
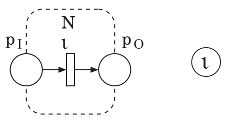
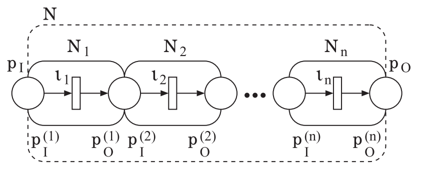


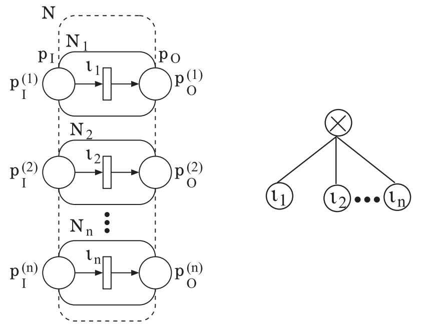
Fig. 5 Process tree of N_1.



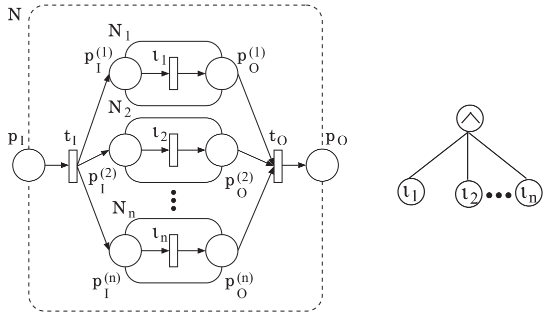
(a) PTB WF-net representing an action label ι



(b) PTB WF-net representing $\rightarrow(\iota_1, \iota_2, \cdots, \iota_n)$



(c) PTB WF-net representing $\times(\iota_1, \iota_2, \cdots, \iota_n)$



(d) PTB WF-net representing $\wedge(\iota_1, \iota_2, \cdots, \iota_n)$

Fig. 6 Illustration of PTB WF-net and its equivalent process tree.

4.2 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 5 (PTB WF-net): For any process tree π, let N be the WF-net itself and N_i (i=1, 2, $\cdots$, n) be the subnet in N. Each p_I and p_O is the source place and the sink place of N, while each $p_I{}^{(n)}$ and $p_O{}^{(n)}$ is the source place and sink place of N_n. See Fig. 6 (the broken lines illustrate the boundaries of internal structure of WF-net N).

(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. 6 (a)).

(ii) If π is $\oplus(\iota_1, \iota_2, \cdots, \iota_n)$ then let $N_1, N_2, \cdots, N_n$ be respectively PTB WF-nets representing action labels $\iota_1, \iota_2, \cdots, \iota_n$.

  a. If $\oplus$ is sequence ($\rightarrow$) then a WF-net constructed by concatenating $N_1, N_2, \cdots, N_n$ which link the sink place of N_i with the source place of N_{i+1}($1 \leq i < n$) is PTB (See Fig. 6 (b)).
  b. If $\oplus$ is exclusive choice ($\times$) then a WF-net constructed by bundling $N_1, N_2, \cdots, N_n$ which forms a selection of concurrent paths between their source places and sink places is PTB (See Fig. 6 (c)).
  c. If $\oplus$ is parallel ($\wedge$) then a WF-net which is constructed by joining respectively all source places with a transition t_I, and sink places with a transition t_O of PTB WF-net $N_1, N_2, \cdots, N_n$ is PTB (See Fig. 6 (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 constructed by concatenating $N_1, N_2, \cdots, N_n$ which 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 exclusive choice then a WF-net constructed by bundling PTB WF-nets $N_1, N_2, \cdots, N_n$ which forms a selection of concurrent paths between their source places and sink places is PTB.
  c. If $\oplus$ is parallel then a WF-net constructed by joining respectively all source places with a transition t_I, and sink places with a transition t_O of PTB WF-net $N_1, N_2, \cdots, N_n$ is PTB. □

N_1 shown in Fig. 4 (a) is PTB. Let us construct N_1 from the process tree shown in Fig. 5. For $\rightarrow(t_2, t_5)$ we construct a WF-net composed of a path $p_2t_2p_4t_5p_7$ based on Item (ii)-a) of Def. 5. 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 can obtain N_1 by bundling those WF-nets.

Lemma 2: 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. [15]'s ST-net†. We show the following: (i) An acyclic bridge-less WS WF-net N is an ST-net. (ii) An acyclic, bridge-less ST-net is PTB.

We first show that an acyclic bridge-less WS WF-net N is an ST-net. Intuitively, ST-nets are constructed from SMs

†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 $N \otimes_t M \in S$.

and MGs by means of refinement[†]. 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 $\mathcal{M}$ 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 an ST-net.

Next we show that an acyclic 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. 5 $\rightarrow(\iota_1, \iota_2, \cdots, \iota_n)$ constructs an WF-net which is a path. It is acyclic, bridge-less and WS. From Item (ii)-b) of Def. 5 $\times(\iota_1, \iota_2, \cdots, \iota_n)$ constructs an acyclic bridge-less SM WF-net. It is WS. From Item (ii)-c) of Def. 5 $\wedge(\iota_1, \iota_2, \cdots, \iota_n)$ constructs an acyclic bridge-less MG WF-net. It is WS. From Item (iii) of Def. 5, 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 WS. **Q.E.D.**

Theorem 2: 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. Any acyclic WS WF-net is sound [16], so all the acyclic bridge-less WS WF-nets are sound. This coincides with van der Aalst's necessary condition on convertibility. All acyclic WS WF-nets, however, cannot always be converted to process trees because some of them have bridges. This is the difference between van der Aalst's necessary condition and our necessary and sufficient condition.

[†]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 $\mathcal{M}$.

By using the necessary and sufficient condition, let us decide whether N_1 shown in Fig. 4 (a) is PTB. N_1 is acyclic bridge-less WS. So N_1 is PTB i.e. convertible to a process tree.

Lemma 3: The following problem can be solved in polynomial time: Given a WF-net N, to decide whether N is PTB. □

Proof: We only have to show that each condition of Theorem 2 can be checked in polynomial time. Acyclicity is obviously decidable in polynomial time (See Ref. [17]). Bridge-less property can also be decided in polynomial time by searching for nodes connecting two parallel paths and handles that does not split and join at the same nodes (See Ref. [18]). We can also decide in polynomial time whether a given WF-net is WS by applying a modified version of the max-flow min-cut technique [4]. **Q.E.D.**

We have proposed a polynomial time algorithm to convert a PTB WF-net to a process tree in Ref. [14].

5. Process Tree Based State Number Calculation

In this section, we propose a polynomial time algorithm to calculate the state number by utilizing process tree.

Lemma 4: Let N and π be respectively a PTB WF-net and its process tree. For each node v of π, $\pi(v)$ denotes the subtree of π whose root is v, $N(v)$ denotes the subnet of N represented as $\pi(v)$, and $s(v)$ denotes the number of possible states in $N(v)$.

- If v is a leaf node then

$$s(v) = 2 \tag{1}$$

- If v is an internal node then, let $v_1, v_2, \cdots, v_n$ be the children of v,
  - If v is sequence ($\rightarrow$) then

$$s(v) = \sum_{i=1}^{n}(s(v_i) - 1) + 1 \tag{2}$$

  - If v is exclusive choice ($\times$) then

$$s(v) = \sum_{i=1}^{n}(s(v_i) - 2) + 2 \tag{3}$$

  - If v is parallel ($\wedge$) then

$$s(v) = \prod_{i=1}^{n} s(v_i) + 2 \tag{4}$$

□

Proof: If v is a leaf node, $N(v)$ is a PTB WF-net which consists of one transition and its input and output places. $N(v)$ is illustrated in Fig. 6 (a). $(N(v), [p_I])$ has two states, $[p_I]$ and $[p_O]$, before and after the firing of the transition. Since $|R(N(v), [p_I])|$=2, we have $s(v) = 2$ =Eq. (1).

If v is sequence ($\rightarrow$), $N(v)$ is a PTB WF-net constructed by concatenating PTB WF-nets $N(v_1), N(v_2), \cdots$, and $N(v_n)$ so as to unite the sink place of $N(v_i)$ and the source place of $N(v_{i+1})$ $(1 \le i < n)$. $N(v)$ is illustrated in Fig. 6 (b). Let $p_I^{(i)}$ and $p_O^{(i)}$ denote respectively the source place and the sink place of $N(v_i)$. In $N(v)$, $[p_I]$ $(=[p_I^{(1)}])$ is reachable to $[p_O^{(1)}]$, $[p_I^{(2)}]$ $(=[p_O^{(1)}])$ is reachable to $[p_O^{(2)}], \cdots, [p_I^{(n)}]$ $(=[p_O^{(n-1)}])$ is reachable to $[p_O^{(n)}]$ $(=[p_O])$, because $N(v_1), N(v_2), \cdots, N(v_n)$ is sound. Since $N(v_i)$ and $N(v_{i+1})$ share only $p_O^{(i)}$ $(= p_I^{(i+1)})$, $(N(v_i), [p_I^{(i)}])$ and $(N(v_{i+1}), [p_I^{(i+1)}])$ have different states except $[p_O^{(i)}]$ $(= [p_I^{(i+1)}])$. Therefore we have

$$\begin{aligned}
&R(N(v), [p_I])\\
&\quad = (R(N(v_1), [p_I^{(1)}]) \backslash \{[p_O^{(1)}]\}) \cup \cdots\\
&\qquad \cup (R(N(v_{n-1}), [p_I^{(n-1)}]) \backslash \{[p_O^{(n-1)}]\}) \cup R(N(v_n), [p_I^{(n)}])\\
&|R(N(v), [p_I])|\\
&\quad =(|R(N(v_1), [p_I^{(1)}])|-1) + \cdots\\
&\qquad +(|R(N(v_{n-1}), [p_I^{(n-1)}])|-1) + |R(N(v_n), [p_I^{(n)}])|\\
&s(v) = \sum_{i=1}^{n} (s(v_i) - 1) + 1 = \text{Eq. (2)}
\end{aligned}$$

If v is exclusive choice ($\times$), $N(v)$ is a PTB WF-net constructed by bundling PTB WF-nets $N(v_1), N(v_2), \cdots$, and $N(v_n)$ so as to unite respectively their source places and all their sink places. $N(v)$ is illustrated in Fig. 6 (c). Note that $p_I = p_I^{(1)} = p_I^{(2)} = \cdots = p_I^{(n)}$ and $p_O = p_O^{(1)} = p_O^{(2)} = \cdots = p_O^{(n)}$. Since $N(v_1)$, $N(v_2)$, $\cdots$, and $N(v_n)$ share only the source places and the sink places, $(N(v_1), [p_I^{(1)}])$, $(N(v_2), [p_I^{(2)}])$, $\cdots$, and $(N(v_n), [p_I^{(n)}])$ have different states except $[p_I]$ $(=[p_I^{(1)}]=[p_I^{(2)}] = \cdots = [p_I^{(n)}])$ and $[p_O]$ $(=[p_O^{(1)}]=[p_O^{(2)}] = \cdots = [p_O^{(n)}])$. Therefore we have

$$\begin{aligned}
&R(N(v), [p_I])\\
&\quad =(R(N(v_1), [p_I^{(1)}]) \backslash \{[p_I^{(1)}], [p_O^{(1)}]\}) \cup \cdots\\
&\qquad \cup (R(N(v_n), [p_I^{(n)}]) \backslash \{[p_I^{(n)}], [p_O^{(n)}]\}) \cup \{[p_I], [p_O]\}\\
&|R(N(v), [p_I])|\\
&\quad =(|R(N(v_1), [p_I^{(1)}])|-2) + \cdots\\
&\qquad +(|R(N(v_n), [p_I^{(n)}])|-2) + 2\\
&s(v) = \sum_{i=1}^{n} (s(v_i) - 2) + 2 = \text{Eq. (3)}
\end{aligned}$$

If v is parallel ($\wedge$), $N(v)$ is a PTB WF-net constructed by bundling PTB WF-nets $N(v_1), N(v_2), \cdots$, and $N(v_n)$ so as to have another source place and another sink place. $N(v)$ is illustrated in Fig. 6 (d). p_I is connected to $p_I^{(1)}$, $p_I^{(2)}$, $\cdots$, and $p_I^{(n)}$ via an additional transition t_I. This means that $[p_I][N(v), t_I\rangle[p_I^{(1)}, p_I^{(2)}, \cdots, p_I^{(n)}]$. Since $N(v_1)$, $N(v_2)$, $\cdots$, and $N(v_n)$ share no node, $(N(v_1), [p_I^{(1)}])$, $(N(v_2), [p_I^{(2)}])$, $\cdots$, and $(N(v_n), [p_I^{(n)}])$ have different states. Therefore $(N(v), [p_I^{(1)}, p_I^{(2)}, \cdots, p_I^{(n)}])$ has a combination of those states. Since $N(v_1)$, $N(v_2)$, $\cdots$, and $N(v_n)$ are sound, $[p_I^{(1)}, p_I^{(2)}, \cdots, p_I^{(n)}]$ is reachable to $[p_O^{(1)}, p_O^{(2)}, \cdots, p_O^{(n)}]$. $p_O^{(1)}$, $p_O^{(2)}$, $\cdots$, and $p_O^{(n)}$ are connected to p_O via another additional transition t_O. This means that $[p_O^{(1)}, p_O^{(2)}, \cdots, p_O^{(n)}]$ $[N(v), t_O\rangle[p_O]$. Therefore we have

$$\begin{aligned}
&R(N(v), [p_I])\\
&\quad = R(N(v_1), [p_I^{(1)}]) \times \cdots \times R(N(v_n), [p_I^{(n)}]) \cup \{[p_I], [p_O]\}\\
&|R(N(v), [p_I])|\\
&\quad = |R(N(v_1), [p_I^{(1)}])| \times \cdots \times |R(N(v_n), [p_I^{(n)}])| + 2\\
&s(v) = \prod_{i=1}^{n} s(v_i) + 2 = \text{Eq. (4)}
\end{aligned}$$

Q.E.D.

Based on Lemma 4, we propose a polynomial time algorithm to solve the problem for the PTB WF-nets. To calculate the number of all possible states in a PTB WF-net, the proposed algorithm utilizes its process tree. The proposed algorithm is based on Depth-First Search (DFS) [17]. The tree traversal is in post-order. Let v be the most recently finished node[†] in the DFS, $s(v)$ is the number of state at v which is calculated. We propose the algorithm as follows:

≪State Number Calculation of PTB WF-net≫

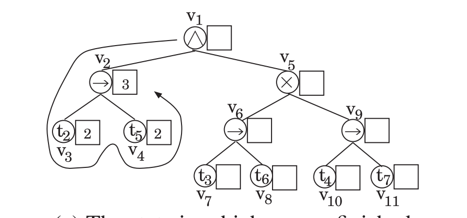
```
Input:   Process tree $\pi$ of PTB WF-net $(N, [p_I])$
Output:  State number $|R(N, [p_I])|$

CALCULATESTATENUMBERPTBWF-NET($(N, [p_I]), \pi$)
1  $v \leftarrow$ the root of $\pi$
2  CALCULATESTATENUMBER($v$)
3  Output $s(v)$ as $|R(N, [p_I])|$, and stop

CALCULATESTATENUMBER($v$)
1  if $v$ is a leaf node
2    $s(v) \leftarrow 2$
3  if $v$ is '$\rightarrow$'
4    for each child $u$ of $v$
5      CALCULATESTATENUMBER($u$)
6    $s(v) \leftarrow \sum_{\text{child } u \text{ of } v}(s(u) - 1) + 1$
7  if $v$ is '$\times$'
8    for each child $u$ of $v$
9      CALCULATESTATENUMBER($u$)
10   $s(v) \leftarrow \sum_{\text{child } u \text{ of } v}(s(u) - 2) + 2$
11 if $v$ is '$\wedge$'
12   for each child $u$ of $v$
13     CALCULATESTATENUMBER($u$)
14   $s(v) \leftarrow \prod_{\text{child } u \text{ of } v} s(u) + 2$
```

Theorem 3: The state number calculation problem can be solved in polynomial time for PTB WF-nets with initial marking $[p_I]$. □

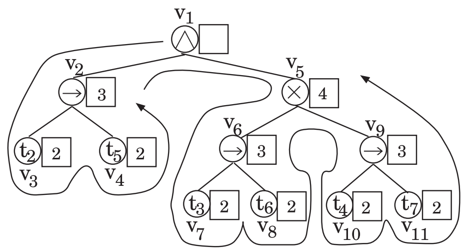
Proof: Algorithm ≪State Number Calculation of PTB WF-net≫ can run in polynomial time because it is based on DFS.
Q.E.D.

As an example, we calculate WF-net N_1 shown in

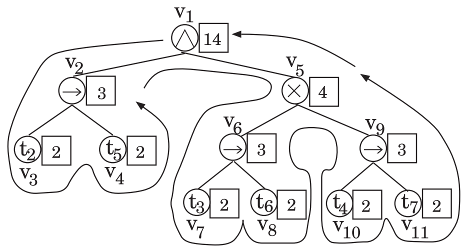
[†]A node is said to be finished if all of its children nodes have been explored.



(a) The state in which v_2 was finished



(b) The state in which v_5 was finished



(c) The state in which v_1 was finished

Fig. 7 The execution of the proposed algorithm for the process tree of N_1.

Fig. 4 (a). The process tree is $\Pi_1 = \wedge(\rightarrow(t_2, t_5), \times(\rightarrow(t_3, t_6), \rightarrow(t_4, t_7)))$ as shown in Fig. 7 (a). We apply the proposed algorithm to $(N_1, [p_1])$. Figure 7 shows the execution. For each node v, the rectangle of its right side represents $s(v)$. (a) 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 $s(v_2) = (s(v_3){-}1) + (s(v_4){-}1) + 1 = 3$. (b) The state in which v_5 was finished. From equation $s(v) = \sum_{i=1}^{n}(s(v_i) - 2) + 2$ which v is exclusive choice ($\times$), then we have $s(v_5) = (s(v_6){-}2) + (s(v_9){-}2) + 2 = 4$. (c) 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 $|R(N_1, [p_1])|$.

6. Evaluation and Application

6.1 Evaluation

We evaluate our algorithm with a tool we had developed named Process Tree Analysis Tool (ProTAT) version 2.0 (See Ref. [14]). We can convert a given WF-net to a process tree, then calculate the state number. The experiment was done on Ubuntu Linux with Intel Xeon 2.4 GHz processor and 4 GB memory. Note that calculation time also includes convertibility check time and conversion time.

We took PTB WF-nets PTB_i (i=1, 2, $\cdots$, 20) as experiment data (See Table 1). Figure 8 shows PTB_1. PTB_{i+1} was constructed by replacing a place of PTB_i with PTB_1 by refinement [15] to increase the number of parallel paths. For example, place p_7 in PTB_1 can be replaced with PTB_1 itself to produce PTB_2, then PTB_2 can be refinemented with PTB_1 again to produce PTB_3. The evaluation result is shown in Table 1. Based on ProTAT result, the calculation took about 32 seconds for PTB_{20} with over 9 million states.

Table 1 Size and computation time for PTB WF-net.

WF-net	$\|P\|$	$\|T\|$	$\|P\|+\|T\|$	State Number	Time [s]
PTB_1	8	7	15	10	0.021
PTB_2	15	14	29	28	0.041
PTB_8	57	56	113	2,296	0.526
PTB_{14}	99	98	197	147,448	6.571
PTB_{20}	141	140	281	9,437,176	32.319

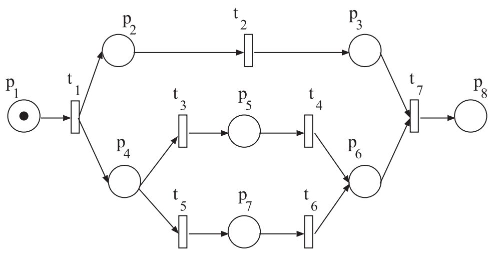


Fig. 8 PTB WF-net PTB_1.

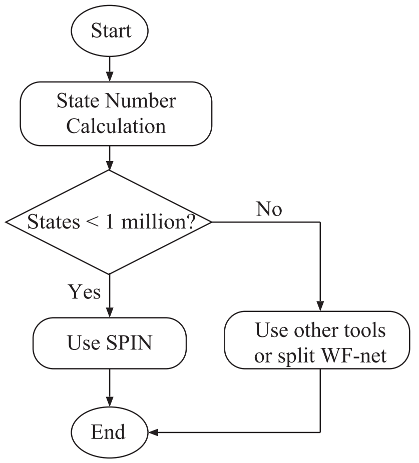


Fig. 9 Application of state number calculation.

6.2 Application

Model checking is a promising method in analysis of Petri nets. A model checking tool, SPIN has been widely used in [19] and [20]. Yamaguchi et al. [19] utilized SPIN for the verification of WF-net's soundness. Hichami et al. [20] also proposed a verification method of task execution in a process chain with SPIN.

SPIN is available to a system with less than 1 million states. Thus we apply our proposed method so that we can decide whether we should use SPIN for a given WF-net. Figure 9 shows our proposed application in model checking. Before using SPIN, we check the state number of the input

WF-net. If the state number is less than 1 million states, we can proceed to model checking with SPIN. Otherwise, we have to use other tools or split the WF-net into parts with less than 1 million states and proceed to model checking.

7. Conclusion

In this paper, we formalized the state number calculation problem. We showed that the problem is solvable and cannot be solved in polynomial time for FC WF-nets with initial marking $[p_I]$ if $P \neq NP$. Then for a given WF-net represented as a process tree, we proposed a polynomial time algorithm to solve the problem for the WF-net by utilizing the process tree.

In our future work, we will include loop operator for cyclic WF-net's state number calculation. We will also present an algorithm which utilizes state number and process tree to divide big WF-nets for parallel model checking.

Acknowledgements

The authors would like to thank Mr. Takeshi Susaki for his contribution to this work. This work was partially supported by Interface Corporation.

References

[1] T. Murata, "Petri nets: Properties, analysis and applications," Proc. IEEE, vol.77, no.4, pp.541–580, 1989.
[2] G.J. Holzmann, The SPIN Model Checker: Primer and Reference Manual, Addison-Wesley, 2004.
[3] D.Y. Chao and Y. Fang, "Number of reachable states for simple classes of Petri nets," Proc. IECON 2011, pp.3788–3791, 2011.
[4] W.M.P. van der Aalst and K.M. van Hee, Workflow Management: Models, Methods, and Systems, The MIT Press, 2002.
[5] 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, 2012.
[6] J. Esparza and M. Silva, "Circuits, handles, bridges and nets," Lecture Notes in Computer Science, vol.483, pp.210–242, 1990.
[7] W.M.P. van der Aalst, "Verification of workflow nets," Lecture Notes in Computer Science, vol.1248, pp.407–426, 1997.
[8] T. Susaki and S. Yamaguchi, "On process tree based calculation of the number of states in Petri nets," Proc. ITC-CSCC 2013, pp.84–87, 2013.
[9] J. Desel and J. Esparza, Free Choice Petri Nets, Cambridge University Press, 1995.
[10] J. Esparza and M. Nielsen, "Decidability issues for Petri nets," Bulletin of the EATCS 52, pp.244–262, 1994.
[11] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms, Second ed., pp.998–1003, The MIT Press, 2001.
[12] A. Ohta and K. Tsuji, "NP-hardness of liveness problem of bounded asymmetric choice net," IEICE Trans. Fundamentals, vol.E85-A, no.5, pp.1071–1075, May 2002.
[13] R.R. Howell, L.E. Rosier, and H. Yen, "Normal and sinkless Petri nets," J. Computer and System Sciences, vol.46, pp.1–26, 1993.
[14] M.A. Bin Ahmadon and S. Yamaguchi, "Convertibility and conversion algorithm of well-structured workflow net to process tree," Proc. CANDAR 2013, pp.122–127, 2013.
[15] K.M. van Hee, N. Sidorova, and M. Voorhoeve, "Soundness and separability of workflow nets in the stepwise refinement approach," Proc. ICATPN 2003, vol.2679, pp.337–356, 2003.
[16] 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, Feb. 2014.
[17] R.E. Tarjan, "Depth-first search and linear graph algorithms," SIAM J. Comput., vol.1, no.2, pp.146–160, 1972.
[18] S. Dohi and S. Yamaguchi, "On properties and a decision method of bridge-less workflow nets," IEICE Technical Report, MSS2013-94, 2014.
[19] S. Yamaguchi, M. Yamaguchi, and M. Tanaka, "A model checking method of soundness for acyclic workflow nets using the SPIN model checker," Int. J. INFORMATION, vol.12, no.1, pp.163–172, 2009.1.
[20] O.E. Hichami, M.A. Achhab, I. Berrada, R. Oucheikh, and B.E.E. Mohajir, "An approach of optimization and formal verification of workflow Petri nets," J. Theoretical and Applied Information Technology, vol.61, no.3, pp.486–495, 2014.

Mohd Anuaruddin Bin Ahmadon graduated from Kumamoto National College of Technology, Japan, in 2012. He received his B.E. degree from Yamaguchi University, Japan in 2014. He is currently a graduate student at Yamaguchi University, Japan. His research interest includes Petri net and its application to software engineering. He is a member of IEEE.

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 an Associate Professor in the Graduate School of Science and Engineering, Yamaguchi University, Japan. His research interests are in the area of net theory and its applications. He is a senior member of IEEE.