A Soundness Verification Tool Based on the SPIN Model Checker for Acyclic Workflow Nets

Shingo YAMAGUCHI, Munenori YAMAGUCHI, and Minoru TANAKA
Graduate School of Science and Engineering, Yamaguchi University
2-16-1 Tokiwadai, Ube, 755-8611, Japan
Tel.: +81-836-85-9510, Fax.: +81-836-85-9501
E-mail : {shingo, k040vk, tanakam}@yamaguchi-u.ac.jp

Abstract: Workflow nets (WF-nets) are Petri nets for modeling workflows, and are utilized to verification and performance evaluation of workflows. A WF-net should have a property, called *soundeness*, which guarantees a logical correctness of the modeled workflow. If a given WF-net is free choice then its soundness can be verified in polynomial time. Otherwise there is no polynomial time method to verify soundness for general WF-nets. Unfortunately, some workflows cannot be represented as free choice WF-nets. For example, the WF-net representing an inter-organizational workflow may become asymmetric choice. Thus an efficient method is required. In this paper, we propose a tool to verify soundness using the SPIN model checker. We also show efficiency of our tool by comparing it with an existing WF-nets analysis tool, *Woflan*, on verification time for asymmetric choice WF-nets.

1. Introduction

A *workflow net* (*WF-net* for short) [1] is a Petri net [2] which represents a workflow. A WF-net is intuitively said to be *sound* [3] iff, for any case, the initial state is transformed to the final state, and there are no dead transitions. Soundness is a requirement which any WF-net should meet. Thus it is very important to verify whether a given WF-net is sound. If the WF-net is free choice then its soundness can be verified in polynomial time. Otherwise there is no polynomial time method to verify soundness for classes larger than free choise. Unfortunately, some workflows cannot be represented as free choice WF-nets. For example, the WF-net representing an inter-organizational workflow may become asymmetric choice [4], which is a class larger than free choice. Therefore even if a given WF-net is not free choice, i.e. general, its soundness should be verified efficiently.

Verbeek et al.[5] have developed a tool, called *Woflan*, to analyze WF-nets. Woflan can verify soundness of general WF-nets. However the verification time may be, in the worst-case, exponential in the net size, because Woflan's algorithm generates the minimal coverability graph and the occurrence graph of the WF-net. Gannod et al.[6] have used a model checking tool, called *SPIN* [7], to verify properties of Petri nets, such as liveness and boundedness. The verification time is relatively short, because SPIN's verification algorithm is based on an optimized depth-first search. Therefore if SPIN can be applied to soundness verification, the verification would become efficient.

In this paper, we propose a tool to verify soundness of acyclic WF-nets using SPIN. We give a method to describe a given WF-net system in the modeling language of SPIN. Next we give a method to express the conditions of soundness property as Linear Temporal Logic formulas. Finally we show efficiency of our tool by comparing it with Woflan on verification time for acyclic asymmetric choice WF-nets.

2. WF-Nets and Soundness

(1) WF-Nets A WF-net is a Petri net which represents a workflow.

Definition 1 (WF-nets [1]) A Petri net $N{=}(P,T,A)$[1] is a WF-net iff (i) N has a single source place p_I, i.e. ${}^\bullet p_I{=}\phi$ and $\forall p{\in}P{-}\{p_I\}{:}\,{}^\bullet p{\neq}\phi$, and a single sink place p_O, i.e. $p_O{}^\bullet{=}\phi$ and $\forall p{\in}P{-}\{p_O\}{:}\,p^\bullet{\neq}\phi$, and (ii) every place or transition is on a path from p_I to p_O. □

Let $N{=}(P,T,A)$ be a WF-net. We represent a *marking* of N as a multi-set over P, e.g. $[p_1, 2p_2]$ denotes the marking with one token in p_1, two tokens in p_2, and no tokens in the other places. A set can also be used as a multi-set. $[p_I]$ and $[p_O]$ represent the initial state and the final state of N, respectively. A WF-net N with a marking M is called a *WF-net system*, denoted by (N,M). The Petri net obtained by connecting p_O with p_I via an additional transition t^* is called the *short-circuited net* of N, denoted by $\overline{N}$ ($=(P, T{\cup}\{t^*\}, A{\cup}\{(p_O,t^*),(t^*,p_I)\})$). The firable rule and firing rule are the same as those of Petri nets. We assume that N is fair [2]. This assumption is reasonable in the context of workflow management, because all choices are made by resources such as humans and machines. A marking M' is said to be *reachable* from a marking M if there exists a firing sequence σ transforming M to M'. This is denoted by $M[N,\sigma\rangle M'$. $M[N,*\rangle M'$ denotes $\exists\sigma : M[N,\sigma\rangle M'$.

Subclasses of WF-nets are as follows: A WF-net is free choice (FC) if $\forall p_1, p_2{\in}P{:}\ ({p_1}^\bullet{\cap}{p_2}^\bullet{\neq}\phi) \Rightarrow ({p_1}^\bullet{=}{p_2}^\bullet)$, and WF-net is asymmetric choice (AC) if $\forall p_1, p_2{\in}P{:}\ ({p_1}^\bullet{\cap}{p_2}^\bullet{\neq}\phi)$ $\Rightarrow$(${p_1}^\bullet{\subseteq}{p_2}^\bullet$ or ${p_1}^\bullet{\supseteq}{p_2}^\bullet$).

(2) Soundness Soundness is a criterion of logical correctness defined for WF-nets. A WF-net is intuitively said to be *sound* iff, for any case, the initial state is transformed to the final state, and there are no dead transitions.

Definition 2 (Soundness [1]) A WF-net N is sound iff
(i) $\forall M : ([p_I][N,*\rangle M) \Rightarrow (M[N,*\rangle[p_O])$;
(ii) $\forall M : ([p_I][N,*\rangle M \wedge M{\geq}[p_O]) \Rightarrow (M{=}[p_O])$; and
(iii) No dead transitions are in $(N,[p_I])$. □

(3) Model Checking Tool: SPIN SPIN [7] is a software tool for model checking, which has been developed by Holzmann et al.[8] SPIN can be used to verify whether a given system

[1] P, T ($\cap P{=}\phi$), and A ($\subseteq(P{\times}T){\cup}(T{\times}P)$) are sets of *places*, *transitions*, and *arcs*, respectively. Let x be a place or a transition. ${}^\bullet x$ and $x^\bullet$ are defined as ${}^\bullet x{=}\{y|(y,x){\in}A\}$ and $x^\bullet{=}\{y|(x,y)\in A\}$, respectively.

```
 1:#define Place       ⟨|P|⟩
 2:#define Transition  ⟨|T|⟩
 3:
 4:/* Variables representing a state of (N, [p_I]) */
 5:int M[Place];        /* Marking */
 6:int X[Transition];   /* Firing count */
 7:
 8:/* Process representing (N, [p_I]) */
 9:init{
10:   M[⟨p_I's ID⟩] = 1;  /* Set the initial marking */
11:   do
12:   :: atomic{ ⟨a firing of t_1⟩ }
13:   :: atomic{ ⟨a firing of t_2⟩ }
                    ⋮
14:
15:   :: atomic{ ⟨a firing of t_Transition⟩ }
16:   od
17:}
```

Figure 1. The Promela description of a WF-net system $(N(=(P,T,A)),[p_I])$.

satisfies a given property. The system to be verified is described as processes communicating with each other through channels in SPIN's modeling language, Promela (PROcess MEta LAnguage). The property to be verified is expressed as a Linear Temporal Logic (LTL) formula. An LTL formula can contain two temporal model operators: $\Box$ and $\Diamond$. $\Box\psi$ states that ψ must always hold. $\Diamond\psi$ states that ψ must eventually hold. SPIN automatically verifies whether the system satisfies the property. If the system satisfies the property, SPIN outputs "valid." Otherwise SPIN outputs "invalid" and a counter-example.

3. A Model Checking Method of Soundness

3.1 A Promela Description Method for WF-Net Systems

We first describe a given WF-net system in Promela. Holzmann et al.[8] have proposed a method to describe a Petri net system in Promela. In the method, a Petri net system is represented as a single process. The process describes each firing of its transitions. No channels are used. To verify soundness by Definition 2, a Promela description has to describe the following:

(i) Marking;
(ii) Firing count of transitions.

In Holzmann et al.'s method, only the marking is expressed into a Promela description. So we propose an improvement of Holzmann et al.'s method to describe a WF-net system including the firing count. In our method, the Promela description of a WF-net system $(N\ (=(P,T,A)),\ [p_I])$ is given as Fig. 1. Let t be a transition with x input places and y output places. $\langle$a firing of $t\rangle$ is described as follows:

```
inpx(M[⟨p^1's ID⟩], ···, M[⟨p^x's ID⟩]) ->
  X[⟨t's ID⟩]++;
  outy(M[⟨q^1's ID⟩], ···, M[⟨q^y's ID⟩])
```

where $p^1,\cdots,p^x$ are the input places of t, $q^1,\cdots,q^y$ are the output places of t. Macro `inp`x is as follows:

```
(M[⟨p^1's ID⟩]>0 && ··· && M[⟨p^x's ID⟩]>0)
   -> M[⟨p^1's ID⟩]-- && ··· && M[⟨p^x's ID⟩]--
```

`inp`x specifies the firable rule of t and the change of the marking of ${}^\bullet t$. Macro `out`y is as follows:

```
M[⟨q^1's ID⟩]++; ···; M[⟨q^y's ID⟩]++
```

`out`y specifies the change of the marking of $t^\bullet$. The Promela description of $(\overline{N},[p_I])$ is also given in the same way.

3.2 LTL Formulas to Verify Soundness

Properties to be verified in SPIN have to be expressed as LTL formulas. We rewrite Conditions (i)–(iii) of Definition 2 into LTL formulas.

Condition (i) $\forall M:([p_I][N,*\rangle M)\Rightarrow(M[N,*\rangle[p_O])$
This states that any marking reachable from $[p_I]$ eventually reaches to $[p_O]$. Therefore this condition can be rewritten as follows:

$$\Diamond(M{=}[p_O]) \text{ in } (N,[p_I]) \tag{1}$$

Condition (ii) $\forall M{:}([p_I][N,*\rangle M\wedge M{\geq}[p_O])\Rightarrow(M{=}[p_O])$
This states that if a marking reachable from $[p_I]$ has tokens in p_O then the marking is always $[p_O]$. Therefore this condition can be rewritten as follows:

$$\Box((M(p_O){\geq}1)\Rightarrow(M{=}[p_O])) \text{ in } (N,[p_I]) \tag{2}$$

Condition (iii) No dead transitions are in $(N,[p_I])$
It is known from Theorem 4.8 in Ref. [9] that if N satisfies Condition (ii), a transition t is dead in $(N,[p_I])$ iff t is dead in $(\overline{N},[p_I])$. It is obvious that no dead transitions are in $(\overline{N},[p_I])$ iff every transition eventually fires in $(\overline{N},[p_I])$ on the assumption of fairness. Therefore this condition can be rewritten as follows:

$$\begin{array}{c}\Diamond(\forall t{\in}T : X(t){>}0) \text{ in } (\overline{N},[p_I])\\ \text{on the fairness assumption}\end{array} \tag{3}$$

where $X(t)$ is a firing count of t. This formula is accompanied with the fairness assumption, but fairness is not considered by SPIN. SPIN's verification algorithm searches a counter-example for this formula without taking account of fairness. To take accout of fairness, we use the converse of this formula, i.e.

$$\Box(\exists t{\in}T : X(t){=}0) \text{ (in } (\overline{N},[p_I])) \tag{4}$$

Note that this formula becomes *false* iff there are no dead transitions in $(N,[p_I])$. To search a counter-example for this formula, SPIN's verification algorithm chooses transitions having not been fired yet. As a result, fairness holds. Note that SPIN outputs "invalid" iff there are no dead transitions in $(N,[p_I])$. From Eqs. (1), (2) and (4), we can obtain the following theorem:

Theorem 1: A WF-net N is sound iff
(i) $\Diamond(M{=}[p_O])$ in $(N,[p_I])$;
(ii) $\Box((M(p_O){\geq}1)\Rightarrow(M{=}[p_O]))$ in $(N,[p_I])$; and
(iii) $\Box(\exists t{\in}T : X(t){=}0)$ (in $(\overline{N},[p_I])$). $\Box$

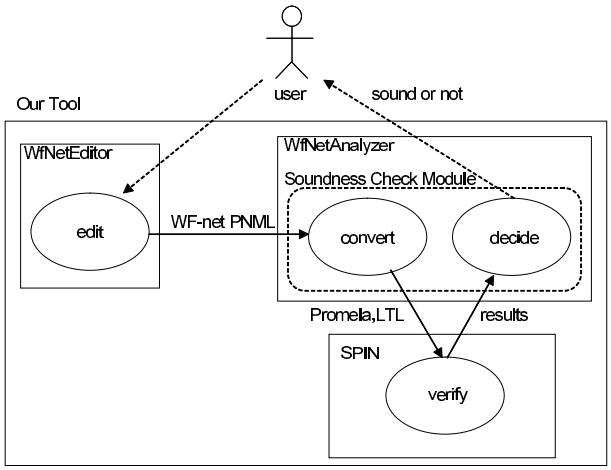


Figure 2. Configuration of our tool.

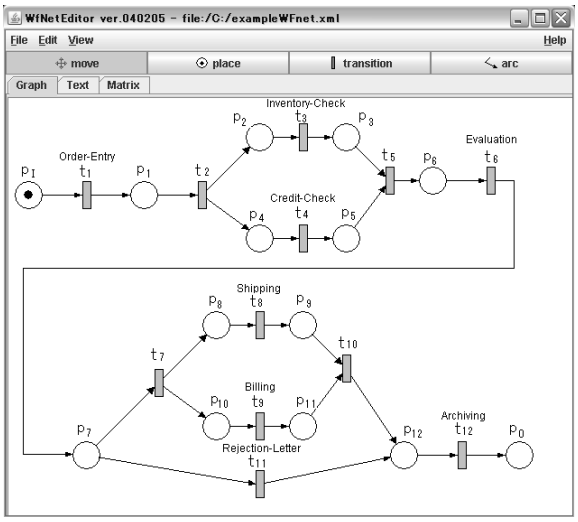


Figure 3. Screenshot of WfNetEditor.

4. Verification Tool and Evaluation

4.1 Verification Tool

We developed a tool to verify soundness based on the proposed method. The configuration of our tool is shown in Fig. 2. Our tool consists of *WfNetEditor* and *WfNetAnalyzer*, which has also been developed by us. The details of them are as follows.

(1) WfNetEditor WfNetEditor is a graphical editor that generates a WF-net as a PNML (Petri Net Markup Language) [10] format file. Figure 3 shows a screenshot of WfNetEditor.

(2) WfNetAnalyzer WfNetAnalyzer is an pluggable analyzer for WF-nets described in PNML. Figure 4 shows a screenshot of WfNetAnalyzer. WfNetAnalyzer can employ plug-in modules in order to analyze various properties of WF-nets. Our model checking method of soundness has been also developed as a plug-in module.

(3) Soundness Check Module The soundness check module converts the PNML file to a Promela description and LTL formulas, and invokes SPIN by using them as parameters.

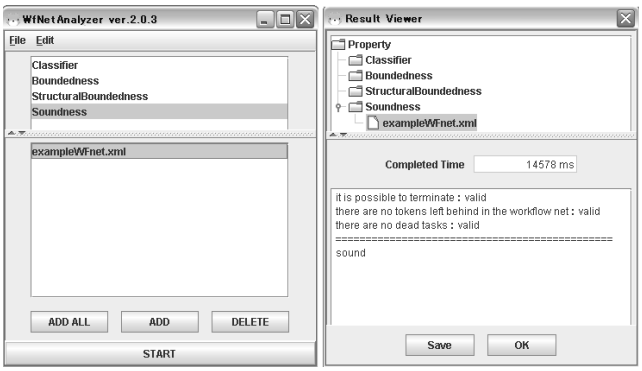


Figure 4. Screenshot of WfNetAnalyzer.

```
#define Place 14
#define Transition 12
#define inp1(x1) (x1>0) -> x1--
#define inp2(x1,x2) (x1>0 && x2>0) -> x1--;x2--
#define out1(x1) x1++
#define out2(x1,x2) x1++; x2++
int M[Place];
int X[Transition];
init
{
  M[0]=1;
  do
  :: atomic{inp1(M[0]) -> X[0]++;out1(M[1])}
  :: atomic{inp1(M[1]) -> X[1]++;out2(M[2],M[4])}
  :: atomic{inp1(M[2]) -> X[2]++;out1(M[3])}
  :: atomic{inp1(M[4]) -> X[3]++;out1(M[5])}
  :: atomic{inp2(M[3],M[5]) -> X[4]++;out1(M[6])}
  :: atomic{inp1(M[6]) -> X[5]++;out1(M[7])}
  :: atomic{inp1(M[7]) -> X[6]++;out2(M[8],M[10])}
  :: atomic{inp1(M[8]) -> X[7]++;out1(M[9])}
  :: atomic{inp1(M[10]) -> X[8]++;out1(M[11])}
  :: atomic{inp2(M[9],M[11]) -> X[9]++;out1(M[12])}
  :: atomic{inp1(M[7]) -> X[10]++;out1(M[12])}
  :: atomic{inp1(M[12]) -> X[11]++;out1(M[13])}
  od
}
```

Figure 5. The Promela description of $(N_1, [p_I])$ shown in Fig. 3.

This module provides SPIN with the LTL formulas shown in Theorem 1 as follows: Formula (i) $\Diamond(M{=}[p_O])$ in $(N, [p_I])$ is rewritten as

$$\texttt{<>M[}\langle p_O\text{'s ID}\rangle\texttt{]==1 (in } (N, [p_I])\text{).}$$

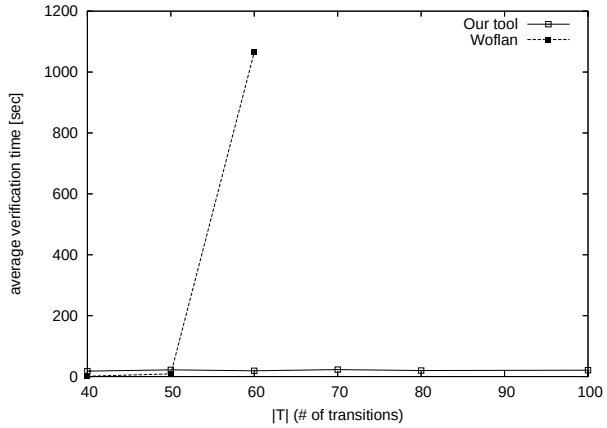
Formula (ii) $\Box((M(p_O){\geq}1){\Rightarrow}(M{=}[p_O]))$ in $(N, [p_I])$ is rewritten as

$$\texttt{[] ((M[}\langle p_O\text{'s ID}\rangle\texttt{]>=1) ->}$$
$$\texttt{(M[}\langle p_I\text{'s ID}\rangle\texttt{]==0 \&\& M[}\langle p_1\text{'s ID}\rangle\texttt{]==0 \&\}\cdots$$
$$\texttt{\&\& M[}\langle p_{(|P|-2)}\text{'s ID}\rangle\texttt{]==0 \&\& M[}\langle p_O\text{'s ID}\rangle\texttt{]==1))}$$
$$\text{(in } (N, [p_I])\text{).}$$

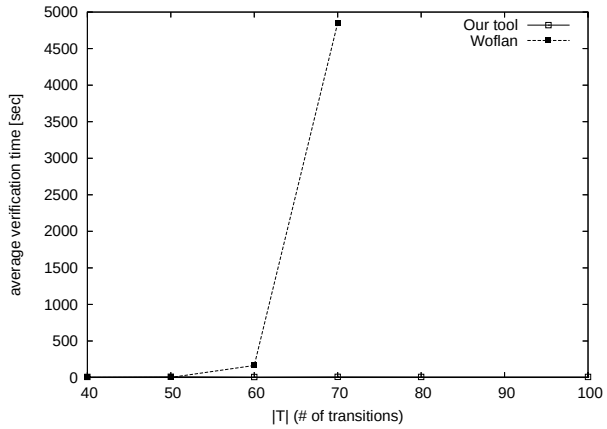
Formula (iii) $\Box(\exists t{\in}T : X(t){=}0)$ (in $(\overline{N}, [p_I])$) is rewritten as

$$\texttt{[] (X[}\langle t_1\text{'s ID}\rangle\texttt{]==0 || X[}\langle t_2\text{'s ID}\rangle\texttt{]==0 ||}\cdots$$
$$\texttt{|| X[}\langle t_{|T|}\text{'s ID}\rangle\texttt{]==0)} \quad \text{(in } (\overline{N}, [p_I])\text{).}$$

The soundness check module receives results of the LTL formulas from SPIN, and decides whether a given WF-net is sound based on the results.



(a) Sound WF-nets



(b) Unsound WF-nets

Figure 6. Experimental results. Note that Woflan couldn't verify the soundness of some WF-nets of $|T|\geq 70$ within 10 hours.

(4) Example We illustrate our tool with an example. We first generate a WF-net N_1 with WfNetEditor. WF-net N_1 is shown in Fig. 3. Then we input WF-net N_1 to WfNetAnalyzer. The soundness check module converts WF-net N_1 to the Promela description shown in Fig. 5 and the following LTL formulas.

(i) `<>(M[13]==1)`

(ii) `[]((M[13]>=1) -> (M[0]==0 && M[1]==0 && ··· && M[12]==0 && M[13]==1))`

(iii) `[](X[0]==0||X[1]==0||··· ||X[11]==0)`

SPIN returns "valid" for LTL formulas (i) and (ii), and "invalid" for LTL formula (iii). The soundness check module decides based on the results that N_1 is sound.

4.2 Evaluation

We compare our method with an existing WF-nets analysis tool, Woflan, on verification time for acyclic ACWF-nets.

(1) Experimental Data As experimental data, we generated 6 groups of 10 sound nets and 10 unsound nets, i.e, 120 ($=6\times(10+10)$) nets. The first, second, $\cdots$, fifth, and sixth group have 40, 50, $\cdots$, 80, and 100 transitions, respectively. It is known that in general, the WF-net representing an actual workflow has at most 100 transitions. Therefore we can say that our experimental data are appropriate from the viewpoint of size.

(2) Experimental results The experimental results for the sound WF-nets and the unsound WF-nets are illustrated in Fig. 6 (a) and (b), respectively. The horizontal axis shows the number of transitions, and the vertical axis shows the average verification time. Those graphs show that the average verification time of Woflan increases suddenly with an increase of the number of transitions, while that of our method increases little. Thus we can say that our method is more efficient than Woflan. For more information on experimental results, the reader is referred to [11].

(3) Consideration Let us consider the reason the average verification time of Woflan increases suddenly with an increase of the number of transitions. Soundness of a WF-net N coincides with liveness and boundedness of $\overline{N}$. Woflan's verification algorithm is based on the property. The algorithm uses the minimal coverability graph of a given WF-net to verify boundedness, and uses the occurrence graph to verify liveness. It is known that the construction time of those graphs may be, in the worst-case, exponential in the size of the WF-net. This is considered to be the reason the average verification time of Woflan increases suddenly with an increase of the number of transitions.

5. Conclusion

In this paper, we have proposed a tool to verify soundness of acyclic WF-nets using SPIN. We have given a method to describe a given WF-net system in Promela. Next we have given a method to express the conditions of soundness property as LTL formulas. Finally we have compared our tool with Woflan on verification time for 120 acyclic ACWF-nets. For the WF-nets of $|T|\geq 60$, the average verification time of our method was shorter than that of Woflan. From the results, we can say that our tool is more efficient than Woflan if the size of a given WF-net is larger. As a future work, we extend our tool for cyclic WF-nets.

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] H.M.W. Verbeek, T. Basten, and W.M.P. van der Aalst, "Diagnosing workflow processes using Woflan," The Computer Journal, vol.44, no.4, pp.246-279, 2001.
[4] S. Yamaguchi, H. Matsuo, Q.W. Ge, and M. Tanaka, "WF-net based modeling and soundness verification of interworkflows," IEICE Trans. Fundamentals, vol.E90-A, no.4, pp.829-835, 2007.
[5] Woflan, `http://is.tm.tue.nl/research/woflan.htm`
[6] G.C. Gannod and S. Gupta, "An automated tool for analyzing Petri nets using SPIN," Proc. of IEEE International Conference on Automated Software Engineering (ASE'01), pp.404–407, 2001.
[7] SPIN model checker, `http://spinroot.com/`
[8] G.J. Holzmann, The Model Checker Spin, Addison-Wesley, 2003.
[9] H.M.W. Verbeek, Verification of WF-nets, Ph.D. Thesis, Technische Universiteit Eindhoven, 2004.
[10] Petri Net Markup Language (PNML) `http://www2.informatik.hu-berlin.de/top/pnml`
[11] S. Yamaguchi, M. Yamaguchi, and M. Tanaka, "On soundness verification of acyclic workflow nets using the SPIN model checker," Proc. of 21th Karuizawa Workshop on Circuits and Systems, pp.255-260, 2008.