Research Article

The Equivalency between Logic Petri Workflow Nets and Workflow Nets

Jing Wang, ShuXia Yu, and YuYue Du

Shandong University of Science and Technology, Qingdao, China

Correspondence should be addressed to Jing Wang; jing_wang@live.cn

Received 30 July 2014; Revised 25 September 2014; Accepted 7 October 2014

Academic Editor: Jung-Fa Tsai

Copyright © 2015 Jing Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Logic Petri nets (LPNs) can describe and analyze batch processing functions and passing value indeterminacy in cooperative systems. Logic Petri workflow nets (LPWNs) are proposed based on LPNs in this paper. Process mining is regarded as an important bridge between modeling and analysis of data mining and business process. Workflow nets (WF-nets) are the extension to Petri nets (PNs), and have successfully been used to process mining. Some shortcomings cannot be avoided in process mining, such as duplicate tasks, invisible tasks, and the noise of logs. The online shop in electronic commerce in this paper is modeled to prove the equivalence between LPWNs and WF-nets, and advantages of LPWNs are presented.

1. Introduction

Petri nets (PNs) [1] are a process modeling technique applied to the simulation and analysis of distributed systems, and PN nets are also an effective description and analysis tool for many fields. With the continuous development of PN theory and the increasing popularity of its application, some of their extensions have been defined, such as colored [2], time [3], fuzzy [4], and stochastic PNs [5]. Logic Petri nets [6–8] are the abstract and extension of high-level PNs and have been applied efficiently to the modeling and analysis of Web services, cooperative systems, and electronic commerce. Transitions restricted by logic expressions are called logic transitions. The inputs and outputs can be described by logic transitions in LPNs. Based on LPNs, the definition of LPWNs is proposed in this paper. An LPWN is logic Petri net with a dedicated source place where the process starts and a dedicated sink place where the process ends. Moreover, all nodes are on at least a path from source to sink.

Larger online shops produce a great quantity of transaction records every day. How to find valuable information in these records is a meaningful task. These records are called event logs in process mining, which are the starting point of process mining [9, 10]. When modeling business processes in terms of Petri nets, a subclass of Petri nets known as Workflow nets is considered [11–14]. WF-nets are also a natural representation for process mining. Process mining [15, 16] is a young cross field and crosses the computational intelligence and data mining field to the modeling process and analysis area. Process mining is regarded as an important bridge between modeling and analysis of data mining and business process [17–19]. LPWNs and WF-nets are evolutions of PNs. The LPWN will be introduced into process mining in our later work, so the equivalency between LPWNs and WF-nets is firstly proved by an online shop model in this paper. Compared with WF-nets, LPWNs can well describe and analyze batch processing functions and passing value indeterminacy in cooperative systems and effectively alleviate the state space explosion problem to an extent.

The rest of this paper is organized as follows. Section 2 reviews definitions of PNs, WF-nets, and LPNs, and the standard forms of logic expressions and LPWNs are put forward. A simple LPWN model is given to explain how the LPWN works. In order to prove the equivalence between LPWNs and WF-nets, isomorphism and equivalent definitions are proposed in Section 3. Theorem 8 has been proved on the basis of isomorphism and equivalent definitions, and the constructing algorithm of an equivalent WF-net from an LPWN is presented. In Section 4, Theorem 8 and the algorithm are
illustrated by an online shop model. Concluding remarks are made in Section 4.

2. Logic Petri Workflow Nets

This section introduces some basic definitions about PNs, LPNs, and WF-nets.

**Definition 1** (see [8]). PN = \((P, T; F, M)\) is a marked PN, where

1. \(N = (P, T; F)\) is a net;
2. \(M : P \rightarrow \mathbb{N}\) is a marking function, where \(M_0\) is the initial marking and \(\mathbb{N} \rightarrow \{0, 1, 2, \ldots\}\);
3. transition firing rules are as follows:
   - (a) \(t\) is enabled at \(M\) if for all \(p \in \mathcal{R}t\) : \(M(p) = 1\), represented by \(M[t >]\);
   - (b) if \(t\) is enabled, it can fire, and a new marking \(M'\) is generated from \(M\), represented by \(M[t > M']\), where
     \[
     M'(p) = \begin{cases} 
     M(p) + 1 & \text{if } p \in t^\prime - t \\
     M(p) - 1 & \text{if } p \in t - t^\prime \\
     M(p) & \text{else}
     \end{cases}
     \] (1)

**Definition 2** (see [8]). Let PN = \((P, T; F, M_0)\) be a Petri net and \(t^*\) a fresh identifier not in \(P \cup T\). The PN is a workflow net (WF-net) if and only if

(a) \(P\) contains an input place \(i\) (also called source place) such that \(i = \varnothing\);
(b) \(P\) contains an output place \(o\) (also called sink place) such that \(o = \varnothing\);
(c) \(PN^* = (P, T \cup \{t^*\}, F \cup ((o, t^*), (t^*, i)))\) is strongly connected.

There is a directed path between any pair of nodes in PN.

**Definition 3** (see [8]). LPN = \((P, T; F, I, O, M)\) is a logic Petri net where

1. \(P\) is a finite set of places;
2. \(T = T_D \cup T_I \cup T_O\) is a finite set of transitions, \(P \cup T \neq \varnothing\), \(P \cap T = \varnothing\), for all \(t \in T_I \cup T_D\); \(t \cap t' = \varnothing\), where
   - (a) \(T_D\) denotes a set of traditional transitions;
   - (b) \(T_I\) denotes a set of logic input transitions, where, for all \(t \in T_I\), the input places of \(t\) are restricted by a logic input expression \(f_i(t)\);
   - (c) \(T_O\) denotes a set of logic output transitions, where, for all \(t \in T_O\), the output places of \(t\) are restricted by a logic output expression \(f_o(t)\);
3. \(F \subseteq (P \times T) \cup (T \times P)\) is a finite set of directed arcs;
4. \(I\) is a mapping from a logic input transition to a logic input expression; that is
   \[
   \forall t \in T_I, I(t) = f_i(t) = A_1 \lor A_2 \lor \cdots \lor A_m; \quad (2)
   \]
5. \(O\) is a mapping from a logic output transition to a logic input expression; that is
   \[
   \forall t \in T_O, O(t) = f_o(t) = B_1 \lor B_2 \lor \cdots \lor B_n; \quad (3)
   \]
6. \(M : P \rightarrow \{0, 1\}\) is a marking function, where, for all \(p \in P\), \(M(p)\) is the number of tokens in \(p\);
7. Transition firing rules are as follows:
   - (a) for all \(t \in T_D\), the firing rules of \(t\) are the same as in PNs;
   - (b) for all \(t \in T_I\), \(t\) is enabled only if \(\exists A_i; \) make \(f_i(t)\mid_M = T_i, M[1 > M']\), where, for all \(p \in t\) and \(p \in A_i\), \(M'(p) = M(p) - 1\); for all \(p \notin t\) and \(p \notin A_i\), \(M'(p) = M(p)\); for all \(p \in t'\), \(M'(p) = M(p) + 1\); and, for all \(p \notin t \cup t'\), \(M'(p) = M(p)\);
   - (c) for all \(t \in T_O\), \(t\) is enabled only if for all \(p \in t\) : \(M(p) = 1, M[1 > M']\), where, for all \(p \notin t\) : \(M'(p) = M(p) - 1\); for all \(p \notin t'\) : \(M'(p) = M(p)\); for all \(p \notin t'\) and \(p \notin B_i\), should satisfy \(f_o(t)\mid_{M'} = T_o\) and for all \(p \in t'\) and \(p \notin B_i\), \(M'(p) = M(p)\).

LPNs are the abstract and extension of IPNs and high-level PNs. In Definition 3, a logic input/output transition is restricted by the logic input/output expression \(f_i(t) / f_o(t)\) in LPNs. All logic input/output transitions are called logic transitions. The logic expressions can describe the indeterminacy of values in input and output places. \(A_i\) and \(B_i\) represent input and output ways of logic transitions, respectively. They are not the disjunctive normal of \(f_i(t) / f_o(t)\).

**Definition 4**. Suppose that a logic input/output transition \(t\) is restricted by \(f_i(t) / f_o(t)\), and the standard form is as follows.

For a logic input transition \(t\), the standard form of \(f_i(t) = A_1 \lor A_2 \lor \cdots \lor A_m\) can be obtained by
\[
A_i = \begin{cases} 
A_i, & \text{if } |A_i| = \lfloor t \rfloor \\
A_i \land \neg p, & \text{if } |A_i| \neq \lfloor t \rfloor, \exists p \in t.
\end{cases} \quad (4)
\]

For a logic input transition \(t\), the standard form of \(f_o(t) = B_1 \lor B_2 \lor \cdots \lor B_n\) can be obtained by
\[
B_i = \begin{cases} 
B_i, & \text{if } |B_i| = \lceil t \rceil \\
B_i \land \neg p, & \text{if } |B_i| \neq \lceil t \rceil, \exists p \in t.
\end{cases} \quad (5)
\]

This definition puts forward the standard form of logic expression. \(A_i\) and \(B_i\) are called the standard minterms.

**Definition 5**. Let LPN = \((P, T; F, I, O, M)\) be a logic Petri net, and the LPN is a logic Petri workflow net (LPWN) if and only if

(a) LPN has \(P = P_C \cup P_D\), where \(P_C/P_D\) are control/data place sets;
(b) there is a source place \(i \in P_C\) such that \(i = \varnothing\); there is a sink place \(o \in P_C\) such that \(o = \varnothing\);
3. Transforming an LPWN into an Equivalent WF-Net

This section puts forward isomorphism and equivalent definitions to prove the equivalence between LPWNs and WF-nets.

**Definition 6.** Let $\Sigma_1 = (P, T; F, I, O, M_0)$ be an LPWN and $\Sigma_2 = (P', T'; F', I', O', M_0')$ a WF-net. $RG(\Sigma_1)$ is the reachable tree of $\Sigma_1$, and $R(\Sigma_1)$ is the node set of $RG(\Sigma_1)$, $i = 1, 2$. If there exists a bijection function $f : R(\Sigma_1) \rightarrow R(\Sigma_2)$, such that, for all $M_1, M_2 \in R(\Sigma_1)$, $t \in T$, $M_1[t > M_2] \Rightarrow \exists ! t' \in T'$, $f(M_1)[t'] > f(M_2)$. Then, $RG(\Sigma_1)$ and $RG(\Sigma_2)$ are isomorphic.

**Definition 7.** Let $\Sigma_1 = (P, T; F, I, O, M_0)$ be an LPWN and $\Sigma_2 = (P', T'; F', I', M_0')$ a WF-net. $\Sigma_1$ and $\Sigma_2$ are equivalent if and only if $RG(\Sigma_1)$ and $RG(\Sigma_2)$ are isomorphic.

Based on Definitions 6 and 7, a theorem is given.

**Theorem 8.** For any LPWN, there exists an equivalent WF-net.

**Proof.** Consider the following.

**Step 1.** Constructing an equivalent WF-net is as follows.

Let $\Sigma_1 = (P, T; F, I, O, M_0)$ be an LPWN, and the deterministic WF-net $\Sigma_2 = (P', T'; F', M_0')$ being equivalent to $\Sigma_1$ should be constructed at the very start.

For all $t \in T$, there are three conditions to transform a transition of $\Sigma_1$ into one or more corresponding transitions in $\Sigma_2$.

**Step 1.1.** For $t_i \in T_P$, let $t_i \in T'$; for all $p \in P$, if $(p, t_i) \in F$, then $(p, t_i) \in F'$; and if $(t_i, p) \in F$, then $(t_i, p) \in F'$.

**Step 1.2.** For $t_i \in T_I$, let $t_i = \{p_{i1}, p_{i2}, \ldots, p_{ik}\}$; $f_j(t_i) = A_{i1} \lor A_{i2} \lor \cdots \lor A_{ik}$; $t_i$ is restricted by the standard logic input expression $f_j(t_i)$. There are $m$ standard minterms of $f_j(t_i)$, and each minterm corresponds to a transition of $\Sigma_2$.

**Step 1.3.** For any standard minterm $A_{ij}$, where $j \in \{1, 2, \ldots, m\}$, assume that $A_{ij}$ corresponds to the transition $t_{ij}$ in $\Sigma_2$; that is, $t_{ij} \in T'$. Then, the arc set related to $t_{ij}$ is defined. For all $p_k \in t_{ij}$, where $k \in \{1, 2, \ldots, n\}$, if $p_k \in A_{ij}$, we have $(t_{ij}, p_k) \in F'$; for all $p \in t_{ij}$, we have $(t_{ij}, p) \in F'$, where $i \in \{1, 2, \ldots, n\}$.

**Step 2.** Proof that the constructing WF-net $\Sigma_2$ is equal to $\Sigma_1$.

Based on Step 1, the place set $P$ and the initial marking $M_0$ in $\Sigma_1$ are the same as those in $\Sigma_2$; that is, $P = P'$, $M_0 = M_0'$, but the transition set $T$ and the flow set $F$ are not; that is, $T \neq T'$, $F \neq F'$, and $|T| \leq |T'|$, $|F| \leq |F'|$, where $|T|$ denotes the size of set $T$. Firing a transition of $\Sigma_2$ corresponds to firing a transition of $\Sigma_1$; that is, if a transition is enabled in $\Sigma_1$, then there must be an enabled transition in $\Sigma_2$ and it is unique. Since $\Sigma_1$ and $\Sigma_2$ have the same initial marking, the equivalence between $\Sigma_1$ and $\Sigma_2$ is proved on the basis of the reachable marking graph.

In $\Sigma_1$, for all $M_1, M_2 \in R(\Sigma_1)$, $t_i \in T$; if $M_1[t_i > M_2$, then there is a mapping function $f : R(\Sigma_1) \rightarrow R(\Sigma_2)$ based on Step 1; we have $f(M_1) = M_1$ and $f(M_2) = M_2$ in $\Sigma_2$. 

![Figure 1: An LPWN model $N_1$.](image-url)
if \( t_1 \in T_D \), then \( \exists t'_1 \in T': t'_1 = t_1 \) and \( f(M_1)[t'_1] > f(M_2) \); if \( t \in T_I \lor T_O \), then \( \exists t'_i \in T' \); we have \( f(M_1)[t'_i] > f(M_2) \). \( f \) is an identity mapping and satisfies injective and surjection requirements at \( M \in R(M_0) \). That is, \( \Sigma_1 \) and \( \Sigma_2 \) have the same behavior characteristics. Moreover, the structure of \( \Sigma_2 \) is unique since its standard form is only one. So \( f \) is a bijective function, and \( RG(\Sigma_1) \) and \( RG(\Sigma_2) \) are isomorphic. Based on Definition 7, \( \Sigma_1 \) and \( \Sigma_2 \) are equivalent. 

Based on Theorem 8 and the construction of \( \Sigma_2 \), the construction algorithm of an equivalent WF-net from an LPWN can be obtained.

In Algorithm 1, the equivalent WF-net has the same place set and traditional transitions compared with its corresponding LPWN. Their differences are the logic transitions and flows. Next, an example is used to prove the correctness and appropriateness of Theorem 8 and Algorithm 1.

4. A Case

In this section, the work processes of an online shop in electronic commerce shown in Figure 2 are modeled by the LPWN, and the validity and usefulness of the presented method are illustrated based on the analysis of the model. Functions of the online shop are modeled by transitions. For example, the transition receive_order represents that the shop owner will get an order from the client, and it is limited by the logic expression \( f_1(\text{receive\_order}) \). Based on Definition 4, all logic transitions and their standard items are shown in Table 1.

Next, the LPWN \( N_1 \) shown in Figure 2 will be transformed into its equivalent WF-net.

In Figure 2, the logic input transition receive_order can be transformed into three traditional transitions as follows.

The receive_order is a logic input transition restricted by \( f_1(\text{receive\_order}) = A_{11} \lor A_{12} \lor A_{13} \), where \( A_{11} = p_1 \land \text{order\_1} \land \neg \text{order\_2} \), \( A_{12} = p_1 \land \text{order\_1} \land \text{order\_2} \), and \( A_{13} = p_1 \land \neg \text{order\_1} \land \text{order\_2} \). The receive_order has three ways to transform tokens. For example, \( (p_1 \land \text{order\_1} \land \neg \text{order\_2}) \) represents \( p_1 \) and \( \text{order\_1} \) loses a token and \( \text{order\_2} \) does not lose a token after the receive_order fires. From Algorithm 1, in the equivalent WF-net, the transition receive_order can be transformed into \( r_{o1}, r_{o2}, \) and \( r_{o3} \), and they are three traditional transitions. Flows \((p_1, \text{receive\_order}), (\text{order\_1}, \text{receive\_order}), \) and \((\text{order\_2}, \text{receive\_order})\) are transformed into seven flows \((p_1, r_{o1}), (p_1, r_{o2}), (p_1, r_{o3}), (\text{order\_1}, r_{o1}), (\text{order\_1}, r_{o3}), (\text{order\_2}, r_{o2}), \) and \((\text{order\_2}, r_{o3}) \). The flow \((\text{receive\_order}, p_2)\) is transformed into three flows \((r_{o1}, p_2), (r_{o2}, p_2), \) and \((r_{o3}, p_2) \). The input transition receive_payment can also be transformed by this method.

In Figure 2, the logic input transition send_to_expres can be transformed into traditional transitions shown in Figure 3 as follows.

The send_to_express is a logic output transition restricted by \( f_2(\text{send\_to\_express}) = B_{11} \lor B_{12} \lor B_{13} \), where \( B_{11} = p_3 \land \neg p_4 \), \( B_{12} = p_3 \land p_4 \) and \( B_{13} = \neg p_3 \land p_4 \). The send_to_expres has three ways to transform tokens. For example, \( (p_3 \land \neg p_4) \) represents that \( p_3 \) gets a token and \( p_4 \) does not get a token after the send_to_express fires. From Algorithm 1, in the equivalent WF-net, the logic output transition send_to_express can be transformed into \( stel, ste2, \) and \( ste3 \), and they are three traditional transitions. Flows \((\text{send\_to\_express}, p_3), (\text{send\_to\_express}, p_4)\) are transformed into four flows \((\text{ste1}, p_3), (\text{ste3}, p_3), (\text{ste3}, p_4), \) and \((\text{ste2}, p_4) \). The flow \((p_2, \text{send\_to\_express})\) is transformed into three flows \((p_2, \text{ste1}), (p_2, \text{ste2}), \) and \((p_2, \text{ste3}) \). Other output transitions confirm_refuses, confirm_goods, and send_money can be transformed by this method.

In Figure 2, \( t_1, t_2, \) and \( t_3 \) are three traditional transitions, and places, transitions, and flows related to them do not change. Based on the above method, the equivalent WF-net can be obtained in Figure 3.

From Figures 2 and 3, the WF-net consists of 21 transitions and 58 flows while its equivalent LPWN model has 9 transitions and 30 flows, and the number of their places is the same. The rates of transitions and flows descending from its
| Transitions                  | Logic expressions | Standard items |
|-----------------------------|-------------------|----------------|
| receive order               | \( f_r(\text{receive\_order}) = A_{11} \lor A_{12} \lor A_{13} \) | \( A_{11} = p_1 \land \text{order\_1} \land \neg \text{order\_2} \) | \( A_{12} = p_1 \land \text{order\_1} \land \text{order\_2} \) | \( A_{13} = p_1 \land \neg \text{order\_1} \land \text{order\_2} \) |
| send to express             | \( f_c(\text{send\_to\_express}) = B_{11} \lor B_{12} \lor B_{13} \) | \( B_{11} = p_1 \land \neg p_4 \) | \( B_{12} = p_1 \land p_4 \) | \( B_{13} = \neg p_1 \land p_4 \) |
| confirm goods               | \( f_c(\text{confirm\_goods}) = B_{11} \lor B_{12} \lor B_{13} \) | \( B_{11} = p_1 \land \text{good\_1} \land \neg \text{good\_2} \) | \( B_{12} = p_1 \land \text{good\_1} \land \text{good\_2} \) | \( B_{13} = \neg p_1 \land \text{good\_1} \land \text{good\_2} \) |
| confirm refueses            | \( f_c(\text{confirm\_refuses}) = B_{11} \lor B_{12} \lor B_{13} \) | \( B_{11} = p_1 \land \text{refuse\_1} \land \neg \text{refuse\_2} \) | \( B_{12} = p_1 \land \text{refuse\_1} \land \text{refuse\_2} \) | \( B_{13} = \neg p_1 \land \text{refuse\_1} \land \text{refuse\_2} \) |
| send, money                 | \( f_c(\text{send\_money}) = B_{11} \lor B_{12} \lor B_{13} \) | \( B_{11} = p_1 \land \text{money\_1} \land \neg \text{money\_2} \) | \( B_{12} = p_1 \land \text{money\_1} \land \text{money\_2} \) | \( B_{13} = p_1 \land \neg \text{money\_1} \land \text{money\_2} \) |
| receive, payment            | \( f_r(\text{receive\_payment}) = A_{21} \lor A_{22} \lor A_{23} \) | \( A_{21} = p_1 \land \text{pay\_1} \land \neg \text{pay\_2} \) | \( A_{22} = p_1 \land \text{pay\_1} \land \text{pay\_2} \) | \( A_{12} = p_1 \land \neg \text{pay\_1} \land \text{pay\_2} \) |
Input: an LPWN $\Sigma_1 = (P, T; F, I, O, M_0)$
Output: an equivalent WF-net $\Sigma_2 = (P', T'; F', M')$

1. $WF$-net.$P' = LPWN.P$;
2. $WF$-net.$M' = LPWN.M_0$
3. For each transition $t_i$ in LPWN.$T_P$
4. $WF$-net.$T_i' = WF$-net.$T_i$ ∪ $\{t_i\}$
5. For each $p$ in $t_i$
6. $WF$-net.$F_i' = WF$-net.$F_i$ ∪ $\{(p, t_i)\}$
7. End for
8. For each $p$ in $t_i^*$
9. $WF$-net.$F_i' = WF$-net.$F_i$ ∪ $\{(t_i, p)\}$
10. End for
11. For each $t_i$ in LPWN.$T_O$
12. For each $A_{ij}$ in the standard form $\bigvee_{j=1}^{m} A_{ij}$ of $f_i(t_i)$
13. $WF$-net.$T_i' = WF$-net.$T_i$ ∪ $\{t_{i1}, \ldots, t_{im}\}$
14. For each $p$ in $A_{ij}$
15. If $p = T_i$ then $WF$-net.$F_i' = WF$-net.$F_i$ ∪ $\{(p, t_{ij})\}$
16. End for
17. End for
18. $WF$-net.$F_i' = WF$-net.$F_i$ ∪ $\{(t_i, p)\}$
19. End for
20. End for
21. For each $t_i$ in LPWN.$T_R$
22. For each $B_{ij}$ in the standard form $\bigvee_{j=1}^{n} B_{ij}$ of $f_i(t_i)$
23. $WF$-net.$T_i' = WF$-net.$T_i$ ∪ $\{t_{i1}, \ldots, t_{im}\}$
24. For each $p$ in $B_{ij}$
25. If $p = T_i$ then $WF$-net.$F_i' = WF$-net.$F_i$ ∪ $\{(p, t_{ij})\}$
26. End for
27. End for
28. $WF$-net.$F_i' = WF$-net.$F_i$ ∪ $\{(p, t_i)\}$
29. End for
30. End for
31. End for
32. End for

**Algorithm 1:** Transforming an LPWN into an equivalent WF-net.

**Figure 3:** The LPWN $N_1$ is transformed into its equivalent WF-net.
The LPWN will be applied efficiently to progress mining. Paper, such as state equivalency, liveness, and reachability. Proposed algorithm have been exemplified by the online shop. Algorithm 1 used to construct an equivalent WF-net from an LPWN is put forward. Effectiveness and practicality of the proposed algorithm have been exemplified by the online shop model.

In further work, the fundamental properties of LPWNs will be investigated according to the results proposed in this paper, such as state equivalency, liveness, and reachability. The LPWN will be applied efficiently to progress mining.

Conflict of Interests
The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments
This work is supported by the National Natural Science Foundation of China under Grants 61170078 and 61173042; the National Basic Research Program of China under Grant 2010CB328101; the Doctoral Program of Higher Education of the Specialized Research Fund of China under Grant 20113718100004; Basic Research Program of Qingdao City of China under Grant no. 13-1-4-116-jch; the SDUST Research Fund of China under Grant 2011KYTD102; and Graduate Innovation Foundation of Shandong University of Science and Technology under Grant YC140360.

References
[1] Z. Li and M. Zhou, “Control of elementary and dependent siphons in Petri nets and their application,” IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, vol. 38, no. 1, pp. 133–148, 2008.
[2] Y.-S. Huang and T.-H. Chung, “Modelling and analysis of air traffic control systems using hierarchical timed coloured Petri nets,” Transactions of the Institute of Measurement and Control, vol. 33, no. 1, pp. 30–49, 2011.
[3] H. Hu, M. C. Zhou, and Z. Li, “Low-cost and high-performance supervision in ratio-enforced automated manufacturing systems using timed Petri nets,” IEEE Transactions on Automation Science and Engineering, vol. 7, no. 4, pp. 933–944, 2010.
[4] W.-L. Chen, C.-D. Kan, C.-H. Lin, and T. Chen, “A rule-based decision-making diagnosis system to evaluate arteriovenous shunt stenosis for hemodialysis treatment of patients using fuzzy petri nets,” IEEE Journal of Biomedical and Health Informatics, vol. 18, no. 2, pp. 703–713, 2014.
[5] Y. Wang, C. Lin, P. D. Ungsunan, and X. Huang, “Modeling and survivability analysis of service composition using Stochastic Petri Nets,” The Journal of Supercomputing, vol. 56, no. 1, pp. 79–105, 2011.
[6] Y. Du, C. Jiang, and M. Zhou, “A Petri net-based model for verification of obligations and accountability in cooperative systems,” IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, vol. 39, no. 2, pp. 299–308, 2009.
[7] W. Liu, Y. Du, and C. Yan, “Soundness preservation in composed logical time workflow nets,” Enterprise Information Systems, vol. 6, no. 1, pp. 95–113, 2012.
[8] Y. Y. Du, Y. H. Ning, and L. Qi, “Reachability analysis of logic Petri nets using incidence matrix,” Enterprise Information Systems, vol. 8, no. 6, pp. 630–647, 2014.
[9] A. K. A. de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst, “Genetic process mining: an experimental evaluation,” Data Mining and Knowledge Discovery, vol. 14, no. 2, pp. 245–304, 2007.
[10] W. M. P. van der Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer, Berlin, Germany, 2011.
[11] W. Tan and M. Zhou, Business and Scientific Workflows: A Service-Oriented Approach, IEEE Press/Wiley, Hoboken, NJ, USA, 2013.
[12] W. Yu, C. Yan, Z. Ding, C. Jiang, and M. Zhou, “Modeling and validating E-commerce business process based on petri nets,” IEEE Transactions on Industrial Informatics, vol. 44, no. 3, pp. 327–341, 2014.
[13] R. M. Dijkman, M. Dumas, and C. Ouyang, “Semantics and analysis of business process models in BPMN,” Information and Software Technology, vol. 50, no. 12, pp. 1281–1294, 2008.
[14] W. M. P. van der Aalst, K. M. van Hee, A. H. M. ter Hofstede et al., “Soundness of workflow nets: classification, decidability, and analysis,” Formal Aspects of Computing, vol. 23, no. 3, pp. 333–363, 2011.
[15] W. M. P. van der Aalst, V. Rubin, H. M. W. Verbeek, B. F. van Dongen, E. Kindler, and C. W. Günther, “Process mining: a two-step approach to balance between underfitting and overfitting,” Software & Systems Modeling, vol. 9, no. 1, pp. 87–111, 2010.
[16] J. E. Ingvaldsen and J. A. Gulla, “Industrial application of semantic process mining,” Enterprise Information Systems, vol. 6, no. 2, pp. 139–163, 2012.
[17] M. P. Wil van der Aalst, “Decomposing Petri nets for process mining: a generic approach,” Distributed and Parallel Databases, vol. 31, no. 4, pp. 471–507, 2013.
[18] M. Goeminne and T. Mens, “A comparison of identity merge algorithms for software repositories,” Science of Computer Programming, vol. 78, no. 8, pp. 971–986, 2013.
[19] Y. Chen, S. Alspaugh, and R. Katz, “Interactive analytical processing in big data systems: a cross-industry study of MapReduce workloads,” Proceedings of the VLDB Endowment, vol. 5, no. 12, pp. 1802–1813, 2012.