Computing and Informatics, Vol. 41, 2022, 1025–1053, doi: 10.31577/cai_2022_4_1025

GUARD-FUNCTION-CONSTRAINT-BASED REFINEMENT METHOD TO GENERATE DYNAMIC BEHAVIORS OF WORKFLOW NET WITH TABLE

Jian Song

Department of Computer Science
Tongji University
201804 Shanghai, China
e-mail: `1910690@tongji.edu.cn`

Dongming Xiang

School of Information Science and Technology
Zhejiang Sci-Tech University
310018 Hangzhou, China
e-mail: `flysky_xdm@163.com`

Guanjun Liu*, Leifeng He

Department of Computer Science
Tongji University
201804 Shanghai, China
e-mail: `{liuguanjun, 1710052}@tongji.edu.cn`

Abstract. In order to model complex workflow systems with databases, and detect their data-flow errors such as data inconsistency, we defined Workflow Net with Table model (WFT-net) in our previous work. We used a Petri net to describe control flows and data flows of a workflow system, and labeled some abstract table operation statements on transitions so as to simulate database operations. Meanwhile, we proposed a data refinement method to construct the state reachability graph

* Corresponding author

of WFT-nets, and used it to verify some properties. However, this data refinement method has a defect, i.e., it does not consider the constraint relation between guard functions, and its state reachability graph possibly has some pseudo states. In order to overcome these problems, we propose a new data refinement method that considers some constraint relations, which can guarantee the correctness of our state reachability graph. What is more, we develop the related algorithms and tool. We also illustrate the usefulness and effectiveness of our method through some examples.

Keywords: WFT-net, state reachability graph, data refinement, pseudo states, Petri net

Mathematics Subject Classification 2010: 68-Q60

1 INTRODUCTION

Due to the complex business logics and a large number of data operations, workflow systems have become increasingly complicated. Thus, it increases the difficulty of verifying the correctness of workflow models. In the design stage of a workflow system, whether the bottom layer that implements specific activity operations, or the upper layer that abstracts its process model into a summary framework, all of their correctness and effectiveness are needed to be guaranteed.

As is well known, the correctness and effectiveness of a process model depends on both control flows and data flows [1]. The control flows record the behavioral profile relations between activities (e.g., strict order relation, exclusiveness relation, and interleaving order relation, etc.) [2]. The data flows reflect the correlation between data items, data operations and guards [3]. If there are unreasonable data operations in the execution of some activities, data-flow errors may occur [4, 5]. In fact, data flows and control flows are unified to detect abnormal data errors, which can strengthen the analysis ability of business process management [6, 7, 8]. A good modeling method contributes to analyzing a workflow system. Petri net, as a good formalization language [9, 10], can greatly describe concurrency and synchronization relations. Currently, it has been widely used in modeling and analyzing of concurrent or distributed systems [11, 12, 13]. In general, the reachability graph of a Petri net is used to detect anomalies[1]. Especially, the guard-driven reachability graph of a workflow net with data (WFD-net). It can avoid pseudo states and alleviate the state space explosion problem [14, 15].

[1] In order to distinguish, it is called the reachability graph in Petri net, and the state reachability graph in other nets.

Trčka et al. [16, 17, 18] proposed WFD-nets to model workflow systems, and detected their data-flow errors by anti-patterns. Furthermore, data footprint was introduced in [19]. As a directed graph representing data flows, it was abstracted from the state reachability graph of a WFD-net. In order to use a model checker to refine the specifications between states, Smith and Derrick [20] improved state symbols so as to avoid blocking in a process model. Ge et al. [21] and Gardiner and Morgen [22] adopted a task refinement method, which used refinement rules and mutual transformations between predicates to analyse the reachability graph of a Petri net, which can avoid state space explosion. Using action optimization has a good effect on processing causal ambiguity systems [23]. When a complex workflow model deals with massive concurrent data operations (e.g., read, write, delete), it is prone to data-flow errors. In order to improve the accuracy of a workflow model, Sidorova et al. [24] added *read*/*write*/*delete* labelling functions to transitions, and they proposed a new data refinement method to analyze false negative/positive activities in a WFD-net.

Although WFD-nets can describe abstract data operations in business processes [25, 26], the actual workflow systems usually cannot work without background databases. Naturally, some data-flow errors related to table operations or logical defects cannot be reflected in a WFD-net. Given this problem, we proposed Workflow Net with Table (WFT-net) [27]. WFT-net uses a WFD-net to model control flows and data flows of workflow systems, and utilizes data statements related to tables to describe database operations. That is, each transition of a WFT-net is marked by the statement of table operations so as to establish the connection between business logics and databases [27]. In our previous work, a data refinement method was given to generate the state reachability graph of WFT-net, which can describe all possible running information of a workflow system. However, this refinement method has a drawback. That is, when guard functions operate on the same data item, pseudo states may be produced in the state reachability graph (c.f. the motivation example in Section 2), since the refinement method does not consider the constraint relation between guard functions. In order to overcome this problem, a new refinement method is proposed based on guard function constraints in this paper. That is, when different guard functions assign values to the same data item, there is a constraint relationship between them, and their expression of guard function constraints is generated. According to this constraint expression, some states satisfying expression are selected. Furthermore, a guard-driven state reachability graph is constructed.

The rest of this paper is organized as follows. Section 2 presents some basic notations. Section 3 gives an example of motivation. Section 4 formalizes WFTC-net (Workflow Net with Table and Constraints) and its firing rules. Moreover, the principle of data refinement and an algorithm for generating the state reachability graph of a WFTC-net are proposed. Section 5 conducts a case study to illustrate the effectiveness of our method. Section 6 develops our tool and does a group of experiments. Section 7 concludes this paper.

2 BASIC NOTATIONS

Definition 1 (Petri net [28, 29, 30, 31])**.** A net is a triple $N = (P, T, F)$, where P is a finite set of places, T is a finite set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation, and $P \cap T = \emptyset \wedge P \cup T \neq \emptyset$. A marking of a net is a mapping function $M : P \to \mathbb{N}$, where $\mathbb{N} = \{0, 1, 2, \dots\}$ is the set of non-negative integers. In other word, $M(p)$ is the number of tokens in the place p. A net N with an *initial marking* M_0 is a *Petri net* and denoted as $PN = (N, M_0)$. Note that our marking is represented by a multi-set. For instance, $M = [p_0 + 2p_2]$ is a marking, where $M(p_0) = 1$ and $M(p_2) = 2$. For each node $x \in P \cup T$, its preset is denoted by ${}^\bullet x = \{y \,|y \in S \cup T \wedge (y, x) \in F\}$. Similarly, its postset is denoted by $x^\bullet = \{y \,|y \in S \cup T \wedge (x, y) \in F\}$.

Given a Petri net $PN = (N, M_0)$ and one marking M, a transition t is *enabled* at M, denoted as $M[t\rangle$, if $\forall p \in {}^\bullet t : M\,(p) \geq 1$. A new marking M' is generated from marking M by firing the transition t, which is denoted as $M[t\rangle M'$, where for $\forall p \in P$:

$$M'\,(p) = \begin{cases} M\,(p) - 1, & \text{if } p \in {}^\bullet t - t^\bullet, \\ M\,(p) + 1, & \text{if } p \in t^\bullet - {}^\bullet t, \\ M\,(p)\,, & \text{otherwise.} \end{cases}$$

Definition 2 (Workflow net [25, 32])**.** A net $N = (P, T, F)$ is a workflow net (WF-net) if:

1. N has two special places, i.e., one source place *start* and one sink place *end* in P such that ${}^\bullet start = \emptyset$ and $end^\bullet = \emptyset$; and
2. $\forall x \in P \cup T$: $(start, x) \in F^*$ and $(x, end) \in F^*$, where F^* is the reflexive-transitive closure of F.

Definition 3 (Table)**.** A table $R = \{r_1, r_2, \dots, r_n\}$ is a set of finite records. Each record $r_i = \{d_1, d_2, \dots, d_k\}$ represents the values of k attributes, where d_k represents the value of the k^{th} attribute value [33].

For example, a table *Student* is shown in Figure 1 b), which contains two records $r_1 = \{name1, id1, grad1\}$ and $r_2 = \{name2, id2, grad2\}$.

Definition 4 (Workflow Net with Table [27])**.** A Workflow Net with Table (WFT-net) is a 14-tuples $N = (P, T, F, G, D, R, rd, wt, dt, sel, ins, del, upd, guard)$ where

1. (P, T, F) is a WF-net;
2. G is a set of guard functions;
3. D is a finite set of data items;
4. $R = \{r_1, r_2, \dots, r_k\}$ is an initial table consisting of k records;
5. $rd : T \to 2^D$ is the labeling function of reading data;
6. $wt : T \to 2^D$ is the labeling function of writing data;

7. $dt : T \to 2^D$ is the labeling function of deleting data;
8. $sel : T \to 2^R$ represents the labeling function of selecting operation in the table R;
9. $ins : T \to 2^R$ represents the labeling function of inserting operation in the table R;
10. $del : T \to 2^R$ represents the labeling function of deleting operation in the table R;
11. $upd : T \to 2^R$ represents the labeling function of updating operation in the table R; and
12. $guard : T \to G_\Pi$ is the assigning function of guard functions. G_Π is a set of guard functions, each of which is a Boolean expression over a set of predicates $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$, where π_i is a predicate defined on D or R.

In a WFT-net, a guard function is a Boolean expression of some data items especially in tables. It is a formal representation of data conditions related to table operations. $Var(G)$ represents the variables in the guard function G.

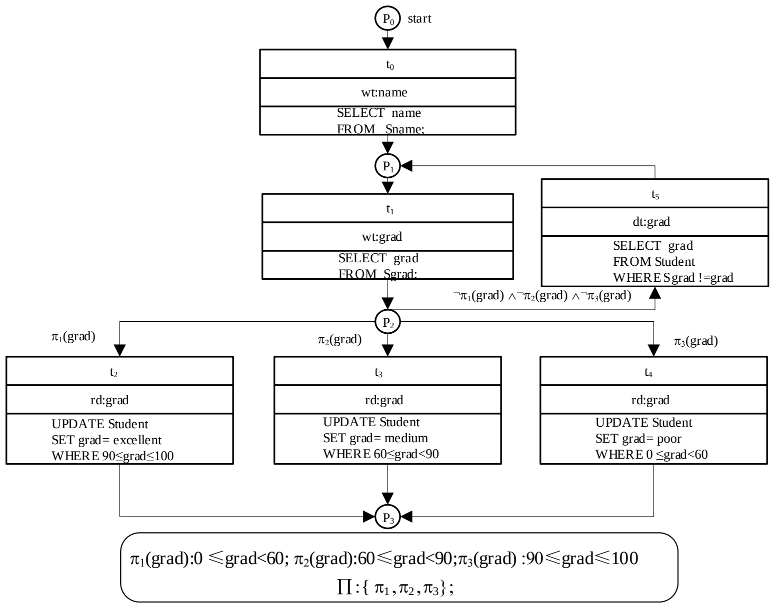
Figure 1 is a simple business process of student performance evaluation. Figure 1 a) describes the basic business logics, data operations, database operations, and guard functions assigned to transitions. Figure 1 b) gives an initial table.

1. $D = \{name, grad\}$ is a set of data items, where *name* represents a student's name and *grad* means the student's grade, which can be regarded as intermediate variables for a user to operate a database;
2. $Student = \{r_1, r_2\}$ is a table, where each record is composed of three attributes, i.e., *Sname*, *Sid* and *Sgrad*, which represent a student's *name*, *id* and *grade*, respectively;
3. $\Pi = \{\pi_1, \pi_2, \pi_3\}$ is a set of predicates, where $G_\Pi = \{\pi_1, \pi_2, \pi_3, \neg\pi_1 \wedge \neg\pi_2 \wedge \neg\pi_3\}$ and $Var(\pi_1) = Var(\pi_2) = Var(\pi_3) = Var(\neg\pi_1 \wedge \neg\pi_2 \wedge \neg\pi_3) = \{grad\}$; and
4. $wt(t_0) = name$, $sel(t_0) = name$.

3 MOTIVATION

In order to describe all running behaviors of a WFT-net, it is necessary to construct its states and their transition relations. Since the generating states and firing rules of WFT-nets are both related to the data operations in a table, data needs to be further refined. In our previous work, a data refinement algorithm was given in [27]. As shown in Algorithm 1, B_d is the value range of data item d, R_d is a set of all table data items associated with d, and Π_d is a set of all predicate expressions associated with d. For a WFT-net, if a transition t has a *write* operation on d, then d needs to be refined. If t can be fired at a state [2], then the refinement method can calculate V_d, i.e., a refinement set of d. In Algorithm 1, if n_{R_d} and $n_{\prod_d}$ are respectively the cardinality of R_d and $\prod_d$, we can get its time complexity is $O\left(n_{R_d} + n_{\prod_d}\right)$.

[2] A state of a WFT-net is usually called as a ***configuration***.



a)

Student

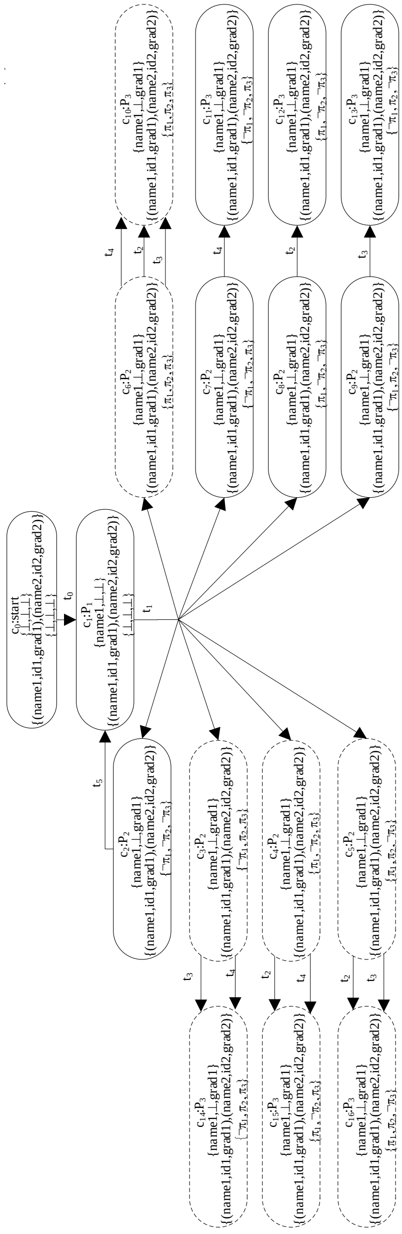
Sname	Sid	Sgrad
name1	id1	grad1
name2	id2	grad2

b)

π_1	π_2	π_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

c)

The data refinement method in Algorithm 1 still has a shortcoming, i.e., it may produce pseudo states in some cases. If a **read/write** operation on a transition is associated with a data item in k guard functions, then the transition needs to consider all possible assignment values of guard functions, i.e., it will generate 2^k states. If there are multiple guard functions, it is easy to cause a rapid growth of states, and result in the state space explosion problem. As shown in Figure 1 a), **grad** is written at t_1, and predicates π_1, π_2 and π_3 are associated with **grad**. After firing transition t_1, it will produce $2^3 = 8$ reachability states (i.e., C_2–C_9). All values of the guard functions are described by the truth table in Figure 1 c). Due to the mutual constraint relation between guard functions, the guard function π_1 satisfies the condition, while π_2 and π_3 do not so. Since Algorithm 1 does not consider this constraint relation, pseudo states are generated after firing t_1. Figure 1 d) is the state reachability graph of the WFT-net in Figure 1 a), where a pseudo state is



d)

Figure 1. a) A WFT-net; b) an initial table *Student*; c) as distribution table of guard function values without constraints; d) a state reachability graph

Algorithm 1 Generating a refinement set V_d of data item d

Input: A WFT-net N, d, B_d, c;

Output: A refinement set V_d;

if $Rd = \emptyset \wedge \Pi_d = \emptyset$ **then**
  Select an arbitrary d_0 from B_d, and add d_0 into V_d;
else
  if $R_d \neq \emptyset$ **then**
    for each $R_i \in R_d$ **do**
      Add all stord values of d in R_i at c into V_d;
    end for
    Select an arbitrary d_0 from $R_d \setminus V_d$, and add d_0 into V_d;
  end if
  if $\Pi_d \neq \emptyset$ **then**
    for each $\pi_i \in \Pi_d$ **do**
      if π_i is defined **then**
        if any data item in V_d cannot make π_i true **then**
          Select a d_i from B_d that make π_i true and then add d_i into V_d;
        end if
        if any data item in V_d cannot make π_i be false **then**
          Select a d_i' from B_d that makes π_i false and then add d_i' into V_d;
        end if
      end if
    end for
  end if
end if
return V_d.

represented by a dashed box. In order to solve this problem, this paper proposes WFTC-net and a new data refinement method.

4 A NEW REFINEMENT METHOD AND REACHABILITY GRAPH GENERATION ALGORITHM

In order to solve the pseudo state problem, WFTC-net is defined and its constraint relations between guard functions are considered. Meanwhile, a new refinement method is proposed to generate an accurate state reachability graph.

4.1 Workflow Net with Table and Constraints

By adding constraint relations between guard functions into WFT-net, Workflow Net with Table and Constraints (WFTC-net) is formalized.

Definition 5 (Workflow Net with Table and Constraints)**.** A Workflow Net with

Table and Constraints (WFTC-net) is a 15-tuples $N = (N', res)$ where

1. N' is a WFT-net; and
2. Given a set of predicates $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$, its elements are combined by $\wedge$, $\vee$ and $\neg$ to form a proposition formula ω. Some formulas are combined by $\vee$ to form a constraint, which is assigned to the values of *true* (**T**) or *false* (**F**). *res* is a set of such constraints.

For the example of Figure 1 a), there is a set of predicates $\Pi = \{\pi_1, \pi_2, \pi_n\}$. Since π_1, π_2 and π_3 are both *write* operations on the same data item *grad*, there exist mutual constraint relations between them. Thus, we can define three proposition formulas: $\omega_1 = \pi_1 \wedge \neg\pi_2 \wedge \neg\pi_3$, $\omega_2 = \neg\pi_1 \wedge \pi_2 \wedge \neg\pi_3$, $\omega_3 = \neg\pi_1 \wedge \neg\pi_2 \wedge \pi_3$, and $\omega_4 = \neg\pi_1 \wedge \neg\pi_2 \wedge \neg\pi_3$. At the same time, we can calculate a constraint $true \models \omega_1 \vee \omega_2 \vee \omega_3 \vee \omega_4$ and a constraint set $res = \{\omega_1 \vee \omega_2 \vee \omega_3 \vee \omega_4\}$.

Definition 6 (State)**.** Give a WFTC-net $N = (N', res)$, a four-tuples $c = \langle M, \theta_D, \vartheta_R, \sigma\rangle$ is called a state of N, where

1. M is a marking of N;
2. $\theta_D : D \to \{\perp, \top\}$ is the value of the data items in the current state. When a data item is read or written, it indicates that the value of the data item is ***defined*** and is represented by the symbol $\top$. Otherwise, its value is ***undefined*** and is represented by the symbol $\perp$;
3. $\vartheta_R : R \to \{\perp, \top\}$ is the value of the current table, which reflects the situation of records in a table. When a data item is read or written in the table, it means that the value of the data item is ***defined*** and is represented by $\top$. Otherwise, its value is ***undefined*** and is represented by $\perp$. Each data item of R associated with N is stored in a two-dimensional table, and each tuple information in table R corresponds to a data item d;
4. $\sigma : \Pi \to \{true, false, \perp\}$ represents the assignment state of each predicate. Since each predicate is associated with some data items, when its relevant data items is written to a specific value, it is assigned to true (**T**) or false (**F**). Otherwise, its value is still ***undefined***$(\perp)$.

For example, the initial state of the WFT-net in Figure 1 is $c_0 = \langle \boldsymbol{start}, \{name = \perp, id = \perp, grad = \perp\}, \{(name1, id1, grad1), (name2, id2, grad2)\}, \{\pi_1 = \perp, \pi_2 = \perp, \pi_3 = \perp\}\rangle$. At this time, t_0 is enabled at c_0, and the token will move from ***start*** to p_1 after firing t_0. At the transition t_0, only the data item *name* is written, so it is *defined*, while the data items *id*, *grad* are still *undefined*. Performing a selection operation on the data item *name* at t_0 will not change the value of the data items in this table. Therefore, the records $\{(name1, id1, grad1), (name2, id2, grad2)\}$ in the table remain unchanged. Since guard functions are not bound at t_0, π_1, π_2 and π_3 are still *undefined*. Thus, firing t_0 generates a new state, i.e., $c_1 = \langle \boldsymbol{start}, \{name = \top, id = \perp, grad = \perp\}, \{(name1, id1, grad1), (name2, id2, grad2)\}, \{\pi_1 = \perp, \pi_2 = \perp, \pi_3 = \perp\}\rangle$.

In order to facilitate the understanding of WFTC-net, the conceptual framework of a WFTC-net is presented in Figure 2. The bottom layer uses a WFD-net to describe the control flows and data flows in a process model, and the upper layer uses a table to represent the database, which realizes the operations of data items in this table. After then a constraint set *res* is added into a WFT-net, and forms a WFTC-net. By marking some operation statements of tables on transitions, it can reflect some data-flow errors in a workflow system.

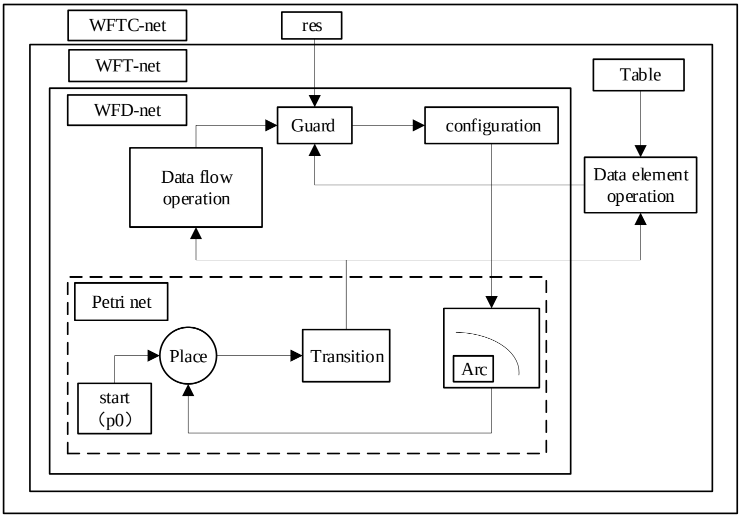


Figure 2. WFTC-Net conceptual framework

4.2 A Data Refinement Method Based on Guard Function Constraints

Based on guard function constraints, a new data refinement method proposed, as shown in Algorithm 2. We first use a truth table to enumerate all possible assignments of guard functions. After then, we utilize the constraints between guard functions to construct a set of expressions, and find out reasonable states in the truth table.

For a WFTC-net N, if a data item d is written at a transition t, then this data item needs to be refined. According to Algorithm 2, *Rvd* is a set of data items refined at the state c, R is a table associated with N, *Sat*(*guard*) is a set of guard functions in N, *Sat*(R) is a set of data items in R, N_D is a set of data items in N, N_R is a set of data items associated with R in N, *guard*(t) is a guard function on transition t, and *res* is a set of contraints. Algorithm 2 gives a detailed data refinement method.

First, it chooses a data item d_i and initializes *Rvd*, as shown in step 1. Then *Rvd* is computed by operating on d_i according to different cases, as shown in steps 6–19.

Finally, Rvd and res are computed, as shown in steps 20–22. In Algorithm 2, n_{N_d} is the cardinality of N_d. Then the time complexity of Algorithm 2 is $O(n_{N_d})$.

Compared with Algorithms 1 and 2 provides a data refinement method under guard function constraints so that our method can avoid generating pseudo states and alleviate the state space explosion problem.

4.3 The State Reachability Graph of WFTC-Net

Based on our data refinement method, we give the firing rules of WFTC-net as follows.

Definition 7 (The firing rule of WFTC-net)**.** Let $N = (N', \mathit{res})$ be a WFTC-net. $t \in T$ is enabled at one state $c = \langle M, \theta_D, \vartheta_R, \sigma\rangle$ (denoted by $c[t\rangle$) if and only if:

1. $\forall t \in T$: $M[t\rangle$;
2. $\forall d \in (rd(t) \cap D) : \theta(d) = \top$; $\forall d \in (wt(t) \cap D) : \theta(d) = \top$; $\forall d \in (dt(t) \cap D) : \theta(d) = \perp$;
3. $\forall d \in (R \cap \mathit{del}(t)) : \vartheta(d) = \perp$; $\forall d \in (R \cap (\mathit{sel}(t) \cup \mathit{ins}(t) \cup \mathit{upd}(t))) : \vartheta(d) = \top$; and
4. $\forall d \in var(G(t))$: $\theta(d) = \top \cap \vartheta(d) = \top$ and $\sigma(G(t)) = \mathit{true}$.

After firing the transition t, a transition t is enabled at state c. A new state $c' = \langle M', \theta'_D, \vartheta'_R, \sigma'\rangle$ is generated, which is denoted as $c[t\rangle c'$ such as:

1. $M[t\rangle M'$;
2. $\forall d \in dt(t) : \theta'(d) = \perp$; $\forall d \in (wt(t) \bigcup rd(t)) : \theta'(d) = \top$; $\forall d' \in Rvd : \theta'(d) = d'$;
3. $\forall d \in (wt(t) \setminus dt(t)) : \theta'(d) = \top$; $\forall d \in D \setminus (dt(t) \cup wt(t)) : \theta'(d) = \theta(d)$;
4. $\forall R' \in \mathit{ins}(t), \forall \mathit{ins}(R') \cap R \neq \emptyset : \vartheta'(R') = \vartheta(R')$; $\forall R' \in \mathit{ins}(t), \forall \mathit{ins}(R') \cap R = \emptyset : \vartheta'(R') = \vartheta(R') \cup \mathit{ins}(R')$;
5. $\forall R' \in \mathit{del}(t), \forall \mathit{del}(R') \subset R : \vartheta'(R') = \vartheta(R') \setminus \mathit{del}(R')$; $\forall R' \in \mathit{del}(t), \forall \mathit{del}(R') \not\subset R : \vartheta'(R') = \vartheta(R')$;
6. $\forall R' \in \mathit{upd}(t), \mathit{upd}(R') \subset R' : \vartheta'(R') = \vartheta(R') \setminus \mathit{upd}(R') \cup \mathit{upd}(R')'$;
7. $\forall R' \in \mathit{sel}(t), \forall \mathit{sel}(R') \subset R : \vartheta'(R') = \top$; $\forall R' \in \mathit{sel}(t), \forall \mathit{sel}(R') \not\subset R : \vartheta'(R') = \perp$;
8. $\exists g \in G$, $d \in (wt(t) \cup rd(t)) \cap (\mathit{upd}(t) \cup \mathit{ins}(t) \cup \mathit{sel}(t)) : \sigma'(g) = \mathit{true}$;
9. $\exists g \in G$, $d \in (dt(t) \cap \mathit{del}(t)) : \sigma'(g) = \mathit{false}$;
10. $\forall g \notin G$, $d \in (\theta_D \cup \vartheta_D) : \sigma'(g) = \perp$; and
11. $\forall g \in G$, $d \in var(\mathit{res}) : \sigma(g) = \mathit{true}$.

According to Definition 11, a constraint $\mathit{true} \models (\pi_1 \wedge \neg\pi_2 \wedge \neg\pi_3) \vee (\neg\pi_1 \wedge \pi_2 \wedge \neg\pi_3) \vee (\neg\pi_1 \wedge \neg\pi_2 \wedge \pi_3) \vee (\neg\pi_1 \wedge \neg\pi_2 \wedge \neg\pi_3)$ is obtained from Figure 1 a), and its truth table is shown in Figure 3 a). The result shows that there are only four reasonable assignments for the three guard functions.

Algorithm 2 A data refinement method for WFTC-net

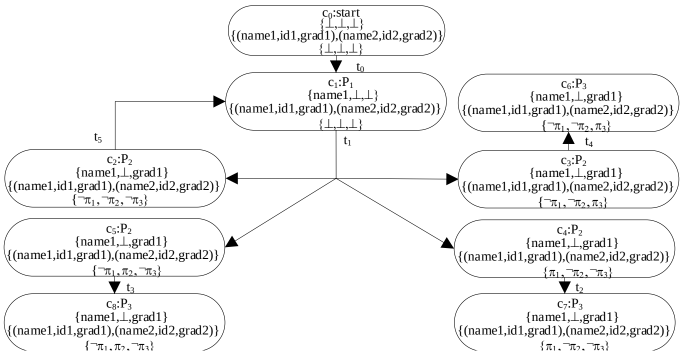
Input: A WFTC-net N, $R = \{D_1, D_2, \ldots, D_n\}$, $Sat(guard)$, $Sat(R)$;
Output: Data refinement set Rvd, constraint set res;

Select data items d_i, and initialize $Rvd \leftarrow \perp$;
if $d_i \in Sat(guard) \wedge d_i \in Sat(R)$ **then**
  $guard(t) \leftarrow \top$;
  for $N_d(d_i) \in N_R$ **do**
    $Rvd \leftarrow d_i$;
  end for
end if
if $d_i \in Sat(guard)) \wedge d_i \notin Sat(R)$ **then**
  $guard(t) \leftarrow \perp$;
  if $d_i \in N_d$ **then**
    $Rvd \leftarrow d_i$;
  else
    $Rvd \leftarrow Rvd + d_i$;
  end if
end if
if $d_i \notin Sat(guard) \wedge d_i \in Sat(R)$ **then**
  $guard(t) \leftarrow guard(^{\bullet}t)$;
  for $d_i \in N_R(j)$ **do**
    $Rvd \leftarrow d_i$;
  end for
end if
if $d_i \notin Sat(guard) \wedge d_i \notin Sat(R)$ **then**
  $guard(t) \leftarrow guard(^{\bullet}t)$
  if $d_i \in N_D$ **then**
    $Rvd \leftarrow d_i$;
  else
    $Rvd \leftarrow Rvd + d_i$;
  end if
end if
if $d_i = null$ **then**
  $guard(t) \leftarrow guard(^{\bullet}t)$, $Rvd \leftarrow Rvd$;
end if
if $d_i \in Sat(guard_i)$ **then**
  $\omega_i = \bigvee_{i=0}^{|Sat(guard_i)|} guard_i \wedge (Sat(guard_i) - guard_i))$, $res \leftarrow \omega_i$;
end if
return res, Rvd.

Algorithm 3 is developed to generate all reachable states from c_0. In this algorithm, *res* and *Rvd* are the results, where *operation*(θ_D) represents operations (i.e., *read*, *write*, *delete*) on data in N, *operation*(ϑ_R) represents operations on data items in R, and *add*(c) represents adding a new state. In this algorithm, steps 5–29 generate a new state c'. Steps 31–34 skip one transition and look for another new enabled transition since the generated state is repeated. In Algorithm 3, the time complexity is $O(1)$.

π_1	π_2	π_3
0	0	0
0	0	1
0	1	0
1	0	0

a)



b)

Figure 3. a) A distribution table of the guard function values with constraints; b) A state reachability graph of WFTC-net

According to the firing rule of WFTC-net, we propose an algorithm for generating its state reachability graph. Based on the depth-first idea, as shown in Algorithm 4.

According to Algorithm 4, the state reachability graph of Figure 1 a) is generated as shown in Figure 3 b). When the data item *grad* is written at t_1, since there is a constraint *true* $\models (\pi_1 \wedge \neg\pi_2 \wedge \neg\pi_3) \vee (\neg\pi_1 \wedge \pi_2 \wedge \neg\pi_3) \vee (\neg\pi_1 \wedge \neg\pi_2 \wedge \pi_3) \vee (\neg\pi_1 \wedge \neg\pi_2 \wedge \neg\pi_3)$, some assignments of π_1, π_2 and π_3 such as $\{(0,1,1),(1,0,1),(1,1,1)\}$ in Figure 1 c) do not exist. Therefore, such pseudo states need to be removed. In fact, the state reachability graph generated by Algorithm 4 is more in line with the actual demand. In Algorithm 4, if n_t and $n_c^{'}$ are respectively the cardinality of t and $c^{'}$, its time complexity is $O(n_t + n_c^{'})$.

5 CASE STUDY

This section shows the effetiveness of our method through a case study of private car application for access control in a community. In order to simplify this process, some irrelevant operations are omitted. We first use a WFT-net to model this process, as shown in Figure 4. A user inputs an account *id* to log in the system (t_0). If s/he has never registered in this system, s/he needs to re-apply for registration (t_2). S/he

Algorithm 3 Generate state set of a WFTC-net

Input: R, WFTC-net N, *res*, *Rvd*;
Output: $sat(c')$;

Transition $t \leftarrow NULL$, State $c \leftarrow NULL$, Hashtable $h \leftarrow NULL$;
The initial state $c_0 = \langle \boldsymbol{start}, \bot, \top, \bot \rangle = \langle m, \theta_D, \omega_R, \sigma \rangle$;

if $t_i \in T \wedge m[t_i\rangle \wedge guard(t_i) = \sigma(\pi_i)$ or $t_i \in T \wedge m[t_i\rangle \wedge guard(t_i) = null$ **then**
  $t \leftarrow t_i$;
else
  $i \leftarrow i + 1$;
  if $t_i \in t \wedge m[t_i\rangle m'$, $add(m')! = h$ **then**
    $c[t_i\rangle c'$; // There is no repeat state
    if $\forall d \in operation(\theta_D)$, $\forall d \in Rvd$ **then**
      $\theta'(d) \leftarrow d$;
    else
      $\forall d \in operation(\theta_D) : \theta'(d) \leftarrow \bot$;
    end if
    if $\forall R' \in operation(\vartheta_R)$, $\forall \vartheta_R[d] \in Rvd$ **then**
      $\vartheta'(R') \leftarrow \vartheta(R)$;
    else
      $\forall R' \notin operation(\vartheta_R)$, $\vartheta'(R') \leftarrow \bot$;
    end if
    if $\forall guard(t) \in operation(\Pi)$, $guard(t) \in Rvd$ **then**
      $\sigma' \leftarrow true$;
    else
      $\sigma' \leftarrow false$
    end if
    if $guard_1 \in res, \ldots, guard_i \in res$ **then**
      $guard_j \in true$, $(res - guard_j) \leftarrow false$;
    end if
    $c' \leftarrow \langle m', \theta'(d), \vartheta'(R'), \sigma' \rangle$, then add c' into $Sat(c')$;
  else
    if $t_i \in t \wedge c[t_i\rangle c_m$, $add(c_m) = h$ **then**
      $t_i \leftarrow t_i + 1$; // There are repeat state
    end if
  end if
end if

return $Sat(c')$.

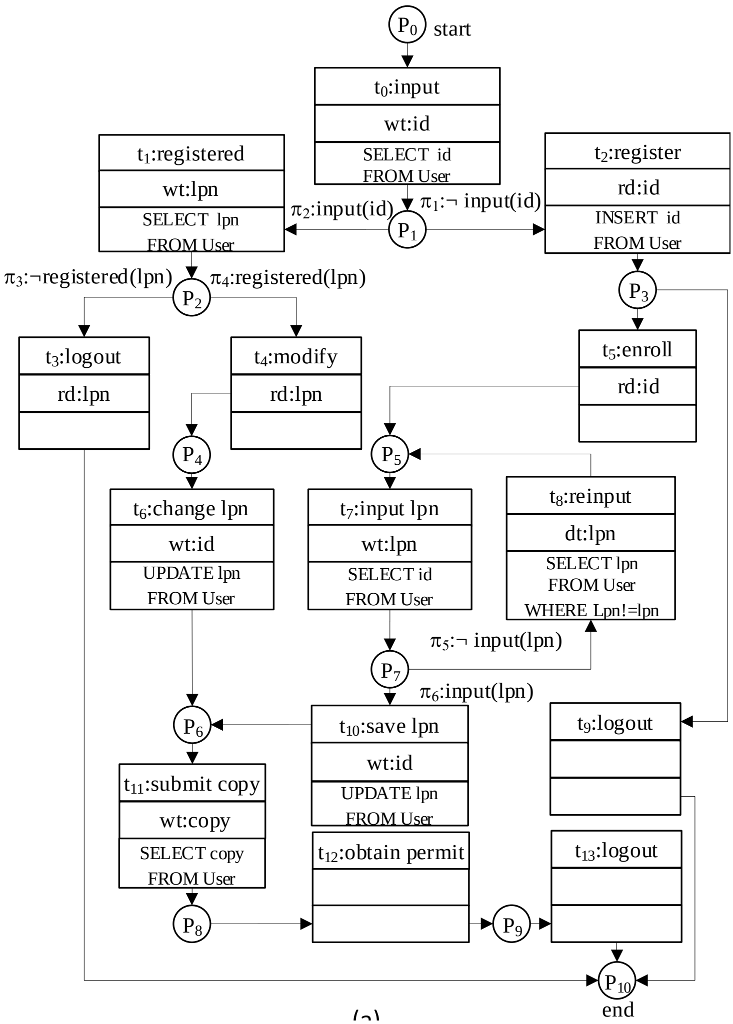
Algorithm 4 Generate the state reachability graph of a WFTC-net

Input: WFTC-net N, $Sat(c')$;
Output: $RG(N)$;

1: Take c_0 as the root node of RG(N) and mark it as ***new***;;
2: **while** there is a node marked ***new*** **do**
3: Make the node as c;
4: **end while**
5: **if** there is a directed path from c_0 to c and the marking of a node is c **then**
6: Change the marking of c to ***old***, and return to step 2;
7: **end if**
8: **if** $\forall t \in T : \neg c[t\rangle$ **then**
9: Change the marking of c to ***endpoint***, and return to step 2;
10: **end if**
11: **for** $\forall t \in T : c[t\rangle$ **do**
12: Calculate $Sat(c')$ according to Algorithm 3;
13: **end for**
14: **if** $\forall t \in T : c[t\rangle$ **then**
15: Calculate c' in $c[t\rangle c'$ according to Definition 11;
16: **end if**
17: **for** $\forall c' \in Sat(c')$ **do**
18: **if** c' already exists in the directed path from c_0 **then**
19: Draw a directed arc from c to c', and mark the side of the arc as t;
20: **else**
21: Generate a node c' and mark it as ***new*** in RG(N); draw a directed arc from c to c' and mark the side of the arc as t; erase the ***new*** label of node c, and return to Step 2.
22: **end if**
23: **end for**

first enters the registration interface to access this system (t_5) and then submits the materials information related to the driver license (t_7). If the review is passed, s/he can continue to modify or update other information (t_{10}). Otherwise, it is necessary to resubmit the information again (t_8). If s/he registers successfully, s/he becomes a registered *user*. This *user* can choose to exit this system (t_3) or modify his/her personal information (t_4). If this *user* needs to modify the license plate number information, s/he needs to enter his/her name first, and then update the license plate number information. At the same time, s/he needs to submit the relevant copy materials and upload them to the system (t_{11}). After all the copy materials (*copy*) are submitted and approved (t_{12}), the permission can be obtained. Finally, this *user* can exit and the process ends (t_{13}).

The initial state of the WFT-net in Figure 4 is $c_0 = \langle \boldsymbol{start}, \{id = \perp, lpn = \perp, copy = \perp\}, \{(id1, lpn1, copy1), (id2, lpn2, copy2)\}, \{\pi_1 = \perp, \pi_2 = \perp, \pi_3 = \perp, \pi_4 = \perp, \pi_5 = \perp, \pi_6 = \perp\}\rangle$. The transition t_0 is enabled at c_0, and the token will move



a)

User

Id	Lpn	Copy
id1	lpn1	copy1
id2	lpn2	copy2

b)

Figure 4. WFT-net for vehicle management system

from p_0 to p_1 after firing t_0. The data item *id* is *defined*, but *lpn* and *copy* are still *undefined*, so only *id* is written at t_0. Guard functions π_1, π_2 are both about this write operation. According to Algorithm 1, all assignments of π_1 and π_2 will be included in a truth table after this write operation is performed at t_0, as shown in Figure 5 a). Firing t_0 generates four new states, i.e., $c_1 = \langle p_1, \{id_3, lpn = \perp, copy = \perp\}, \{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}, \{\pi_1, \pi_2, \perp, \perp, \perp, \perp\}\rangle$, $c_2 = \langle p_1,$ $\{id, lpn = \perp, copy = \perp\}, \{(id1, lpn1, copy1), (id2, lpn2, copy2)\}, \{\neg\pi_1, \pi_2, \perp, \perp, \perp,$

$\perp\}\rangle$, $c_3 = \langle p_1, \{id_3\ , lpn = \perp, copy = \perp\}, \{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}, \{\pi_1, \neg\pi_2, \perp, \perp, \perp, \perp\}\rangle$, and $c_4 = \langle p_1, \{id, lpn = \perp, copy = \perp\}, \{(id1, lpn1, copy1), (id2, lpn2, copy2)\}, \{\neg\pi_1, \neg\pi_2, \perp, \perp, \perp, \perp\}\rangle$. Similarly, all assignments of π_3 and π_4, and π_5 and π_6 are included in their corresponding truth tables, as shown in Figures 5 b) and 5 c), respectively. Figure 5 d) shows data information in an updatad table, and $id3$ is an updatad data item. Figure 5 e) shows a state reachability graph of the WFT-net in Figure 4, and pseudo states are represented by a dotted line. The state information in the state reachability graph is shown in Table 1, where θ_D is the value of $\{id, lpn, copy\}$, ϑ_R is the records of table $User$, and σ is one assignment of guard functions $\{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6\}$ ($T = true$, $F = false$).

C	m	θ_D	ϑ_R	σ
C_0	p_0	$\{\perp, \perp, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = \perp$, $\pi_2 = \perp$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_1	p_1	$\{id3, \perp, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_2	p_1	$\{id, \perp, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_3	p_1	$\{id3, \perp, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_4	p_1	$\{id, \perp, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_5	p_3	$\{id3, \perp, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp, p4 = \perp, p5 = \perp, p6 = \perp$
C_6	p_{10}	$\{id3, \perp, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp\pi_4 = \perp$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_7	p_5	$\{id3, \perp, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_8	p_7	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = F$, $\pi_6 = F$
C_9	p_7	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = F$, $\pi_6 = T$
C_{10}	p_7	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = T$, $\pi_6 = F$
C_{11}	p_7	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = T$, $\pi_6 = T$
C_{12}	p_6	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = F$, $\pi_6 = T$
C_{13}	p_8	$\{id3, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = F$, $\pi_6 = T$
C_{14}	p_9	$\{id3, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = F$, $\pi_6 = T$
C_{15}	p_{10}	$\{id3, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = F$, $\pi_6 = T$
C_{16}	p_6	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = T$, $\pi_6 = T$
C_{17}	p_8	$\{id3, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = T$, $\pi_6 = T$
C_{18}	p_9	$\{id3, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = T$, $\pi_6 = T$
C_{19}	p_{10}	$\{id3, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = T$, $\pi_6 = T$
C_{20}	p_2	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{21}	p_2	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{22}	p_2	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = F$, $\pi_5 = \perp$, $\pi_6 = \perp$

C_{23}	p_2	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = F$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{24}	p_{10}	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = F$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{25}	p_{10}	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{26}	p_4	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{27}	p_6	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{28}	p_8	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{29}	p_9	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{30}	p_{10}	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{31}	p_4	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{32}	p_6	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{33}	p_8	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{34}	p_9	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{35}	p_{10}	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = T$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{36}	p_2	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{37}	p_2	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = F$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{38}	p_2	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{39}	p_2	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = F$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{40}	p_{10}	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{41}	p_4	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{42}	p_6	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{43}	p_8	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{44}	p_9	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{45}	p_{10}	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{46}	p_{10}	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = T$, $\pi_4 = F$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{47}	p_4	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{48}	p_6	$\{id, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{49}	p_8	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{50}	p_9	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{51}	p_{10}	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1 = F$, $\pi_2 = T$, $\pi_3 = F$, $\pi_4 = T$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{52}	p_3	$\{id3, \perp, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T, \pi_2 = F, \pi_3 = \perp, \pi_4 = \perp, \pi_5 = \perp, \pi_6 = \perp$
C_{53}	p_{10}	$\{id3, \perp, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = \perp$, $\pi_6 = \perp$

C_{54}	p_5	$\{id3, \perp, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = \perp$, $\pi_6 = \perp$
C_{55}	p_7	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = T$, $\pi_6 = F$
C_{56}	p_7	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = F$, $\pi_6 = F$
C_{57}	p_7	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = F$, $\pi_6 = T$
C_{58}	p_6	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = F$, $\pi_6 = T$
C_{59}	p_8	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = F$, $\pi_6 = T$
C_{60}	p_9	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = F$, $\pi_6 = T$
C_{61}	p_{10}	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = F$, $\pi_6 = T$
C_{62}	p_7	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = T$, $\pi_6 = T$
C_{63}	p_6	$\{id3, lpn, \perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = T$, $\pi_6 = T$
C_{64}	p_8	$\{id3, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = T$, $\pi_6 = T$
C_{65}	p_9	$\{id3, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = T$, $\pi_6 = T$
C_{66}	p_{10}	$\{id3, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3, \perp, \perp)\}$	$\pi_1 = T$, $\pi_2 = F$, $\pi_3 = \perp$, $\pi_4 = \perp$, $\pi_5 = T$, $\pi_6 = T$

Table 1. Concrete states information in the state reachability graph

The pseudo states cannot exist in actual process model, as shown in Figure 5 e). According to Algorithm 3, when *id* is written at t_0, there is a constraint relation between guard functions π_1 and π_2. Given two proposition formulas $\omega_1 = (\neg\pi_1 \wedge \pi_2)$ and $\omega_2 = (\pi_1 \wedge \neg\pi_2)$, the constraint $true \models \omega_1 \vee \omega_2$ is calculated by Definition 2, and its truth table is shown in Figure 6 a). Then only transitions that satisfy *true* can be enabled. When t_0 is enabled at c_0, Algorithm 1 generates four states c_1, c_2, c_3 and c_4. In fact, guard functions in c_1 and c_4 do not satisfy the constraint *true*. Similarly, two constraints $true \models (\neg\pi_3 \wedge \pi_4) \vee (\pi_3 \wedge \neg\pi_4)$ and $true \models (\neg\pi_5 \wedge \pi_6) \vee (\pi_5 \wedge \neg\pi_6)$ are contructed and their truth tables are shown in Figure 6 b) and 6 c), respectively. Then the constraint set $res = \{(\neg\pi_1 \wedge \pi_2) \vee (\pi_1 \wedge \neg\pi_2), (\neg\pi_3 \wedge \pi_4) \vee (\pi_3 \wedge \neg\pi_4), (\neg\pi_5 \wedge \pi_6) \vee (\pi_5 \wedge \neg\pi_6)\}$ is constructed. Figure 6 d) shows the state reachability graph without pseudo states of the WFTC-net in Figure 4 generated by Algorithm 4. The state information is recorded in Table 2. The reachability graph without pseudo states is more in line with actual requirements when characterizing the dynamic behavior of this system.

6 TOOL AND EXPERIMENTS

Based on our algorithms, we develop a tool to generate the state reachability graph of a WFC-net, which is written in C++ programming language. After inputting a WFTC-net (.txt file) and a table (.txt file), our tool can read them, and generate state reachability graphs. Figure 7 a) describes an abstract file information of the WFT-net in Figure 7, where 7 b) shows a constraint set *res*, 7 c) represents an initial

π_1	π_2
0	0
1	0
0	1
1	1

a)

π_3	π_4
0	0
1	0
0	1
1	1

b)

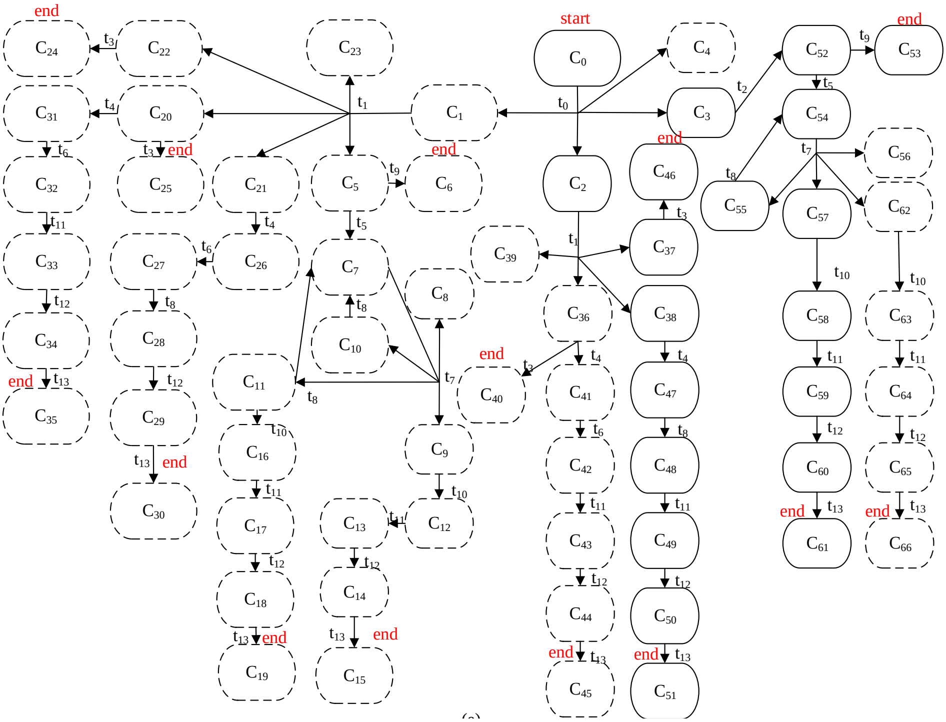
π_5	π_6
0	0
1	0
0	1
1	1

c)

User

Id	Lpn	Copy
Id1	Lpn1	Copy1
Id2	Lpn2	Copy2
Id3	$\perp$	$\perp$

d)



e)

Figure 5. a), b) and c) are the truth tables of the three sets of guard functions without constraints; d) An updatad table User; e) A state reachability graph of the WFT-net

table *user*, 7 d) is a state reachability graph of WFT-net, and 7 e) shows a state reachability graph of WFTC-net.

The results show that the WFT-net in Figure 4 produces a total of 91 states and 146 state arcs, while WFTC-net produces a total of 63 states and 82 state arcs. The main reason is that WFT-net does not consider the constraint relation between guard functions and generates pseudo states while our method overcomes this problem. In order to further show the effectiveness of our tool, we do the experiments on 10 different examples, and their results are shown in Table 3. All

C	m	θ_D	ϑ_R	σ
C_0	p_0	$\{\perp,\perp,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1=\perp, \pi_2=\perp, \pi_3=\perp, \pi_4=\perp, \pi_5=\perp, \pi_6=\perp$
C_1	p_1	$\{id,\perp,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1=F, \pi_2=T, \pi_3=\perp, \pi_4=\perp, \pi_5=\perp, \pi_6=\perp$
C_2	p_1	$\{id3,\perp,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3,\perp,\perp)\}$	$\pi_1=T, \pi_2=F, \pi_3=\perp, \pi_4=\perp, \pi_5=\perp, \pi_6=\perp$
C_3	p_2	$\{id, lpn,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1=F, \pi_2=T, \pi_3=T, \pi_4=F, \pi_5=\perp, \pi_6=\perp$
C_4	p_2	$\{id, lpn,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1=F, \pi_2=T, \pi_3=F, \pi_4=T, \pi_5=\perp, \pi_6=\perp$
C_5	p_{10}	$\{id, lpn,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1=F, \pi_2=T, \pi_3=T, \pi_4=F, \pi_5=\perp, \pi_6=\perp$
C_6	p_4	$\{id, lpn,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1=F, \pi_2=T, \pi_3=F, \pi_4=T, \pi_5=\perp, \pi_6=\perp$
C_7	p_6	$\{id, lpn,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1=F, \pi_2=T, \pi_3=F, \pi_4=T, \pi_5=\perp, \pi_6=\perp$
C_8	p_8	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1=F, \pi_2=T, \pi_3=F, \pi_4=T, \pi_5=\perp, \pi_6=\perp$
C_9	p_9	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1=F, \pi_2=T, \pi_3=F, \pi_4=T, \pi_5=\perp, \pi_6=\perp$
C_10	p_{10}	$\{id, lpn, copy\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2)\}$	$\pi_1=F, \pi_2=T, \pi_3=F, \pi_4=T, \pi_5=\perp, \pi_6=\perp$
C_{11}	p_3	$\{id3,\perp,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3,\perp,\perp)\}$	$\pi_1=T, \pi_2=F, \pi_3=\perp, \pi_4=\perp, \pi_5=\perp, \pi_6=\perp$
C_{12}	p_{10}	$\{id3,\perp,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3,\perp,\perp)\}$	$\pi_1=T, \pi_2=F, \pi_3=\perp, \pi_4=\perp, \pi_5=\perp, \pi_6=\perp$
C_{13}	p_5	$\{id3,\perp,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3,\perp,\perp)\}$	$\pi_1=T, \pi_2=F, \pi_3=\perp, \pi_4=\perp, \pi_5=\perp, \pi_6=\perp$
C_{14}	p_7	$\{id3, lpn,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3,\perp,\perp)\}$	$\pi_1=T, \pi_2=F, \pi_3=\perp, \pi_4=\perp, \pi_5=T, \pi_6=F$
C_{15}	p_7	$\{id3, lpn,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3,\perp,\perp)\}$	$\pi_1=T, \pi_2=F, \pi_3=\perp, \pi_4=\perp, \pi_5=F, \pi_6=T$
C_{16}	p_6	$\{id3, lpn,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3,\perp,\perp)\}$	$\pi_1=T, \pi_2=F, \pi_3=\perp, \pi_4=\perp, \pi_5=F, \pi_6=T$
C_{17}	p_8	$\{id3, lpn,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3,\perp,\perp)\}$	$\pi_1=T, \pi_2=F, \pi_3=\perp, \pi_4=\perp, \pi_5=F, \pi_6=T$
C_{18}	p_9	$\{id3, lpn,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3,\perp,\perp)\}$	$\pi_1=T, \pi_2=F, \pi_3=\perp, \pi_4=\perp, \pi_5=F, \pi_6=T$
C_{19}	p_{10}	$\{id3, lpn,\perp\}$	$\{(id1, lpn1, copy1), (id2, lpn2, copy2), (id3,\perp,\perp)\}$	$\pi_1=T, \pi_2=F, \pi_3=\perp, \pi_4=\perp, \pi_5=F, \pi_6=T$

Table 2. Concrete states information in the state reachability graph

Model	WFT-NET								WFTC-RG			WFT-RG			
	No. of Guards	No. of Data Operations			No. of Table Operations				No. of States	No. of Arcs	Time	No. of States	No. of Arcs	Time	No. of Pseudo States
		wt	rd	dt	sel	ins	del	upd							
1	2	3	8	1	6	1	1	2	63	82	8.348	91	146	9.704	28
2	4	3	8	1	6	1	1	2	66	80	10.791	91	140	16.732	25
3	6	3	8	1	7	1	1	1	39	43	8.918	86	145	12.786	47
4	0	0	0	0	0	0	0	0	11	14	7.46	11	14	8.448	0
5	0	2	7	1	6	1	1	2	66	79	8.58	128	155	11.49	62
6	4	3	8	1	7	1	1	2	30	36	8.818	178	304	9.399	148
7	6	3	8	0	5	2	1	3	29	34	34.866	126	210	38.164	97
8	2	3	9	1	7	1	1	3	34	38	8.906	92	118	9.683	58
9	3	2	10	0	6	1	1	2	41	51	8.152	99	163	9.774	58
10	4	2	11	0	7	2	1	2	50	56	8.688	72	83	9.414	16

Table 3. The test results

π_1	π_2
1	0
0	1

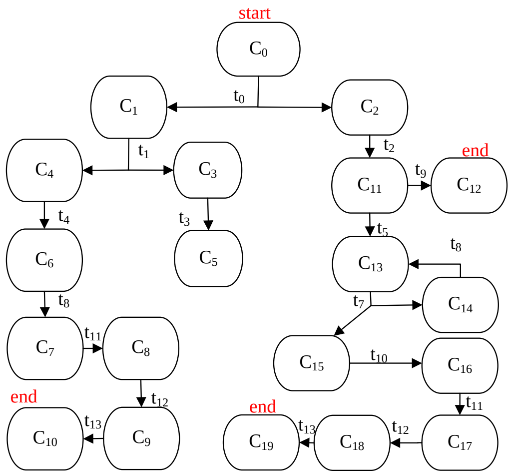
a)

π_3	π_4
1	0
0	1

b)

π_5	π_6
1	0
0	1

c)



d)

Figure 6. a), b) and c) are the truth tables of the three constraints; d) A state reachability graph of the WFTC-net

the experiments are done on the Intel Core I5-8500 CPU (3.00 GHz) and 8.0 GB memory.

The results in Table 3 show that our method can effetively reduce pseudo states when producing the state reachability graph of a WFT-net. When multiple guard functions are operating on the same data item, WFT-net lacks consideration of the constraint relation between guard functions. Thus, it is easy to generate pseudo states in its state reachability graph. WFTC-net considers this constraint relation, and thus pseudo states can be avoided. Naturally, the operating behaviors of the system can be described more accurately. Obviously, the state reachability graph of WFTC-net (WFTC-RG) spends less time in comparison with the corresponding WFT-RG.

Additionally, in order to study the influence of *user* numbers on the state reachability graph of a WFT-net, we choose the models of group 1 and group 10 in Table 3 to do the experiments. The results are shown in Table 4, where x/y means the model of x users and group y.

As shown in the experimental results in Table 4, when the user information in the table increases gradually, the number of states generated in the state reachability graph of WFT-net and WFTC-net also increases. With the increase of data items in the table, the operations on the data in the table will also increase. Thus, the state will gradually increase in the process of generating the state reachability graph. Obviously, WFTC-RG spends less time in comparison with the corresponding WFT-RG, which further illustrates the superiority of our method.

transition	preset	postset	guard	dataoperation	dataschema
t0	0.	1.	1,0.	wt,id.	sel,id.
t1	1.	2.	1,2.	rd,id.	sel,id.
t2	1.	3.	0,1.	rd,id3.	ins,id3.
t3	2.	10.	0,0.	rd,lpn.	sel,lpn.
t4	2.	4.	0,0.	rd,lpn.	sel,lpn.
t5	3.	5.	0,0.	rd,id3.	.
t6	4.	6.	0,0.	rd,id1.	upd,lpn11.
t7	5.	7.	0,0.	wt,lpn3.	sel,id3.
t8	7.	5.	0,0.	dt,id.	sel,id.
t9	3.	10.	0,0.	rd,id.	.
t10	7.	6.	0,0.	wt,license.	upd,license.
t11	6.	8.	0,0.	wt,license3.	sel,license3.
t12	8.	9.	0,0.	dt,license3.	sel,license3.
t13	9.	10.	0,0.	rd,license3.	upd,copy3.
@					
place	name	token			
0	p0	1			
1	p1	0			
2	p2	0			
3	p3	0			
4	p4	0			
5	p5	0			
6	p6	0			
7	p7	0			
8	p8	0			
9	p9	0			
10	p10	0			
@					
pipnr	piname	pidata	piguard		
1	pi1	id1,id2.	1,2.		
2	pi2	lpn1,lpn2.	3,4.		
3	pi3	copy1,copy2.	5,6.		
@					

a)

gpnr	goperator	gformer	glatter
1	pi1	0	0
*			
1	pi1	0	0
2	not	0	1
*			
1	pi2	0	0
*			
1	pi2	0	0
2	not	0	1
*			
1	pi3	0	0
*			
1	pi3	0	0
2	not	0	1
@			
cpnr	coperator	cformer	clatter
1	g1	0	0
2	g2	0	0
3	not	0	1
4	not	0	2
5	and	3	2
6	and	1	4
7	or	5	6
*			
1	g3	0	0
2	g4	0	0
3	not	0	1
4	not	0	2
5	and	3	2
6	and	1	4
7	or	5	6
*			
1	g5	0	0
2	g6	0	0
3	not	0	1
4	not	0	2
5	and	3	2
6	and	1	4
7	or	5	6
@			

b)

User	Id	Lpn	Copy
0	id1	lpn1	copy1
1	id2	lpn2	copy2
@			

c)

```
"C:\Users\SJ\Desktop\first paper tool\Debug\WFT-RG.e...
please enter USEfile name: user2.txt
please enter WFTfile name:wft1.txt
1:Insert a new element,2:STOP!
Input: 1

The Net with Constraints produces 91 states!
The Net with Constraints generates 146 state arcs!
The total run time is:11.076s
Press any key to continue
```

d)

```
"C:\Users\SJ\Desktop\first paper tool\Debug\WFTC-RG....
please enter USEfile name: user2.txt
please enter WFTfile name:wft1.txt
1:Insert a new element,2:STOP!
Input: 1

The Net with Constraints produces 63 states!
The Net with Constraints generates 82 state arcs!
The total run time is:9.587s
Press any key to continue
```

e)

Figure 7. a) A WFT-net; b) A constraint set *res*; c) An initial table *user*; d) A state reachability graph of WFT-net; e) A state reachability graph of WFTC-net

User/ Model	WFT-NET								WFTC-RG			WFT-RG			
	No. of Guards	No. of Data Operations			No. of Table Operations				No. of States	No. of Arcs	Time	No. of States	No. of Arcs	Time	No. of Pseudo States
		wt	rd	dt	sel	ins	del	upd							
1/1	2	3	8	1	6	1	1	2	43	52	9.037	55	77	12.843	34
2/1									63	82	8.348	91	146	9.704	28
3/1									79	122	9.192	123	255	10.79	44
4/1									100	173	9.295	160	403	10.476	60
5/1									120	231	9.504	196	582	16.533	76
6/1									140	297	11.509	232	793	15.136	92
1/10	4	2	11	0	7	2	1	2	32	34	9.021	45	50	11.172	13
2/10									50	56	8.688	72	83	9.414	16
3/10									58	65	8.493	89	101	9.919	31
4/10									75	84	9.665	115	130	10.311	40
5/10									92	103	9.211	141	159	9.486	49
6/10									109	122	10.149	167	188	13.945	58

Table 4. The test results

To further illustrate that our constraints also apply to WFD-net, we do a set of experiments in order to compare WFD-RG and WFDC-RG in terms of state space and construction time. The following benchmarks are used:

- B1 is an example with two threads accessing two shared variables, then it produces concurrency bugs [34].
- B2 is a classic algorithm for solving the mutual exclusion problem in concurrent systems [35].
- B3 is a concurrent program, where multiple concurrent threads manipulate a shared hash table [36].
- B4 is a system-level modeling language that offers a wide range of features to describe concurrent systems at different levels of abstraction [37].
- B5 is a tutorial program to detect and fix data races [38].
- B6 is a sequence of instructions where any branch is at the end and there are shared variables access [39].
- B7 is a simple producer-consumer example. It is a set of interconnected modules communicating through channels using transactions, events and shared variables [40].
- B8 is a test driver for a simplified version of a Bluetooth driver [41].

For each benchmark, we first use WFD-net to model it, and use our tool to obtain their WFD-RG and WFDC-RG, respectively. Each benchmark tested 10 times, and the result of running time is their averages. Table 5 is the results of our experiments. It shows the number of states, the number of arcs, and the construction time of WFD-RG and WFDC-RG for all benchmarks. From this table, we can see that the scale of WFDC-RG is much smaller than WFD-RG. Obviously, it spends less time to produce a WFDC-RG in comparison with the corresponding WFD-RG.

Benchmarks	WFD-nets					WFD-RG			WFDC-RG		
	$\lvert T\rvert$	$\lvert P\rvert$	$\lvert F\rvert$	$\lvert D\rvert$	$\lvert G\rvert$	Nos. of States	Nos. of Arcs	Time (s)	Nos. of States	Nos. of Arcs	Time (s)
BM1	17	17	35	8	4	410	665	0.802	201	325	0.462
BM2	25	27	50	3	8	183	405	0.924	76	148	0.377
BM3	11	9	28	6	6	83	82	0.472	17	18	0.302
BM4	18	17	40	9	4	1057	3201	1.681	417	1281	0.613
BM5	22	21	47	10	6	95	128	0.589	25	30	0.435
BM6	22	24	48	10	6	96	97	0.482	27	29	0.415
BM7	41	37	84	18	6	11385	36593	39.664	1905	6357	6.037
BM8	57	53	122	15	16	11174	24313	24.09	928	2167	2.045

Table 5. The test results

7 CONCLUSION

WFT-net is used to simulate a workflow system that operates on data tables, and the state reachability graph is defined to describe all its possible running behaviors. In fact, the data refinement method of WFT-net proposed in [27] has shortcomings due to its lacks of constraint relations between guard functions. Thus, it is easy to generate pseudo states (or illegal states) in the state reachability graph of WFT-net. In this paper, we propose an improved data refinement method. In our guard-driven state reachability graph, the constraint relation between guard functions are considered so that the running behaviors of workflow systems can be expressed more accurately. This paper also proposes some algorithms for generating reachability graphs for WFTC-net. Futhermore, we develop a modeling and analysis tool. The case study and experiments show the usefulness and effectiveness of our methods. In the future, we will consider the application of first-order computation tree logic (CTL) in the state reachability graph of WFTC-net, we want to verify whether there are logic errors in WFTC-net, and develop a new model checking tool based on WFTC-net.

Acknowledgement

This paper is supported in part by the National Nature Science Foundation of China (No. 62172299).

REFERENCES

[1] Reichert, M.: Process and Data: Two Sides of the Same Coin? In: Meersman, R. et al. (Eds.): On the Move to Meaningful Internet Systems (OTM 2012). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7565, 2012, pp. 2–19, doi: 10.1007/978-3-642-33606-5_2.

[2] Weidlich, M.—Polyvyanyy, A.—Desai, N.—Mendling, J.: Process Compliance Measurement Based on Behavioural Profiles. In: Pernici, B. (Ed.): Advanced Information Systems Engineering (CAiSE 2010). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6051, 2010, pp. 499–514, doi: 10.1007/978-3-642-13094-6_38.

[3] Dolean, C. C.—Petrusel, R.: Data-Flow Modeling: A Survey of Issues and Approaches. Informatica Economica, Vol. 16, 2012, No. 4, pp. 117–130.

[4] Castro, L. M.—Arts, T.: Testing Data Consistency of Data-Intensive Applications Using QuickCheck. Electronic Notes in Theoretical Computer Science, Vol. 271, 2011, pp. 41–62, doi: 10.1016/j.entcs.2011.02.010.

[5] Fu, X.—Wang, F.—Liu, X.—Ji, K.—Zou, P.: Dataflow Weaknesses Analysis of Scientific Workflow Based on Fault Tree. 2012 Sixth International Symposium on Theoretical Aspects of Software Engineering, IEEE, 2012, pp. 227–230, doi: 10.1109/TASE.2012.18.

[6] Sadiq, S.—Orlowska, M.—Sadiq, W.—Foulger, C.: Data Flow and Validation in Workflow Modelling. Proceedings of the 15th Australasian Database Conference – Volume 27 (ADC 2004), 2004, pp. 207–214.

[7] Sun, S. X.—Zhao, J. L.—Nunamaker, J. F.—Sheng, O. R. L.: Formulating the Data-Flow Perspective for Business Process Management. Information Systems Research, Vol. 17, 2006, No. 4, pp. 374–391, doi: 10.1287/isre.1060.0105.

[8] Wang, J.—Rosca, D.—Tepfenhart, W.—Milewski, A.—Stoute, M.: Dynamic Workflow Modeling and Analysis in Incident Command Systems. IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, Vol. 38, 2008, No. 5, pp. 1041–1055, doi: 10.1109/TSMCA.2008.2001080.

[9] He, C.—Ding, Z.: More Efficient On-the-Fly Verification Methods of Colored Petri Nets. Computing and Informatics, Vol. 40, 2021, No. 1, pp. 195–215, doi: 10.31577/cai_2021_1_195.

[10] Ding, Z.—Liu, J.—Wang, J.—Wang, F.: An Executable Service Composition Code Automatic Creation Tool Based on Petri Net Model. Computing and Informatics, Vol. 32, 2013, No. 5, pp. 968–986.

[11] Jiang, F. C.—Hsu, C. H.—Wang, S.: Logistic Support Architecture with Petri Net Design in Cloud Environment for Services and Profit Optimization. IEEE Transactions on Services Computing, Vol. 10, 2017, No. 6, pp. 879–888, doi: 10.1109/tsc.2016.2514506.

[12] Moutinho, F.—Gomes, L.: Asynchronous-Channels Within Petri Net-Based GALS Distributed Embedded Systems Modeling. IEEE Transactions on Industrial Informatics, Vol. 10, 2014, No. 4, pp. 2024–2033, doi: 10.1109/tii.2014.2341933.

[13] Wang, J.: Emergency Healthcare Workflow Modeling and Timeliness Analysis. IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, Vol. 42, 2012, No. 6, pp. 1323–1331, doi: 10.1109/TSMCA.2012.2210206.

[14] Xiang, D.—Liu, G.—Yan, C.—Jiang, C.: Detecting Data Inconsistency Based on the Unfolding Technique of Petri Nets. IEEE Transactions on Industrial Informatics, Vol. 13, 2017, No. 6, pp. 2995–3005, doi: 10.1109/tii.2017.2698640.

[15] Clarke, E.: Counterexample-Guided Abstraction Refinement. 10th International

Symposium on Temporal Representation and Reasoning, 2003 and Fourth International Conference on Temporal Logic, 2003, pp. 7–8, doi: 10.1109/time.2003.1214874.

[16] Trčka, N.—Van der Aalst, W. M. P.—Sidorova, N.: Data-Flow Anti-Patterns: Discovering Data-Flow Errors in Workflows. In: van Eck, P., Gordijn, J., Wieringa, R. (Eds.): Advanced Information Systems Engineering (CAiSE 2009). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5565, 2009, pp. 425–439, doi: 10.1007/978-3-642-02144-2_34.

[17] Meda, H. S.—Sen, A. K.—Bagchi, A.: On Detecting Data Flow Errors in Workflows. Journal of Data and Information Quality (JDIQ), Vol. 2, 2010, No. 1, Art. No. 4, doi: 10.1145/1805286.1805290.

[18] von Stackelberg, S.—Putze, S.—Mülle, J.—Böhm, K.: Detecting Data-Flow Errors in BPMN 2.0. Open Journal of Information Systems (OJIS), Vol. 1, 2014, No. 2, pp. 1–19.

[19] Trčka, N.: Workflow Data Footprints. In: Abramowicz, W., Tolksdorf, R. (Eds.): Business Information Systems (BIS 2010). Springer, Berlin, Heidelberg, Lecture Notes in Business Information Processing, Vol. 47, 2010, pp. 218–229, doi: 10.1007/978-3-642-12814-1_19.

[20] Smith, G.—Derrick, J.: Verifying Data Refinements Using a Model Checker. Formal Aspects of Computing, Vol. 18, 2006, No. 3, pp. 264–287, doi: 10.1007/s00165-006-0002-7.

[21] Ge, J.—Hu, H.—Lu, J.: Invariant Analysis for the Task Refinement of Workflow Nets. 2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA '06), IEEE, 2006, pp. 209–209, doi: 10.1109/CIMCA.2006.135.

[22] Gardiner, P. H. B.—Morgan, C.: A Single Complete Rule for Data Refinement. Formal Aspects of Computing, Vol. 5, 1993, No. 4, pp. 367–382, doi: 10.1007/BF01212407.

[23] Majster-Cederbaum, M.—Wu, J.—Yue, H.: Refinement of Actions for Real-Time Concurrent Systems with Causal Ambiguity. Acta Informatica, Vol. 42, 2006, No. 6-7, pp. 389–418, doi: 10.1007/s00236-005-0172-4.

[24] Sidorova, N.—Stahl, C.—Trčka, N.: Workflow Soundness Revisited: Checking Correctness in the Presence of Data While Staying Conceptual. In: Pernici, B. (Ed.): Advanced Information Systems Engineering (CAiSE 2010). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6051, 2010, pp. 530–544, doi: 10.1007/978-3-642-13094-6_40.

[25] van der Aalst, W. M. P.—van Hee, K. M.—ter Hofstede, A. H. M.—Sidorova, N.—Verbeek, H. M. W.—Voorhoeve, M.—Wynn, M. T.: Soundness of Workflow Nets: Classification, Decidability, and Analysis. Formal Aspects of Computing, Vol. 23, 2011, No. 3, pp. 333–363, doi: 10.1007/s00165-010-0161-4.

[26] Xiang, D.—Liu, G.: Checking Data-Flow Errors Based on the Guard-Driven Reachability Graph of WFD-Net. Computing and Informatics, Vol. 39, 2020, No. 1-2, pp. 193–212, doi: 10.31577/cai_2020_1-2_193.

[27] Tao, X.—Liu, G.—Yang, B.—Yan, C.—Jiang, C.: Workflow Nets with Tables

and Their Soundness. IEEE Transactions on Industrial Informatics, Vol. 16, 2020, No. 3, pp. 1503–1515, doi: 10.1109/TII.2019.2949591.

[28] van der Aalst, W. M. P.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems and Computers, Vol. 8, 1998, No. 1, pp. 21–66, doi: 10.1142/S0218126698000043.

[29] Zhou, G. F.—Du, Z. M.: Petri Nets Model of Implicit Data and Control in Program Code. Ruanjian Xuebao/Journal of Software, Vol. 22, 2011, No. 12, pp. 2905–2918, doi: 10.3724/sp.j.1001.2011.03956 (in Chinese).

[30] Li, J.—Yang, R.—Ding, Z.—Pan, M.: A Method for Learning a Petri Net Model Based on Region Theory. Computing and Informatics, Vol. 39, 2020, No. 1-2, pp. 174–192, doi: 10.31577/cai_2020_1-2_174.

[31] Wang, J.—Li, D.: Resource Oriented Workflow Nets and Workflow Resource Requirement Analysis. International Journal of Software Engineering and Knowledge Engineering, Vol. 23, 2013, No. 5, pp. 677–693, doi: 10.1142/S0218194013400135.

[32] Bai, E.—Su, N.—Liang, Y.—Qi, L.—Du, Y.: Method for Repairing Process Models with Selection Structures Based on Token Replay. Computing and Informatics, Vol. 40, 2021, No. 2, pp. 446–468, doi: 10.31577/cai_2021_2_446.

[33] Silberschatz, A.—Korth, H. F.—Sudarshan, S.: Database System Concepts. 3rd Edition. McGraw-Hill New York, 1997.

[34] Huang, J.—Zhang, C.—Dolby, J.: CLAP: Recording Local Executions to Reproduce Concurrency Failures. ACM SIGPLAN Notices, Vol. 48, 2013, No. 6, pp. 141–152, doi: 10.1145/2499370.2462167.

[35] Burnim, J.—Sen, K.—Stergiou, C.: Testing Concurrent Programs on Relaxed Memory Models. Proceedings of the 2011 International Symposium on Software Testing and Analysis (ISSTA '11), 2011, pp. 122–132, doi: 10.1145/2001420.2001436.

[36] Flanagan, C.—Godefroid, P.: Dynamic Partial-Order Reduction for Model Checking Software. ACM SIGPLAN Notices, Vol. 40, 2005, No. 1, pp. 110–121, doi: 10.1145/1047659.1040315.

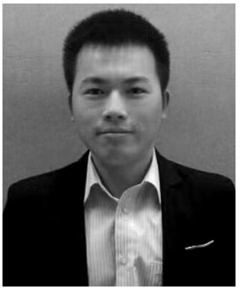
[37] Blanc, N.—Kroening, D.: Race Analysis for SystemC Using Model Checking. ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 15, 2010, No. 3, Art. No. 21, doi: 10.1145/1754405.1754406.

[38] Oracle: Sun Studio 12: Thread Analyzer User's Guide.

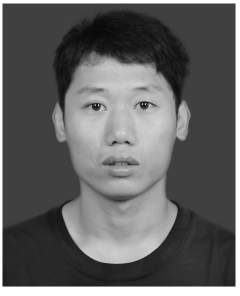
[39] Lee, J.—Padua, D. A.—Midkiff, S. P.: Basic Compiler Algorithms for Parallel Programs. ACM SIGPLAN Notices, Vol. 34, 1999, No. 8, pp. 1–12, doi: 10.1145/329366.301105.

[40] Kundu, S.—Ganai, M.—Gupta, R.: Partial Order Reduction for Scalable Testing of SystemC TLM Designs. 2008 45th ACM/IEEE Design Automation Conference, IEEE, 2008, pp. 936–941, doi: 10.1145/1391469.1391706.

[41] Razavi, N.—Ivančić, F.—Kahlon, V.—Gupta, A.: Concurrent Test Generation Using Concolic Multi-Trace Analysis. In: Jhala, R., Igarashi, A. (Eds.): Programming Languages and Systems (APLAS 2012). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7705, 2012, pp. 239–255, doi: 10.1007/978-3-642-35182-2_17.



Song Jian received his M.Sc. degree from the School of Mechanics and Optoelectronics Physics, Anhui University of Science and Technology, Huainan, China, in 2019. He is currently working toward the Ph.D. degree in the Department of Computer Science and Technology, School of Electronics and Information Engineering, Tongji University, Shanghai, China. His current research interests include model checking, Petri net, workflow, control-flow, data-flow.



Dongming Xiang received his Ph.D. degree in computer science and technology from the Tongji University, Shanghai, China, in 2018. He is currently Lecturer with the Department of Computer Science and Technology, Zhejiang Sci-Tech University, Hangzhou, China. His research interests include model checking, Petri net, software verification, business process management, and service computing.

Guanjun Liu received his Ph.D. degree in computer software and theory from the Tongji University, Shanghai, China, in 2011. He was Post-Doctoral Research Fellow with the Singapore University of Technology and Design, Singapore, from 2011 to 2013. He was Post-Doctoral Research Fellow with the Humboldt University of Berlin, Germany, from 2013 to 2014, funded by the Alexander von Humboldt Foundation. In 2013, he joined the Department of Computer Science of Tongji University as Associate Professor, and now is Professor. His research interests include Petri net theory, model checking, Web service, workflow, discrete event systems, machine learning and credit card fraud detection.

Leifeng He received his B.Sc. degree in science and technology from the University of Shanghai, Shanghai, China, in 2015. He is currently working toward his Ph.D. degree in the Department of Computer Science and Technology, School of Electronics and Information Engineering, Tongji University, Shanghai, China. His research interests include model checking, Petri net, workflow, multi-agent systems, and epistemic logic.