Computing and Informatics, Vol. 43, 2024, 1320–1351, doi: 10.31577/cai_2024_6_1320

MODELING AND ANALYSIS OF BUSINESS PROCESS MANAGEMENT SYSTEMS USING TIMED WORKFLOW NETS WITH TABLES

Jian Song, Guanjun Liu*

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

Abstract. The existing modeling methods for business process management systems (BPMS) focus on the logical and abstract data layers but often ignore operations related to the underlying database, thus failing to describe system behavior accurately. Workflow nets with tables (WFT-nets) compensate for the lack of description of database table operations in existing modeling methods. However, for timed business process management systems (TBPMS), their correct behavior depends on the logical correctness of the results obtained by the operational process and the time required for each activity to be correctly executed within a specified time. Since WFT-nets do not consider time properties, they are unable to describe time-related activities in TBPMS, potentially leading to incorrect results. This paper introduces timed workflow nets with tables (TWFT-nets), which add time elements to transitions to simulate time-constrained activities in the system. Additionally, we assign different labels to represent the execution strategies of activities under different time constraints. To analyze the soundness of TWFT-nets, we propose a timed database computation tree logic (TDCTL) model checking method and define soundness from three perspectives: logic control layer, data design requirements, and time constraints. We transform soundness into TDCTL formulas, provide a model checking algorithm, and develop a tool. Experiments show the effectiveness of our methods.

Keywords: BPMS, TDCTL, model checking, soundness, TWFT-net

Mathematics Subject Classification 2010: 68-Q60

* Corresponding author

1 INTRODUCTION

Business process management systems (BPMS) are increasingly crucial in real-life applications and industrial production, such as smart healthcare systems and intelligent rail transportation systems [1, 2]. Due to these systems' complex business logic and extensive data operations, rule violations or unreasonable designs can lead to security vulnerabilities or financial losses. Therefore, it is important to model and analyse the system to verify its correctness. Ensuring such systems' security and high reliability is a current research hotspot [3, 4, 5].

Formal methods are a mathematical and logic-based approach widely applied in system development, where properties such as correctness and security require formal modeling, analysis, and verification. As a formal language, Petri nets are widely used in the modeling and analysis of BPMS, leveraging their structural characteristics to uncover errors within systems [6, 7, 8]. Sidorova et al. [9, 10] proposed workflow nets with data (WFD-nets), which combine traditional workflow nets with abstract data operation to describe the data flow aspect of the BPMS. However, when data flow operations are introduced into the business process model, there are a lot of data interactions between activities, which make it easy to produce data flow errors. To detect the defects caused by data in business process design, Combi et al. [11, 12, 13] proposed a modeling approach that integrates business processes with data. This method involves constructing activity diagrams to capture the data operations performed by activities, providing an analysis method for BPMS.

The design requirements validation of workflow models is typically performed using model checking techniques on their state reachability graph (SRG) [14, 15]. Sbaï et al. [16] transformed the system design requirements to be verified into an LTL formula and performed verification on the constructed Petri nets model. To verify the initialization of hardware devices, Mönnich et al. [17] abstractly modeled the execution planning of devices as a Petri net and condition abstraction as temporal logic, utilizing the NuSMV tool for verification. Trčka et al. [18, 19] proposed using WFD-net to model BPMS, describing various data flow errors through temporal logic and employing model checking techniques to identify data flow errors in the business process model. With the development of a timed business process management system (TBPMS), the correctness of the system is not only related to logic behavior in the process but also to time constraints. Therefore, researchers have proposed formal methods for TBPMS, such as time Petri net [20, 21, 22] and timed automata [23, 24].

The activities in actual BPMS are about the data item operation in the underlying database table (select, insert, delete, and update). Tao et al. [25] proposed a model called workflow nets with tables (WFT-nets) to satisfy the modeling requirements for such systems. WFT-nets can describe concrete data item values in database tables and represent additional behavioral information. However, with the continuous application and development of TBPMS, the system has higher requirements for real time [26, 27]. WFT-nets lack consideration of time constraints, so they can not accurately describe the operation behavior of each activity in the TBPMS.

To precisely characterize TBPMS, we have extended WFT-nets by attaching time elements to transitions, resulting in timed workflow nets with tables (TWFT-nets) model. TWFT-nets retain the advantages of traditional modeling methods in control flow and data flow and provide a detailed representation of the operation of the concrete data item value in database tables. Adding the time elements to transitions makes the model more closely related to real systems, thus facilitating convenient analysis of actual systems. To validate the soundness of TWFT-nets, we propose a model checking method of timed database computation tree logic. We define soundness from three perspectives: logic control layer, data design requirements, and time constraints. We translate soundness into corresponding TDCTL formulas, thereby enabling the detection and prevention of potential adverse behaviors in the system. We propose the related model checking algorithms and develope a tool based on TWFT-nets. Experiments and case study show the advantages and usefulness of our method and tool.

This paper is organized as follows. Section 2 reviews related work. Section 3 presents some basic notations and gives a motivating example. Section 4 defines TWFT-nets and their firing rules and shows their state reachability graph generation algorithms. Section 5 introduces timed database model checking methods and defines their soundness. Section 6 provides a group of experiments to demonstrate our methods' effectiveness. Section 7 concludes this paper.

2 RELATED WORK

Formal modeling methods have been widely applied in analyzing and validating BPMS [28, 29, 30]. Most researchers focus on the soundness validation of business process models [31, 32]. This section introduces some conceptual models and validation techniques to analyze business process models.

2.1 Modeling Conceptual of BPMS

In the early research, Van der Aalst [33] and Morimoto [34] proposed to use WF-nets to model BPMS and describe the control flow relationship between activities. With the development of BPMS, many activities within the system became associated with data flow. Sidorova et al. [9, 18] proposed a modeling method for WFD-nets. This method extends WF-nets with abstract data operations (read/write/delete) to describe the operations related to data elements in the system. Time Petri nets and timed automata have been proposed to describe time-related activities in business processes and the interaction between activities [35, 36]. As the complexity of business processes increases, each activity has various attributes and colored Petri nets can further model the attributes of activities and improve the modeling ability [37, 38]. The underlying activity of the BPMS are about the background database operation. Tao et al. [25] proposed a modeling method named WFT-net, which can simulate the operation of data elements in the database and more accurately describe the operation behavior of activities in the system.

The actual BPMS are related to control flow, data flow, and time attributes. Previous research in formalizing BPMS focused on control and data flow or emphasized time attributes. None of the existing methods integrated all three aspects into the modeling process. The TWFT-net modeling method we propose effectively incorporates these attributes into the model. The TWFT-net modeling method effectively incorporates these attributes into the model.

2.2 Model Verification

Model soundness verification is a current research hotspot. To detect data flow errors in business process models, Song et al. [39] proposed three crucial criteria that maintain data flow correctness in process adaptation. To ensure that business process models do not lead to livelock and deadlock, Sidorova et al. [9, 40] proposed a soundness verification method, which guarantees the soundness of business process models at the control flow level. Considering time constraints in process models, Liu et al. [41] proposed a logical time workflow nets modeling method, which can solve the problem of logical defects in cross-organizational workflow nets. Model checking is also one of the ways to verify the soundness of business process models. Researchers need to transform design requirements into corresponding logical formulas and then verify the correctness of these formulas on the state space [42, 10]. Stackelberg et al. [13] proposed an approach for detecting data-flow errors in BPMN 2.0 process models. They defined a set of anti-patterns representing data-flow errors in BPMN 2.0 process models. Utilizing model checking techniques to validate data flow errors enhances the correctness of the models.

The existing soundness verification methods are mainly based on control flow or data flow without considering the values of data items related to database operations in the underlying design of business processes, and time elements have not been taken into account in the soundness verification. Therefore, our proposed TCTL model checking method considers the soundness of the model from three design requirements: control flow, data flow, and time constraints. Our method further ensures the correctness of the model.

3 BASIC CONCEPTS AND MOTIVATING EXAMPLE

3.1 Basic Concepts

A net is a 3-tuple $N_0 = (P, T, F)$, 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 $m : P \to \mathbb{N}$, where $\mathbb{N}$ is the set of non-negative integers and $m(p)$ is the number of tokens in place p. For each node $x \in P \cup T$, its preset is denoted by ${}^\bullet x = \{y \in P \cup T | (y, x) \in F\}$, and its postset is denoted by $x^\bullet = \{y \in P \cup T | (x, y) \in F\}$.

Definition 1 (Petri net and firing rule [33, 43, 44])**.** A net N_0 with an *initial mark-*

ing m_0 is a *Petri net* and denoted as $PN = (N_0, m_0)$. If $\forall p \in P : p \geq 1$, then transition t is enabled at marking m, which is denoted by $m[t\rangle$. Firing an enabled transition t at marking m yields a new marking m', which is denoted as $m[t\rangle m'$, where $\forall p \in P$: $m'(p) = \begin{cases} m(p) - 1, & \text{if } p \in {}^\bullet t - t^\bullet; \\ m(p) + 1, & \text{if } p \in t^\bullet - {}^\bullet t; \\ m(p), & \text{otherwise.} \end{cases}$

A marking m_k is reachable from m if there is a firing transition sequence $\sigma = t_1, t_2, \ldots, t_k$ and the marking sequence $m_1, \ldots, m_k$ such that $m[t_1\rangle m_1 \ldots m_{k-1}[t_k\rangle m_k$. $m[\sigma\rangle m_k$ represents that m reaches m_k after firing σ. The set of all marking reachable from m is denoted as $R(N_0, m)$. This paper only considers bounded nets, i.e., $\exists k \in \mathbb{N}, \forall m \in R(N_0, m), \forall p \in P : m(p) \leq k$.

Definition 2 (Workflow net [40])**.** A net $WN = (P, T, F)$ is a workflow net (WF-net) if:

1. *WN* has two special places, i.e., a single source place *start* and a single 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.

The second condition in Definition 2 means that if a new transition t is added to the WF-net such that ${}^\bullet t = \{end\}$ and $t^\bullet = \{start\}$, then the extended WF-net is strongly connected.

Definition 3 (Table)**.** A table $R = \{r_1, r_2, \ldots, r_k\}$ is a finite set of records. Each record $r_i = (A_1, A_2, \ldots, A_n)$ in the R consists of n attribute values, where A_j represents the j^{th} attribute value in the i^{th} record, $j \in \{1, 2, \ldots, n\}$.

For example, an initial table *Patient* is shown in Figure 2 (c), which contains two records $r_1 = (id_1, nurse_1, blood_1, ct_1, bed_1)$ and $r_2 = (id_2, nurse_2, blood_2, ct_2, bed_2)$.

Definition 4 (Workflow Net with Table [25, 45])**.** A workflow net with table (WFT-net) is a 13-tuple $N = (P, T, F, D, R, rd, wt, dt, sel, ins, del, upd, guard)$ where

1. (P, T, F) is a WF-net;
2. D is a finite set of data elements;
3. R is an initial table;
4. $rd : T \to 2^D$ is a labeling function of reading data;
5. $wt : T \to 2^D$ is a labeling function of writing data;
6. $dt : T \to 2^D$ is a labeling function of deleting data;
7. $sel : T \to 2^R$ is a labeling function of selecting a record in a table;
8. $ins : T \to 2^R$ is a labeling function of inserting a record in a table;

9. $del : T \to 2^R$ is a labeling function of deleting a record in a table;
10. $upd : T \to 2^R$ is a labeling function of updating a record in a table; and
11. $guard : T \to G_\Pi$ is a labeling function of guards. $G_\Pi = \{g_1, g_2, \ldots, g_m\}$ is a set of guards, each of which is a Boolean expression over a set of predicates $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$. Labeling functions $\ell_1 : \Pi \to 2^D \cup 2^R$ represents the set of data elements and table occurring in a predicate, and $\ell_2 : G_\Pi \to 2^D \cup 2^R$ represents the set of data elements and table occurring in a guard.

An example of a WFT-net is the workflow of a healthcare management system in Figure 2. Its data elements are $D = \{Id, Nurse, Blood, CT, Bed\}$ and its set of predicates is $\Pi = \{\pi_1 = arrange(nurse), \pi_2 = search(bed)\}$. Furthermore, $\pi_1 = arrange(nurse)$ depends on the value of the data item *nurse* in a table *Patient*. If the value of the data item *nurse* belongs to the *Patient*, then the π_1 is true. Otherwise, it is false. Guards are $G_\Pi = \{g_1, g_2, g_3, g_4\}$. The labeling functions are $\ell_1(\pi_1) = \{Nurse, nurse_1, nurse_2\}$ and $\ell_2(g_1) = \{nurse_1, nurse_2\}$. A writing labeling function is $wt(t_0) = id$, which means that a *write* operation on data element *id* is performed when firing t_0, and a *select* operation on the attribute *Id* in *Patient* is also associated with t_0, i.e., $sel(t_0) = \{id1, id2\}$. Other labeling functions can be understood similarly.

3.2 Motivating Example

This section gives an example of a healthcare management system [46]. Figure 1 describes the workflow in the system by the Business Process Model and Notation (BPMN) activity diagrams and Figure 2 is also a transformed WFT-net. We first review this business process.

A patient with liver and kidney function issues goes to the hospital for physical examination. First, s/he needs to register and make payment (t_0). Then, a doctor randomly assigns a nurse to the patient (t_1). If the nurse is available (t_2), then the nurse will accompany the patient for the examination (t_5). Otherwise, the patient needs to wait for a nurse until one becomes available (t_4). Afterward, the nurse guides the patient for a CT scan (t_7) and blood test (t_6). The patient waits for the examination reports (t_8) and hands them to the doctor for review (t_9). If the examination reports show that the patient's health indicators are normal (t_{10}), it means the patient does not need to be hospitalized (t_{19}). Otherwise, the patient needs further observation in the hospital (t_{11}). The doctor arranges for the patient to be admitted and instructs the nurse to assign a hospital bed to the patient (t_{12}). When a hospital bed becomes available (t_{13}), the nurse arranges for the patient to be hospitalized directly (t_{17}). Otherwise, there are no available beds (t_{14}), the nurse notifies the logistics staff to check for available beds until an empty bed is found (t_{15}), and the patient is admitted smoothly (t_{17}). The patient undergoes the examination and recovers (t_{18}), and is discharged smoothly (t_{19}).

Generally, soundness is used as a criterion to judge whether a model is correct [9, 10, 47]. A WFT-net shown in Figure 2 satisfies soundness because its end state

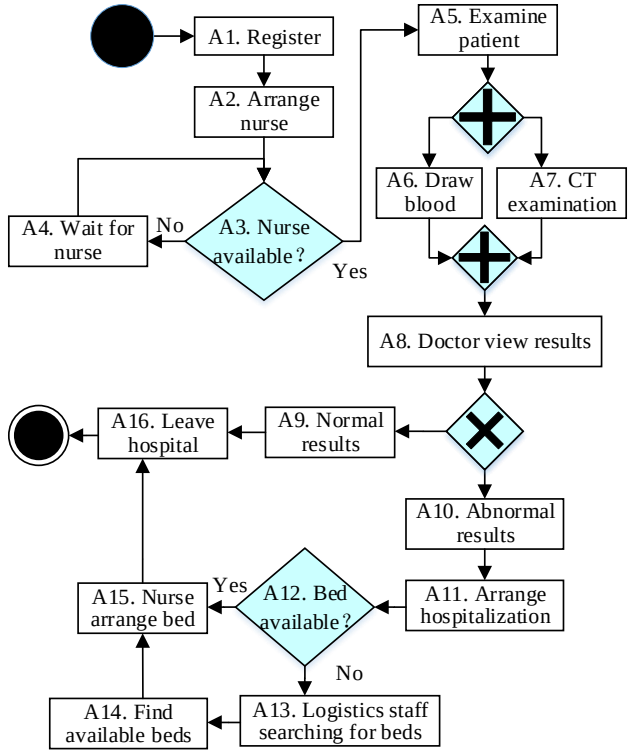


Figure 1. The BPMN model of a healthcare management system

is always reachable. However, analyzing the WFT-net shown in Figure 2 a), we find that the model has the following problems. When at peak time, the nurses are always busy, and no additional nurses are available, i.e., the transition t_4 is always waiting. For a patient who needs to examine liver and kidney function, s/he needs to get blood drawn on an empty stomach. Therefore, the hospital's blood draw is usually scheduled in the morning, while the CT examination is not time-limited. If the patient performs the CT examination activity first, s/he may miss the morning blood draw appointment specified by the hospital. To complete both examinations on the same day, the patient needs to finish them before the blood draw deadline to avoid this situation. When more patients are in the hospital, there is a shortage of available beds. In such case, the staff is constantly searching for an available bed (t_{14}), and the patient has to wait for the search results. If there are no discharged patients, there will not be any available beds, resulting in hospitalized patients waiting. These problems arise because the WFT-net does not consider time elements when modeling the system, leading to incorrect verification results when analyzing the model.

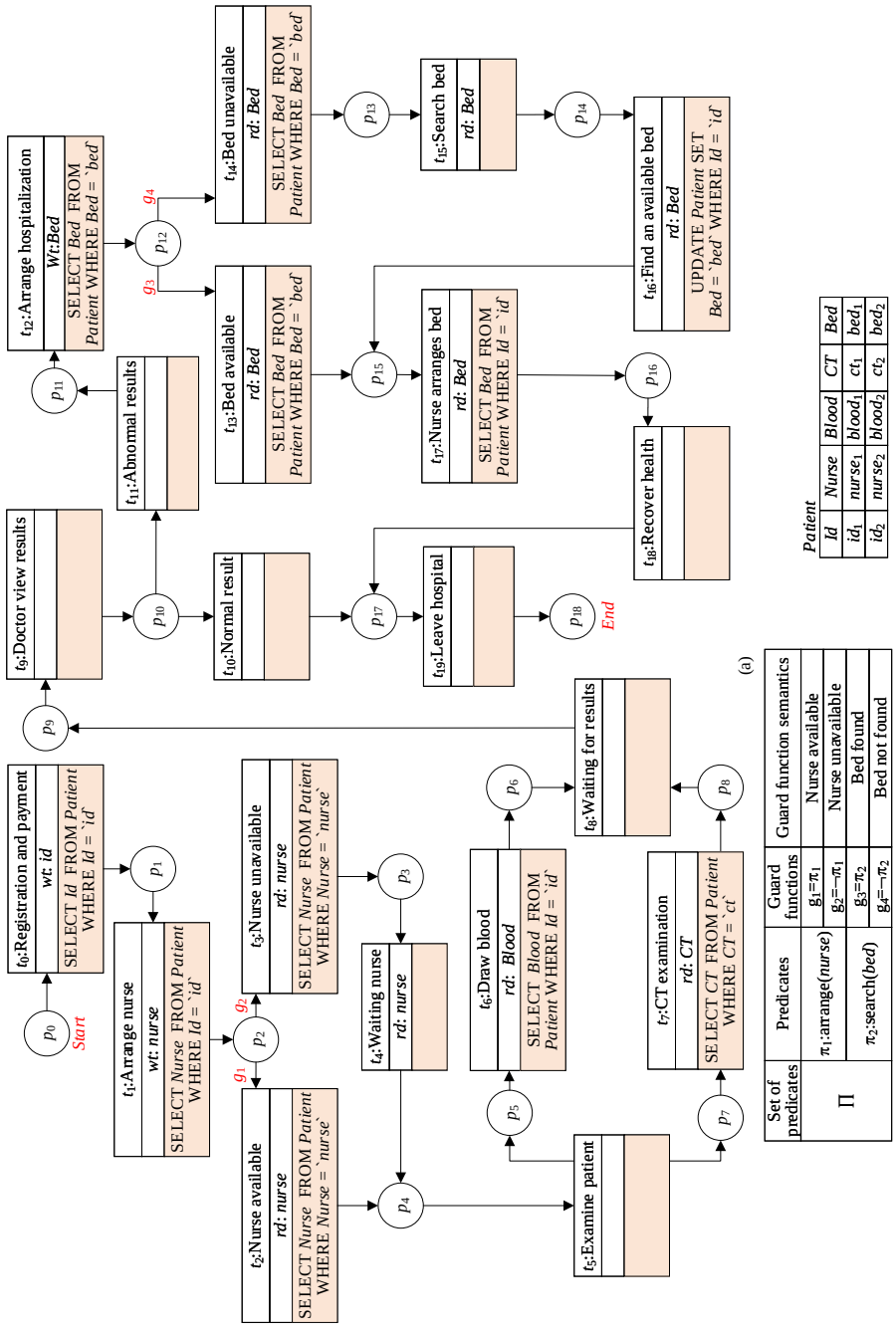


Figure 2. a) A WFT-net; b) guards; c) an initial table

A healthcare management system is a typical TBPMS. From an SRG shown in Figure 3, we can see that the process can be successfully completed because the activity is not constrained by time. However, some activities are strictly constrained by time in a TBPMS. If these activities exceed the specified time limit, the activities immediately stop. For instance, when the blood draw time exceeds the specified time, the blood draw window will be closed, and the patient cannot complete the examination. However, the SRG shown in Figure 3 indicates that the activity is not constrained by time, and the patient can have a blood draw at any time. Therefore, when simulating the TBPMS with a WFT-net, it cannot accurately describe time-related activities, leading to incorrect results in system analysis. Literature [48] provides the firing rule of WFT-net, and this paper does not repeat it.

Through the analysis of motivation cases, we can see that WFT-net ignores the consideration of time elements, leading to system security risks. Therefore, we need a new model to describe the system. This model should not only abstractly model the backend database tables of the system but also consider the influence of time on activities within the system. A timed workflow net with a table is such a model.

4 TIMED WORKFLOW NET WITH TABLE AND FIRING RULES

In order to accurately describe the operation behavior of TBPMS, we propose a timed workflow net with the table (TWFT-net) and its transition firing rules in this section.

4.1 Timed Workflow Net with Table

Definition 5 (Timed workflow net with table)**.** A timed workflow net with table (TWFT-net) is a 4-tuple $N' = (N, Const, Within, Deadline)$ where:

1. N is a WFT-net;
2. $Const : T \to \mathbb{N}$ is a label function, which indicates that firing a transition t requires a-time units to complete ($a \in \mathbb{N}$);
3. $Within : T \to \mathbb{N}$ is a label function, which indicates that firing a transition t must be within a-time units to complete, otherwise the current activity stops; and
4. $Deadline : T \to \mathbb{N}$ is a label function, which indicates that firing a transition t must be within a-time units to complete, otherwise the process will end.

In a TWFT-net, if the time label function of transition t is $Const$, it indicates that the firing time of the t is a fixed value, which we refer to as a static transition. T_c is a set of transitions with time label function being $Const$ [1]. Otherwise, the firing time of t is a dynamic value and t is called a dynamic transition, and T_v is a set of dynamic transitions. Figure 4 a) shows a TWFT-net, which describes basic

[1] $Const[t]$ can be omitted in the TWFT-net, i.e. $Const(t) = [a]$.

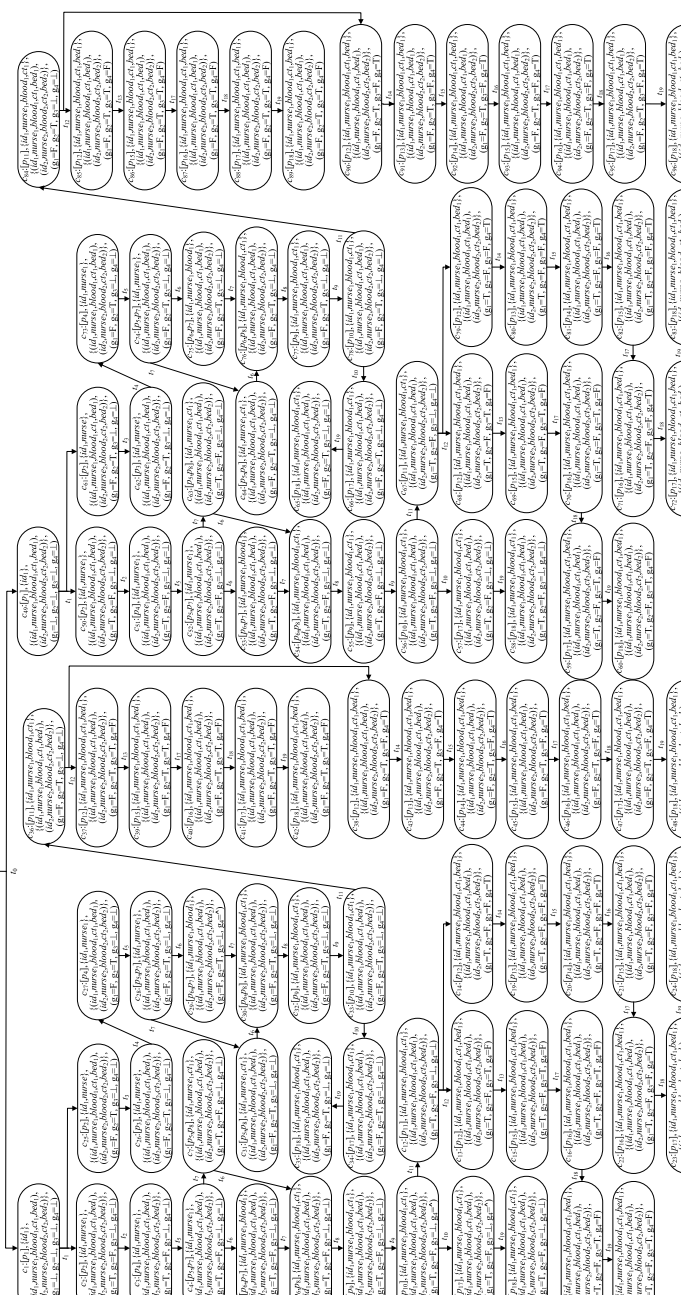


Figure 3. State reachability graph of WFT-net

business logic, data elements operation, database table operations, guards, and time constraints; Figure 4 b) shows the guards; and Figure 4 c) shows an initial table. In the TWFT-net, $T_v = \{t_4, t_6, t_{15}\}$, $T_c = T - T_v$, $Deadline[t_6] = 3$, and $Within[t_{15}] = 2$. In the TWFT-net, data element operations and database table operations are the same as in the WFT-net. Compared with WFT-net, the database table in TWFT-net also records the actual fire time of each user under the dynamic transition. For example, at transition t_4, a user id_1 execution time is $Within[t_4] = 2$, indicating that id_1 requires 2-time units to complete the blood draw activity, which satisfies the 3-time units specified by the system.

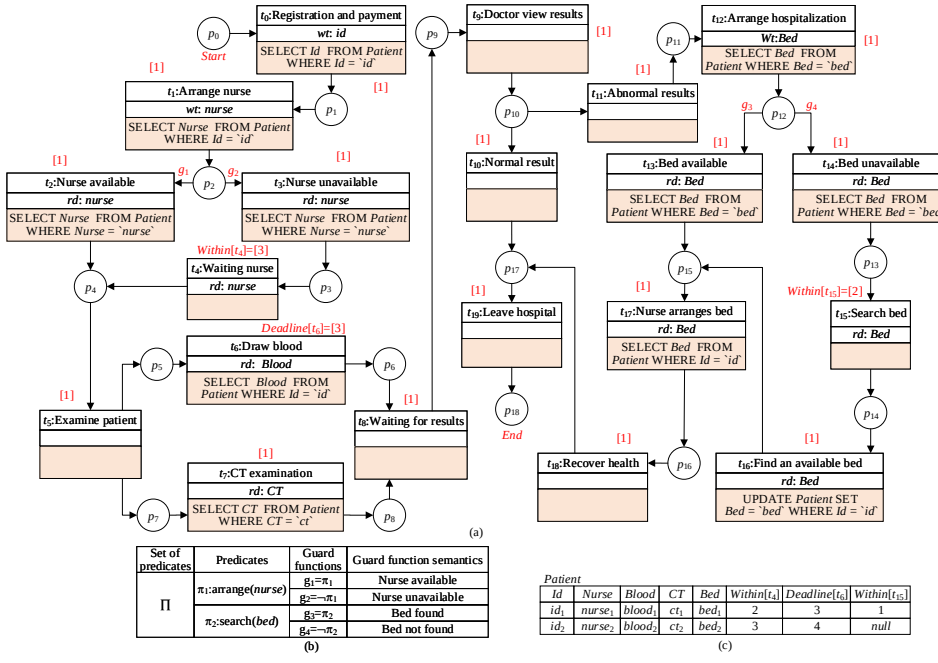


Figure 4. a) A TWFT-net; b) guards; c) an initial table

4.2 The Firing Rules and State Reachability Graph of TWFT-Net

For a TWFT-net, since the time elements have been added to the transitions in a WFT-net, the state of the TWFT-net should consist of the following five components: a marking, data elements operations, the distribution of data item values in a table, guards, and the time elements.

Definition 6 (State)**.** Let N' be a TWFT-net, $c = \langle m, \theta_D, \vartheta_R, \sigma, I \rangle$ being a state of N', where

1. m is a marking of N';
2. $\theta_D : D \to \{\perp, \top\}$ is the values of the data elements in the current state. At an initial state, every data element is ***undefined*** which is represented as $\perp$. When a write or a read operation is executed on a data element, it is assigned a value which means that it is ***defined*** and represented by $\top$;
3. $\vartheta_R : R \to 2^{A_1 \times A_2 \times \cdots \times A_n}$ represents the values of the data items in a table at state c. Notice that if an attribute of a record in the table is not assigned or deleted, then it is null. If an *update* or an *insert* operation is executed on an attribute of a record in the table, then it is assigned a concrete value;
4. $\sigma : \Pi \to \{true, false, \perp\}$ represents the assignment of each predicate. Since each predicate is associated with some data items, after all related data items of a predicate are written, this predicate is ***true*** (T) or ***false*** (F). Otherwise, its value is ***undefined***; and
5. $I = (I_a, I_b)$ is a 2-tuple of timed label, where I_a represents the value of the actual time from the initial state to the current state, and I_b indicates the maximum time value specified by the system from the initial state to the current state.

An initial state of the healthcare management system in Figure 4 is defined to be $c_0 = \langle [\textbf{\textit{start}}], \{Id = \perp, Nurse = \perp, Blood = \perp, CT = \perp, Bed = \perp\}, \{(id_1, nurse_1, blood_1, ct_1, bed_1, 2, 3, 1), (id_2, nurse_2, blood_2, ct_2, bed_2, 3, 4, null)\}, (g_1 = \perp, g_2 = \perp, g_3 = \perp, g_4 = \perp), (0, 0)\rangle$. It means that in the c_0, only place p_0 contains a token, data elements have undefined values, data item values in the table are defined values, guards have undefined values, and the time values are assigned as 0. After firing a transition t, if the t is bound to the operation of the relevant data item in the table, corresponding changes are made to the associated database table. For convenience of description, we define $Ins(R')$ as the set of insert operations on data items in the table R', $Del(R')$ as the set of delete operations on data items in the R', $Sel(R')$ as the set of select operations on data items in the R', $Upd(R')$ as the set of update operations that will be performed on the data items in the R', and $Upd(R')'$ as the set of data items that have been updated. $\Upsilon(DI(t))$ represents the value of actual time required at transition t to fire, and $DI(t)$ represents the value of maximum time allowed at transition t to fire in the system. In summary, we formalize the transitions' enabling conditions and firing rules as follows:

Definition 7 (Firing rules for TWFT-net)**.** Let N' be a TWFT-net. A transition $t \in T$ is enabled at a state $c = \langle m, \theta_D, \vartheta_R, \sigma, I\rangle$, denoted by $c[t\rangle$, if and only if:

1. $m[t\rangle$;
2. $\Upsilon(DI(t)) \leq DI(t)$;
3. $\forall d \in ((rd(t) \cup wt(t)) \cap D) : \theta_D(d) = \top$;
4. $\forall d \in (sel(t) \cup ins(t) \cup upd(t)) : \vartheta_R = \top$;
5. $\sigma(guard(t)) = true$;
6. $\forall t \in T_v \wedge \Upsilon(DI(t)) \leq DI(t) : I = (I_a, I_b) = (\Upsilon(DI(t)), \Upsilon(DI(t)))$; and

7. $\forall t \in T_c : I = (I_a, I_b) = (DI(t), DI(t))$.

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

1. $m[t\rangle m'$;
2. $\forall d \in dt(t) : \theta'_D(d) = \bot$; $\forall d \in (wt(t) \cup rd(t)) : \theta'_D(d) = \top$;
3. $\forall d \in wt(t) \setminus dt(t) : \theta'_D(d) = \top$; $\forall d \in D \setminus (wt(t) \cup dt(t)) : \theta'_D(d) = \theta_D(d)$;
4. $\forall R'.d \in ins(t) \wedge \forall Ins(R') \cap R' \neq \varnothing : \vartheta'_{R'} = \vartheta_{R'}$; $\forall R'.d \in ins(t) \wedge \forall Ins(R') \cap R' = \varnothing : \vartheta'_{R'} = \vartheta_{R'} \cup Ins(R')$;
5. $\forall R'.d \in del(t) \wedge \forall Del(R') \subseteq R' : \vartheta'_{R'} = \vartheta_{R'} \setminus Del(R')$; $\forall R'.d \in del(t) \wedge \forall Del(R') \nsubseteq R' : \vartheta'_{R'} = \vartheta_{R'}$;
6. $\forall R'.d \in upd(t) \wedge \forall Upd(R') \subseteq R' : \vartheta'_{R'} = (\vartheta_{R'} \setminus Upd(R')) \cup Upd(R')'$;
7. $\forall R'.d \in sel(t) \wedge \forall Sel(R') \subseteq R' : \vartheta'_{R'} = \top$; $\forall R'.d \in sel(t) \wedge \forall Sel(R') \nsubseteq R' : \vartheta'_{R'} = \bot$;
8. $\forall g \in G_\pi \wedge \ell_2(g) \in ((wt(t) \vee rd(t) \vee dt(t)) \wedge (upd(t) \vee ins(t) \vee sel(t) \vee del(t))) : \sigma'(g) = \{true, false\}$;
9. $\forall g \in G_\pi \wedge \ell_2(g) \notin ((wt(t) \vee rd(t) \vee dt(t)) \wedge (upd(t) \vee ins(t) \vee sel(t) \vee del(t))) : \sigma'(g) = \bot$;
10. $\forall g \notin G_\pi \wedge (\ell_2(g) \cap (wt(t) \vee rd(t))) = \varnothing : \sigma'(g) = \sigma(g)$;
11. $\forall t \in T_v \wedge \Upsilon(DI(t)) \leq DI(t) : I' = (I'_a, I'_b) = (I_a + \Upsilon(DI(t)), I_b + \Upsilon(DI(t)))$; and
12. $\forall t \in T_c : I' = (I'_a, I'_b) = (I_a + DI(t), I_b + DI(t))$.

For example, given an initial state $c_0 = \langle [\textbf{\textit{start}}], \{Id = \bot, Nurse = \bot, Blood = \bot, CT = \bot, Bed = \bot\}, \{(id_1, nurse_1, blood_1, ct_1, bed_1, 2, 3, 1), (id_2, nurse_2, blood_2, ct_2, bed_2, 3, 4, null)\}, (g_1 = \bot, g_2 = \bot, g_3 = \bot, g_4 = \bot), (0, 0)\rangle$ of the TWFT-net in Figure 3. According to Definition 7, a transition t_0 is enabled at c_0. After firing t_0, a token in ***start*** will be moved into p_1. Since only a *write* operation is performed on the data element *id* at t_0, *id* is a defined value, i.e., $\theta_{id} = id_1$ or $\theta_{id} = id_2$, and other data elements remain undefined values. The database table is $Patient = \{(id_1, nurse_1, blood_1, ct_1, bed_1, 2, 3, 1), (id_2, nurse_2, blood_2, ct_2, bed_2, 3, 4, null)\}$ since t_1 is to select from the *Patient* but not to change it. Since the guards are unrelated to data element *id*, they are still undefined. Due to t_0 is bound to a time element, time will change. When $\theta_{id} = id_1$, the firing time of the t_0 is $Const(t_0) = 1$, so $I = (1, 1)$. Similarly, when $\theta_{id} = id_2$, we obtain $I = (1, 1)$. Therefore, firing t_0 generates two new states: $c_1 = \langle [\textbf{\textit{start}}], \{id_1\}, \{(id_1, nurse_1, blood_1, ct_1, bed_1, 2, 3, 1), (id_2, nurse_2, blood_2, ct_2, bed_2, 3, 4, null)\}, (g_1 = \bot, g_2 = \bot, g_3 = \bot, g_4 = \bot), (1, 1)\rangle$; and $c_2 = \langle [\textbf{\textit{start}}], \{id_2\}, \{(id_1, nurse_1, blood_1, ct_1, bed_1, 2, 3, 1), (id_2, nurse_2, blood_2, ct_2, bed_2, 3, 4, null)\}, (g_1 = \bot, g_2 = \bot, g_3 = \bot, g_4 = \bot), (1, 1)\rangle$.

At state c_1, according to Definition 7, t_1 is enabled at c_1. After firing t_1, since t_1 performs writing operation on data element *Nurse*, the guards g_1 and g_2 are associated with *Nurse*, generating two states: $c_2 = \langle [p_2], \{id_1, nurse_1\}, \{(id_1, nurse_1,$

Algorithm 1 A method to generate an SRG

Input: TWFT-net N'
Output: SRG

Take c_0 as the root node of N' and mark it as ***new***;
while there is a node marked ***new*** **do**
  Select any node labeled as ***new*** and denoted it by c;
  if there is a directed path from c_0 to c' and c' is the same as c **then**
    Mark c as ***old***, and return to step 2;
  if $\forall t \in T : \neg c[t\rangle$ **then**
    Erase the ***new*** label from node c, and return to step 2;
  for $t \in T$ **do**
    if $t \in T_c \wedge c[t\rangle$ **then**
      According to Definition 7, calculate $c[t\rangle c'$;
    if $t \in T_v \wedge \Upsilon(DI(t)) \leq DI(t) \wedge c[t\rangle$ **then**
      According to Definition 7, calculate $c[t\rangle c'$;
    else if $DI(t).name = Within$ **then**
      continue;
    else
      break;
    if c' already exists in the directed path from c_0 **then**
      Draw a directed arc from c to c', and mark the side of the arc as t;
    else
      Generate a node c' and mark it as ***new*** in SRG(N'). Then, draw a directed arc from c to c', marking the side of the arc as t and $\Upsilon(DI(t))$. Erase the ***new*** label from node c, and return to step 2.

$blood_1, ct_1, bed_1, 2, 3, 1), (id_2, nurse_2, blood_2, ct_2, bed_2, 3, 4, null)\}, (g_1 = T, g_2 = F, g_3 = \perp, g_4 = \perp), (2, 2)\rangle$, and $c_{25} = \langle [p_2], \{id_1, nurse\}, \{(id_1, nurse_1, blood_1, ct_1, bed_1, 2, 3, 1), (id_2, nurse_2, blood_2, ct_2, bed_2, 3, 4, null)\}, (g_1 = F, g_2 = T, g_3 = \perp, g_4 = \perp), (2, 2)\rangle$.

Let N' be a TWFT-net, according to transition enabling conditions and firing rules, we propose an algorithm for generating an SRG of the N', as shown in Algorithm 1.

We assume that the number of nodes in a TWFT-net is n and the number of iterations in the **while** loop is m, and then the time complexity of the Algorithm 1 is $O(|n| \bullet |m|)$. In Algorithm 1, the labeling of the nodes changes gradually during the loop according to the conditions, which eventually leads to no node being labeled as ***new***, making the loop end. Therefore, Algorithm 1 can be terminated.

Based on Algorithm 1, we generated an SRG of the TWFT-net, as shown in Figure 5. Since a user id_2 did not complete the blood draw within the specified time, it results in the failure of the entire physical examination process, leading to process termination. Consequently, compared with the WFT-net, the SRG generated by the

TWFT-net has a total reduction of 38 invalid states. We can know from the SRG, as shown in Figure 5, that TWFT-net can accurately reflect the operating behavior of the actual system when describing the TBPMS. The SRG shows that the id_2 has no successor states at transition t_4. It occurs because id_2 undergoes the blood draw after a delay of 4-time units, significantly exceeding the system's specified time. As a result, the system automatically stops the blood draw, preventing the user from completing subsequent examinations on the same day and terminating the entire process. In contrast, since the time constraints are not considered in the WFT-net, users can undergo examinations at any time, which contradicts the realistic scenario. As a result, the SRG generated by the WFT-net without time constraints (as shown in Figure 3) contains many invalid states. However, the TWFT-net considers the influence of time on the workflow system, effectively avoiding such situations.

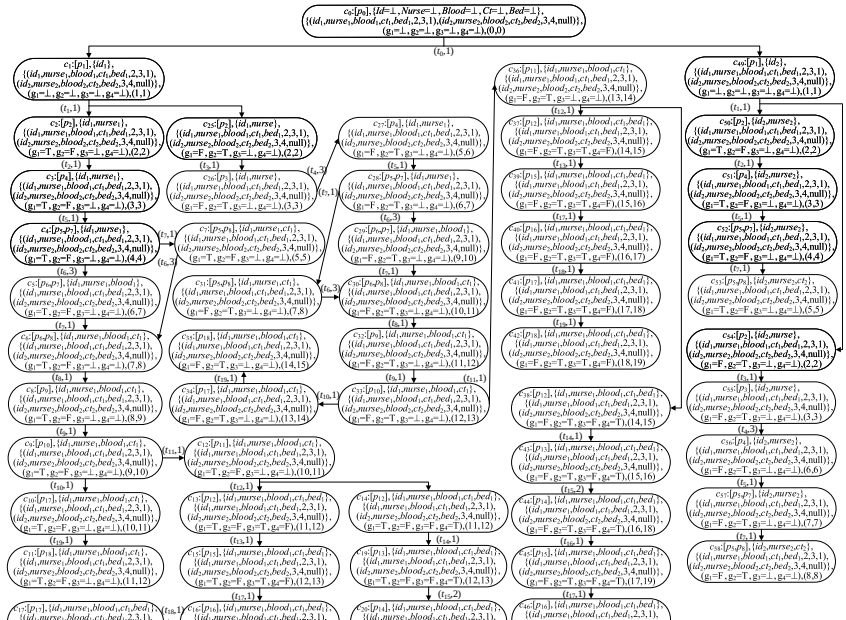


Figure 5. State reachability graph of TWFT-net

5 TIMED DATABASE COMPUTATION TREE LOGIC AND SOUNDNESS ANALYSIS

Model checking is a technique used for automatically verifying the soundness of a workflow model. Its basic idea is to traverse an SRG of the model to check if the model satisfies the given design requirements. Generally speaking, we will verify whether the structure and dynamic behavior of a TWFT-net are consistent with the design requirements of the modeled system and detect whether the system terminates at an acceptable state. Therefore, in order to verify the soundness of a TBPMS, this section presents a model checking method based on timed database computation tree logic (TDCTL). First, we propose a TDCTL model checking method and define a soundness property of TWFT-net. Then, the soundness definition is transformed into corresponding TDCTL formulas. Finally, the satisfaction of TDCTL formulas in the SRG is verified. If satisfied, it indicates that the system satisfies the design requirements; otherwise, counterexamples are provided. Thus, the issue of system soundness is transformed into the satisfiability problem of TDCTL formulas. In the following sections, we will introduce the relevant concepts of TDCTL and soundness.

5.1 Syntax and Semantics of TDCTL

The soundness of a TBPMS depends not only on control flow and data flow but also on the precise execution time of various activities within the system. Therefore, this paper proposes an approach called TDCTL to describe the design requirements of the TBPMS. The syntax of TDCTL is based on TWFT-net, and the semantics of TDCTL are explained using the SRG of TWFT-net. The SRG structure can be defined as a 5-tuple SRG $= \langle AP, C, c_0, \Re, L\rangle$, where:

1. AP is a finite set of atomic propositions;
2. C is a finite set of states;
3. $c_0 \in C$ is an initial state;
4. $\Re \subseteq C \times C$ is a transition relation; and
5. $L : C \to 2^{AP}$ is a labeling function used to label the set of atomic propositions satisfied on a state.

An SRG of TWFT-net, as shown in Figure 5, a set of atomic propositions $AP = \{p_0, \ldots, p_{18}, id_1, nurse_1, blood_1, ct_1, bed_1, 2, 3, 1, id_2, nurse_2, blood_2, ct_2, bed_2, 3, 4\}$, a set of states $C = \{c_0, c_1, \ldots, c_{58}\}$, a labeling function $L(c_0) = \{p_0, \{(id_1, nurse_1, blood_1, ct_1, bed_1, 2, 3, 1), (id_2, nurse_2, blood_2, ct_2, bed_2, 3, 4, null)\}$, and a transition relation is $\Re = \{(c_0, c_1), (c_0, c_2), \ldots, \}$.

Giving a TWFT-net $N' = (N, DI)$, the TDCTL syntax is defined as follows:

Definition 8 (Syntax of TDCTL)**.** TDCTL is defined with the following grammar: $\varphi ::= true|p|D_i^j|\neg\varphi|\varphi\wedge\varphi_1|EX\varphi|EG\varphi|E\varphi U_{\triangle a}\varphi_1|A\varphi U_{\triangle a}\varphi_1|\exists D_i^j \in R, [\varphi(D_i^j)\triangle a]|\forall D_i^j$

$\in R, [\varphi(D_i^j) \triangle a]$, where $\triangle \in \{<, \leq, =, \geq, >\}$, $p \in P$, $D_i^j \in R$, $a \in \mathbb{R}$ is natural numbers, φ and φ_1 are TDCTL formulas.

D_i^j represents the data item of the j^{th} dimensional vector of the i^{th} record in the table, e.g., Figure 4 c) shows an initial table $Patient = \{(id_1, nurse_1, blood_1, ct_1, bed_1, 2, 3, 1), (id_2, nurse_2, blood_2, ct_2, bed_2, 3, 4, null)\}$, we can compute $D_1^2 = nurse_1$. Other basic operators are derived from the above ones: $deadlock \equiv \neg EX true$, $EF_{\triangle a}\varphi \equiv E(true U_{\triangle a}\varphi)$, $AF_{\triangle a}\varphi \equiv A(true U_{\triangle a}\varphi)$, $AG\varphi \equiv \neg EF\neg\varphi$, $AX\varphi \equiv \neg EX\neg\varphi \wedge \neg deadlock$, and $\varphi \rightarrow \varphi_1 \equiv \neg\varphi \vee \varphi_1$.

Definition 9 (Semantics of TDCTL)**.** Given an SRG of TWFT-net, φ is a TDCTL formula, and SRG, $c \models \varphi$ means that this formula is satisfied at state c of the SRG. If the SRG is clear from the context, it can be omitted. The relation $\models$ is recursively defined as follows:

1. $c \models true$;
2. $c \models p \Leftrightarrow p \in L(c)$;
3. $c \models D_i^j \Leftrightarrow D_i^j \in L(c)$;
4. $c \models \neg\varphi \Leftrightarrow c \not\models \varphi$;
5. $c \models \varphi_1 \wedge \varphi_2 \Leftrightarrow c \models \varphi_1$ and $c \models \varphi_2$;
6. $c \models \exists D_i^j \in R, [\varphi(D_i^j) \triangle a] \Leftrightarrow \exists D_i^j \in R$ and $c \models \varphi(D_i^j) \triangle a$;
7. $c \models \forall D_i^j \in R, [\varphi(D_i^j) \triangle a] \Leftrightarrow \forall D_i^j \in R$ and $c \models \varphi(D_i^j) \triangle a$;
8. $c \models \mathrm{EX}\varphi \Leftrightarrow \exists (c, c_1) \in \Re(c)$ and $c_1 \models \varphi$;
9. $c \models EG\varphi \Leftrightarrow$ there exists some paths $\langle c_1, c_2\rangle\langle c_2, c_3\rangle \ldots$, and for all c_i along the path, we have $c_i \models \varphi$, where $c_1 = c$;
10. $c \models E(\varphi_1 U_{\triangle a}\varphi_2) \Leftrightarrow$ for some paths $\langle c_1, c_2\rangle \ldots \langle c_{k-1}, c_k\rangle$, we have: (1) $\exists k \geq 0, \pi(k) \models \varphi_2$; (2) $\forall 0 \leq j < k, \pi(j) \models \varphi_1$; (3) $c_k.I_a \triangle a$; and
11. $c \models A(\varphi_1 U_{\triangle a}\varphi_2) \Leftrightarrow$ for all paths $\langle c_1, c_2\rangle \ldots \langle c_{k-1}, c_k\rangle$, we have: (1) $\exists k \geq 0, \pi(k) \models \varphi_2$; (2) $\forall 0 \leq j < k, \pi(j) \models \varphi_1$; (3) $c_k.I_a \triangle a$.

TDCTL expands upon CTL by adding database table operations and specific time element constraints to CTL. It describes design requirements related to time and database tables in TBPMS using quantified CTL temporal operators. Generally, we express design requirements using TDCTL formulas φ. When φ is valid in a TWFT-net, it means that the calculation result of φ on the SRG of TWFTC-net is true, indicating that the system satisfies the design requirements. For example, in the TBPMS, as shown in Figure 4, we require users to finally be able to complete the examination when they go to the hospital. This design requirement can be described using a TDCTL formula $\varphi_1 = AG(\forall id \in R, [id \neq \varnothing] \rightarrow AFp_{18})$. From the SRG shown in Figure 5, we can see that id_2 exceeds the time specified by the system during the blood draw, resulting in id_2 being unable to complete the examination. Therefore, φ_1 is incorrect.

Algorithm 2 Computation of the $\mathrm{Sat}(\varphi)$

Input: A TWFT-net N', SRG, TDCTL formula φ
Output: $\mathrm{Sat}(\varphi) = \{c \in C | c \models \varphi\}$;

1: **if** $\varphi == true$ **then**
2: Return C;
3: **if** $\varphi == p \vee D_i^j$ **then**
4: Return $(\{c|\varphi \in L(c)\})$;
5: **if** $\varphi == \neg\varphi_1$ **then**
6: Return C-$\mathrm{Sat}(\varphi_1)$;
7: **if** $\varphi == \varphi_1 \wedge \varphi_2$ **then**
8: Return $\mathrm{Sat}(\varphi_1) \cap \mathrm{Sat}(\varphi_2)$;
9: **if** $\varphi == EX\varphi_1$ **then**
10: Return $\{c \in C|(c, c') \in \Re \wedge c' \in \mathrm{Sat}(\varphi_1)\}$;
11: **if** $\varphi == EG\varphi_1$ **then**
12: Return $\mathrm{Sat}_{EG}(\varphi_1)$;
13: **if** $\varphi == \exists D_i^j \in R, [\varphi_1(D_i^j) \triangle a]$ **then**
14: Return $\mathrm{Sat}(\exists, \varphi_1(D_i^j), \triangle, a)$;
15: **if** $\varphi == \forall D_i^j \in R, [\varphi_1(D_i^j) \triangle a]$ **then**
16: Return $\mathrm{Sat}(\forall, \varphi_1(D_i^j), \triangle, a)$;
17: **if** $\varphi == E(\varphi_1 U_{\triangle a} \varphi_2)$ **then**
18: Return $\mathrm{Sat}_{EU}(\varphi_1, U, a, \varphi_2)$;
19: **if** $\varphi == A(\varphi_1 U_{\triangle a} \varphi_2)$ **then**
20: Return $\mathrm{Sat}_{AU}(\varphi_1, U, a, \varphi_2)$;

5.2 Model Checking Algorithms of TDCTL

This section gives TDCTL model checking algorithms based on TWFT-net. First, we generate an SRG of the TWFT-net, where C is the set of states in the SRG. Then, we provide a TDCTL formula φ that describes the properties of the system. Finally, we identify all state sets in C that satisfy the formula φ and denote them as $\mathrm{Sat}(\varphi)$. $Post(c)$ represents the successor set of state c and $Pre(c)$ represents the predecessor set of state c. If the formula φ is true on the SRG, it means that the system satisfies the property specifications. Algorithm 2 shows the process of calculating $\mathrm{Sat}(\varphi)$. It recursively calculates the state sets that satisfy each subformula of φ, leading to the final state set $\mathrm{Sat}(\varphi)$.

In Algorithm 2, except for the last four operators, the rest are traditional CTL operators, so they will not be described in detail. Interested readers can refer to [49, 16, 50]. This section only provides the $\mathrm{Sat}(\varphi)$ solving algorithm for operators related to data items and time. The algorithms for calculating $\mathrm{Sat}(\exists, \varphi(D_i^j), \triangle, a)$, $\mathrm{Sat}(\forall, \varphi(D_i^j), \triangle, a)$, $\mathrm{Sat}_{EU}(\varphi_1, U, a, \varphi_2)$, and $\mathrm{Sat}_{AU}(\varphi_1, U, a, \varphi_2)$ can be found in Algorithm 3, Algorithm 4, Algorithm 5, and Algorithm 6, respectively.

Algorithm 3 shows the detailed steps for calculating $\mathrm{Sat}(\exists, \varphi(D_i^j), \triangle, a)$. First, we need to compute set $\mathrm{Sat}(\varphi(D_i^j))$. Then, the algorithm iterates through each state $c \in \mathrm{Sat}(\varphi(D_i^j))$, if there exists c satisfies $c.D_i^j \triangle a$, then add c to $\mathrm{Sat}(c)$. Finally, it returns $\mathrm{Sat}(c)$.

Algorithm 3 Computation of the $\mathrm{Sat}(\exists, \varphi(D_i^j), \triangle, a)$

Input: A TWFT-net N', SRG, TDCTL formula φ_1
Output: $\mathrm{Sat}(\exists, \varphi(D_i^j), \triangle, a)$;
1: $\mathrm{Sat}(\varphi(D_i^j)) = \{c \in C | c.D_i^j \in R\}$;
2: $\mathrm{Sat}(c) = \{c | \exists c \in \mathrm{Sat}(\varphi(D_i^j)) : c.D_i^j \triangle a\}$.
3: Return $\mathrm{Sat}(c)$.

For example, we calculate a TDCTL formula $\varphi = \exists id.time(I_b) \in R$, $[id.time(I_b) \leq 3]$ based on an SRG as shown in Figure 5, as follows:

1. $\mathrm{Sat}(\varphi_1) = \mathrm{Sat}(id.time(I_b)) = \{c \in C | c.id.time(I_b) \in R\} = \{c_0, c_1, \ldots, c_{58}\}$; and
2. $\mathrm{Sat}(\varphi) = \{c | \exists c.id.time(I_b) \in \mathrm{Sat}(\varphi_1) : c.id.time(I_b) \triangle a\} = \{c_0, c_1, c_2, c_3, c_{25}, c_{26}, c_{49}, c_{50}, c_{51}, c_{54}, c_{55}\}$.

Algorithm 4 shows the detailed steps for calculating $\mathrm{Sat}(\forall, \varphi(D_i^j), \triangle, a)$. First, we need to compute set $\mathrm{Sat}(\varphi(D_i^j))$. Then, the algorithm iterates through each state $c \in \mathrm{Sat}(\varphi(D_i^j))$, if all states c satisfy $c.D_i^j \triangle a$, then add c to $\mathrm{Sat}(c)$. Finally, it returns $\mathrm{Sat}(c)$.

Algorithm 4 Computation of the $\mathrm{Sat}(\forall, \varphi(D_i^j), \triangle, a)$

Input: A TWFT-net N', SRG, TDCTL formula φ
Output: $\mathrm{Sat}(\forall, \varphi(D_i^j), \triangle, a)$;
1: $\mathrm{Sat}(\varphi(D_i^j)) = \{c \in C | c.D_i^j \in R\}$;
2: $\mathrm{Sat}(c) = \{c | \forall c \in \mathrm{Sat}(\varphi(D_i^j)) : c.D_i^j \triangle a\}$.
3: Return $\mathrm{Sat}(c)$.

For example, we calculate a TDCTL formula $\varphi = \forall id.time(I_b) \in R$, $[id.time(I_b) \leq 3]$ based on an SRG. Compared with Algorithm 3, the Algorithm 4 has stronger constraints and requires that all states satisfy φ. The process of calculating $\mathrm{Sat}(\varphi)$ based on the SRG, as shown in Figure 5, is as follows:

1. $\mathrm{Sat}(\varphi_1) = \mathrm{Sat}(id.time(I_b)) = \{c \in C | c.id.time(I_b) \in R\} = \{c_0, c_1, \ldots, c_{58}\}$; and
2. $\mathrm{Sat}(\varphi) = \{c | \forall c.id.time(I_b) \in \mathrm{Sat}(\varphi_1) : c.id.time(I_b) \triangle a\} = \varnothing$.

Algorithm 5 calculates the set of states that satisfy $E(\varphi_1 U_{\triangle a} \varphi_2)$ based on the traditional CTL model checking algorithm, denoted as $\mathrm{Sat}_{EU}(\varphi_1, \triangle, a, \varphi_2)$. First, the algorithm initializes the sets Q_1, Q_2, Q_{old}, and Q_{new} as empty, then iterating through states c satisfying φ_2 and $c.time \triangle a$, adding them to Q_{new}, and updating

Q_2 to match Q_{new}. Subsequently, states c satisfying φ_1 and $c.time \triangle a$ are added to Q_1. During the **while** loop where $Q_{old} \neq Q_{new}$, Q_{old} is updated to Q_{new}, and a new $Q_{new} = Q_2 \cup (Q_1 \cap \{c \in C | Post(c.time \triangle a) \cap Q_{new} \neq \varnothing\})$ is computed. Finally, the algorithm returns Q_{old} as the output result.

Algorithm 5 Computation of the $\mathrm{Sat}_{EU}(\varphi_1, \triangle, a, \varphi_2)$

Input: A TWFT-net N', SRG, TDCTL formula φ
Output: $\mathrm{Sat}_{EU}(\varphi_1, \triangle, a, \varphi_2)$;

$Q_1 = Q_2 = Q_{old} = Q_{new} = \varnothing$;
for $c \in \mathrm{Sat}(\varphi_2) \wedge c.time \triangle a$ **do**
    $Q_{new} = Q_{new} \cup c$;
$Q_2 = Q_{new}$
for $c \in \mathrm{Sat}(\varphi_1) \wedge c.time \triangle a$ **do**
    $Q_1 = Q_1 \cup c$;
while $Q_{old} \neq Q_{new}$ **do**
    $Q_{old} = Q_{new}$;
    $Q_{new} = Q_2 \cup (Q_1 \cap \{c \in C | Post(c.time \triangle a) \cap Q_{new} \neq \varnothing\})$;
Return Q_{old}.

For a TBPMS illustrated in Figure 4, we require that once a user completes the payment, they must undergo a blood draw within 8-time units. This design requirement can be described using a TDCTL formula $\varphi = E(\exists id \in RU_{\leq 8} p_6)$. Based on the SRG shown in Figure 5 and Algorithm 5, we calculate the $\mathrm{Sat}(\varphi)$ process as follows:

1. $\mathrm{Sat}(\varphi_2) = \mathrm{Sat}(p_6) = \{c_5, c_6, c_{29}, c_{30}\}$;
2. $\mathrm{Sat}(\varphi_2 \wedge \varphi_2.c.time \leq 8) = \{c_5, c_6\}$;
3. $\mathrm{Sat}(\varphi_1) = \mathrm{Sat}(\exists id \in R) = \{c_0, c_1, \ldots, c_{58}\}$;
4. $\mathrm{Sat}(\varphi_1 \wedge \varphi_1.c.time \leq 8) = \{c_0, \ldots, c_7, c_{25}, \ldots, c_{28}, c_{31}, c_{49}, \ldots, c_{58}\}$;
5. $Q_{old} = \mathrm{Sat}(pre(c_5)) \cap \mathrm{Sat}(\varphi_1 \wedge \varphi_1.c.time \leq 8) = \{c_0, c_1, c_2, c_3, c_4, c_5\}$; and
6. $\mathrm{Sat}(\varphi) = Q_{old} \cup (\mathrm{Sat}(pre(c_6)) \cap \mathrm{Sat}(\varphi_1 \wedge \varphi_1.c.time \leq 8)) = \{c_0, c_1, c_2, c_3, c_4, c_5\}$.

Algorithm 5 calculates the set of states that satisfy $E(\varphi_1 U_{\triangle a} \varphi_2)$ based on the traditional CTL model checking algorithm, denoted as $\mathrm{Sat}_{EU}(\varphi_1, \triangle, a, \varphi_2)$. First, the algorithm initializes the sets Q_1, Q_2, Q_{old}, and Q_{new} as empty, then iterating through states c satisfying φ_2 and $c.time \triangle a$, adding them to Q_{new}, and updating Q_2 to match Q_{new}. Subsequently, states c satisfying φ_1 and $c.time \triangle a$ are added to Q_1. During the **while** loop where $Q_{old} \neq Q_{new}$, Q_{old} is updated to Q_{new}, and a new $Q_{new} = Q_2 \cup (Q_1 \cap \{c \in C | Post(c.time \triangle a) \cap Q_{new} \neq \varnothing\})$ is computed. Finally, the algorithm returns Q_{old} as the output result.

Algorithm 6 shows a process of computing $\mathrm{Sat}_{AU}(\varphi_1, \triangle, a, \varphi_2)$. First, the algorithm initializes the sets Q_1, Q_2, Q_{old}, and Q_{new} as empty, then iterating through

states c satisfying φ_2 and $c.time \triangle a$, adding them to Q_{new}, and updating Q_2 to match Q_{new}. Subsequently, states c satisfying φ_1 and $c.time \triangle a$ are added to Q_1. During the **while** loop where $Q_{old} \neq Q_{new}$, Q_{old} is updated to Q_{new}, and a new $Q_{new} = Q_2 \cup (Q_1 \cap \{c \in C | Post(c.time \triangle a) \subseteq Q_{new}\})$ is computed. Finally, the algorithm returns Q_{old} as the output result.

Algorithm 6 Computation of the $\mathrm{Sat}_{AU}(\varphi_1, \triangle, a, \varphi_2)$

Input: A TWFT-net N', SRG, TDCTL formula φ
Output: $\mathrm{Sat}_{AU}(\varphi_1, \triangle, a, \varphi_2)$;

```
Q_1 = Q_2 = Q_old = Q_new = ∅;
for c ∈ Sat(φ_2) ∧ c.time△a do
    Q_new = Q_new ∪ c;
Q_2 = Q_new
for c ∈ Sat(φ_1) ∧ c.time△a do
    Q_1 = Q_1 ∪ c;
while Q_old ≠ Q_new do
    Q_old = Q_new;
    Q_new = Q_2 ∪ (Q_1 ∩ {c ∈ C|Post(c.time△a) ⊆ Q_new});
Return Q_old.
```

For a TBPMS illustrated in Figure 4, we require that all users undergo a blood draw within 8-time units after completing their payment. This design requirement can be described using the TDCTL formula $\varphi = A(\forall id \in RU_{\leq 8} p_6)$. Based on the SRG shown in Figure 5 and Algorithm 6, we calculate the $\mathrm{Sat}(\varphi)$ process as follows:

1. $\mathrm{Sat}(\varphi_2) = \mathrm{Sat}(p_6) = \{c_5, c_6, c_{29}, c_{30}\}$;
2. $\mathrm{Sat}(\varphi_2 \wedge \varphi_2.c.time \leq 8) = \{c_5, c_6\}$;
3. $\mathrm{Sat}(\varphi_1) = \mathrm{Sat}(\forall(id \in R)) = \varnothing$;
4. $\mathrm{Sat}(\varphi_1 \wedge \varphi_1.c.time \leq 8) = \varnothing$;
5. $Q_{old} = \mathrm{Sat}(pre(c_5)) \cap \mathrm{Sat}(\varphi_1 \wedge \varphi_1.c.time \leq 8) = \varnothing$; and
6. $\mathrm{Sat}(\varphi) = Q_{old} \cup (\mathrm{Sat}(pre(c_6)) \cap \mathrm{Sat}(\varphi_1 \wedge \varphi_1.c.time \leq 8)) = \varnothing$.

Since user id_2 took more time than the system's specified limit ($4 > 3$) during the blood draw, the blood test failed. Therefore, the TDCTL formula φ is not satisfied.

We assume that n is the size of the set C, m is the size of the set $\mathrm{Sat}(\varphi(D_i^j))$, x is the size of the set $\mathrm{Sat}(\varphi_1)$, and y is the size of the set $\mathrm{Sat}(\varphi_2)$. Then the time complexity of Algorithm 3 is $O(|m| + |n|)$, the time complexity of Algorithm 4 is $O(|n|)$, and the time complexity of Algorithm 5 and 6 are both $O(|n| + |x| + |y|)$.

5.3 Soundness Verification

Soundness is employed extensively to define the correctness of workflow models [51, 50, 52]. However, traditional soundness is only considered from the control flow perspective. It requires every activity in a workflow model to be executed except for requiring the workflow model to be terminated. For a TBPMS, its correctness depends on control flow, data flow, and time elements. Therefore, this paper redefines soundness, which primarily consists of three components: logical control layer, data design requirements, and time constraints. The redefined soundness not only preserves the traditional definition but also provides effective guarantees for the workflow model in terms of data flow and time constraints.

Definition 10 (Soundness)**.** Let N' be a TWFT-net with initial state $c_0 = \langle m_0, \theta_0, \vartheta_0, \sigma_0, I_0\rangle$ and final state $c_{end} = \langle m_{end}, \theta_{end}, \vartheta_{end}, \sigma_{end}, I_{end}\rangle$. Let $\mathrm{R}(N')$ be a set of reachable states of N' and ι stands for a firing sequence. $\Theta = (\varphi_1, \varphi_2, \ldots, \varphi_n)$ is a set of TDCTL formulas representing data design requirements and time constraint requirements of N'. C_{end} is a set of *end* states, c_{end} is a *end* state, and p_{end} is a *end* place. τ stands for the time sequence. A TWFT-net is sound if all the following properties hold:

1. **P1**. $\forall c \in R(N'), \exists C_{end} \in c_{end} : c \xrightarrow{(\iota,\tau)} c_{end}$;
2. **P2**. $\forall c \in R(N') : c(M) \geq c(p_{end}) \Rightarrow c(m) = c(p_{end})$; and
3. **P3**. $\forall \varphi_i \in \Theta, N' \vDash \varphi_i$.

The first condition means that for each non-end state, there exists a reachability path from it to an end state. The second condition means that the end state is always reachable and there is no additional token in the N'. The last condition means that N' satisfies all TDCTL formulas.

Referring to the example in Figure 4, TWFT-net is not sound since ***P1*** and ***P2*** do not hold. When a patient id_2 misses a blood draw, the examination stops, and the activity cannot reach the final state. From an SRG shown in Figure 5, we can see that since id_1 completes all the examinations within the specified time, id_1 reaches the final state. However, id_2 does not. Compared to the SRG in Figure 3, Figure 5 considers the time constraints. Therefore, TWFT-net is more accurate in describing TBPMS.

Model checking techniques are widely used for workflow system verification. Leveraging the syntax and semantics of TDCTL, we convert the soundness property into corresponding TDCTL formulas. Consequently, the verification of soundness becomes a matter of assessing whether the logical formulas specified in TDCTL are satisfied by the system, as shown in Table 1.

6 TOOL AND EXPERIMENTS

In this section, we evaluate our method and tool through relevant experiments.

Property	Explain	TDCTL formula
P1	There exist some reachability paths where the states along those paths can reach the final state.	$\varphi_{p1} = AG(EFc_{end})$
P2	In all reachability paths, the end state can be reached without deadlock states.	$\varphi_{p2} = AF(C_{end}.p = p_{end})$
P3	N' satisfies all data design requirements and time design requirements.	$\varphi_{p4} = \forall\varphi_i \in \Theta, N' \vDash \varphi_i$

Table 1. TDCTL formula corresponding to soundness

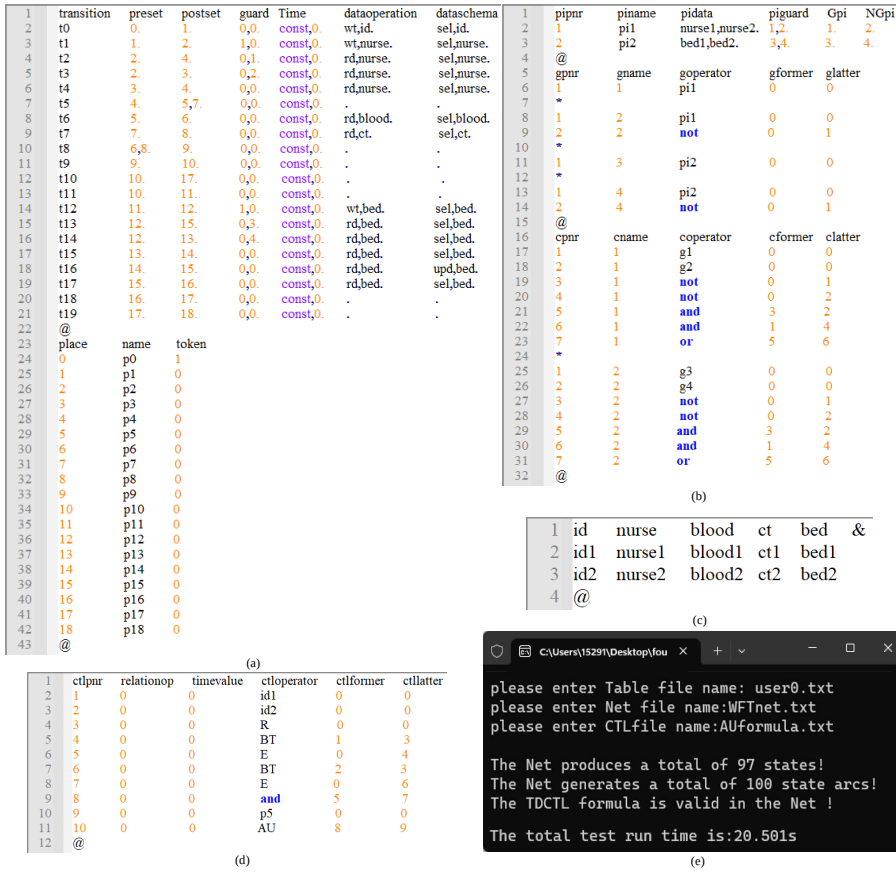
Section 6.1 introduces our tool and development environment, and shows our tool through motivating example. In Section 6.2, we conduct experiments on the motivating example using various performance metrics to demonstrate the effectiveness and feasibility of our method and tool. In Section 6.3, we conduct experiments on established benchmark cases, comparing our method with existing modeling approaches to further highlight its superiority.

6.1 Tool

We have implemented our algorithm and developed a tool for TWFT-net model checking. Our experiments and development environment were conducted on a PC with an Intel Core i5-8500 CPU (3.00 GHz) and 8.0 GB of memory. Our tool is written in the C++ programming language. The input to our tool includes two text files: one describing the TWFT-net and the other specifying the design requirements using TDCTL formulas. After our tool reads the text files, it automatically outputs the verification results for TDCTL formulas and displays the number of states and arcs in the SRG. Figure 6 a) shows the text file of the WFT-net in Figure 2, 6 b) shows an initial table, 6 c) shows the guards, 6 d) represents a TDCTL formula $\varphi_1 = A(\exists(id_1 \in R, id_2 \in R)Up_5)$ with time is 0, and 6 e) shows the experimental verification results [2].

A similar analysis, Figure 7 (a) shows the text file of a TWFT-net from Figure 4; 7 (b) shows an initial table, 7 (c) shows the guards; 7 (d) presents a TDCTL formula $\varphi_2 = A(\exists(id_1 \in R, id_2 \in R)U_{\leq 15}p_5)$ with a time not equal to 0; and 7 (e) shows the experimental verification results (the red dashed lines in Figure 7 pertain to the time constraints). By analyzing the experimental results shown in Figure 6 (e), we find that the SRG consists of 97 states and 100 state arcs. Since the WFT-net is not considered with time constraints, the φ_1 formula is evaluated as true. Compared with Figure 6 (e), the experimental results in Figure 7 (e) show a total of 59 states and 60 state arcs. Since the time constraints considered in the TWFT-net resulted in the user id_2 process being incomplete, the TDCTL formula φ_2 verification result is false. It confirms the accuracy of our previous analysis. Because id_2 does

[2] In Figure 6, E represents $\exists$, BT represents $\in$, and represents $\wedge$, and not represents $\neg$.



transition	preset	postset	guard	Time	dataoperation	dataschema
t0	0.	1.	0,0.	const,0.	wt,id.	sel,id.
t1	1.	2.	1,0.	const,0.	wt,nurse.	sel,nurse.
t2	2.	4.	0,1.	const,0.	rd,nurse.	sel,nurse.
t3	2.	3.	0,2.	const,0.	rd,nurse.	sel,nurse.
t4	3.	4.	0,0.	const,0.	rd,nurse.	sel,nurse.
t5	4.	5,7.	0,0.	const,0.	.	.
t6	5.	6.	0,0.	const,0.	rd,blood.	sel,blood.
t7	7.	8.	0,0.	const,0.	rd,ct.	sel,ct.
t8	6,8.	9.	0,0.	const,0.	.	.
t9	9.	10.	0,0.	const,0.	.	.
t10	10.	17.	0,0.	const,0.	.	.
t11	10.	11.	0,0.	const,0.	.	.
t12	11.	12.	1,0.	const,0.	wt,bed.	sel,bed.
t13	12.	15.	0,3.	const,0.	rd,bed.	sel,bed.
t14	12.	13.	0,4.	const,0.	rd,bed.	sel,bed.
t15	13.	14.	0,0.	const,0.	rd,bed.	sel,bed.
t16	14.	15.	0,0.	const,0.	rd,bed.	upd,bed.
t17	15.	16.	0,0.	const,0.	rd,bed.	sel,bed.
t18	16.	17.	0,0.	const,0.	.	.
t19	17.	18.	0,0.	const,0.	.	.

@

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
11	p11	0
12	p12	0
13	p13	0
14	p14	0
15	p15	0
16	p16	0
17	p17	0
18	p18	0

@

(a)

ctlpnr	relationop	timevalue	ctloperator	ctlformer	ctllatter
1	0	0	id1	0	0
2	0	0	id2	0	0
3	0	0	R	0	0
4	0	0	BT	1	3
5	0	0	E	0	4
6	0	0	BT	2	3
7	0	0	E	0	6
8	0	0	**and**	5	7
9	0	0	p5	0	0
10	0	0	AU	8	9

@

(d)

pipnr	piname	pidata	piguard	Gpi	NGpi
1	pi1	nurse1,nurse2.	1,2.	1.	2.
2	pi2	bed1,bed2.	3,4.	3.	4.

@

gpnr	gname	goperator	gformer	glatter
1	1	pi1	0	0
*				
1	2	pi1	0	0
2	2	**not**	0	1
*				
1	3	pi2	0	0
*				
1	4	pi2	0	0
2	4	**not**	0	1

@

cpnr	cname	coperator	cformer	clatter
1	1	g1	0	0
2	1	g2	0	0
3	1	**not**	0	1
4	1	**not**	0	2
5	1	**and**	3	2
6	1	**and**	1	4
7	1	**or**	5	6
*				
1	2	g3	0	0
2	2	g4	0	0
3	2	**not**	0	1
4	2	**not**	0	2
5	2	**and**	3	2
6	2	**and**	1	4
7	2	**or**	5	6

@

(b)

id	nurse	blood	ct	bed	&
id1	nurse1	blood1	ct1	bed1	
id2	nurse2	blood2	ct2	bed2	

@

(c)

(e)

Figure 6. a) A WFT-net; b) an initial table; c) guards; d) a TDCTL formula; e) the verification result

not complete the blood draw before the hospital's specified deadline, id_2 could not complete the blood draw and subsequent examination. However, in the WFT-net, where time constraints are not considered, id_2 is not bound by any time constraints, meaning the blood draw can be completed at any time. It is contrary to the actual hospital procedures.

6.2 Soundness Verification

To validate the effectiveness of our method and demonstrate that TDCTL can describe more design requirements. We convert soundness definitions into the corresponding TDCTL formulas. We have introduced 6 performance metrics (PMs) based on the motivating example. These PMs correspond to the design require-

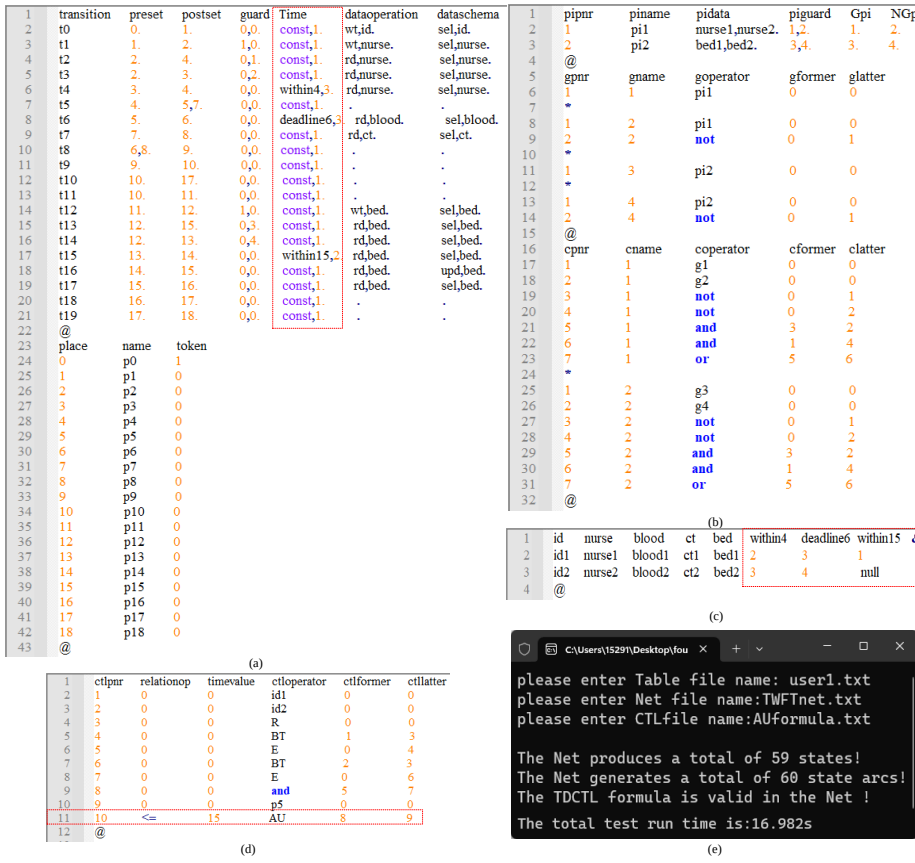


Figure 7. a) A TWFT-net; b) an initial table; c) guards; d) a TDCTL formula; (e) the verification result

ments in soundness and convert them into TDCTL formulas (researchers can also write corresponding PMs based on their design requirements). We have converted property ***P1*** into PM1 and ***P2*** into PM2. As for property ***P3***, since it involves data design requirements and time constraints, P3 consists of PM3, PM4, and PM5, i.e., $\boldsymbol{P3} = \{PM3, PM4, PM5\}$. We use our tool to validate various PMs and give their verification results. Detailed information about the PMs and experimental results are shown in Table 2.

The experimental results from Table 2 indicate that the PM2 and PM5 have incorrect calculations. It is because id_2 missed the blood draw time, resulting in a process failure.

Properties	PMs	TDCTL formulas	Examples	Results
P1	PM1	$\varphi_1 = AG(EFc_{end})$	$\varphi_1 = AG(EFc_{24})$	True
P2	PM2	$\varphi_2 = AFp_{end}$	$\varphi_2 = AFp_{18}$	False
P3	PM3	$\varphi_3 = AG(\forall(d_1 \in R, d_2 \in R), [d_1 \neq d_2] \rightarrow d_1^i \neq d_2^i)$	$\varphi_3 = AG(\forall(id_1 \in R, id_2 \in R), [id_1 \neq id_2] \rightarrow id_1.blood_1 \neq id_2.blood_2)$	True
	PM4	$\varphi_4 = A(\forall(d_1 \in R, d_2 \in R)U_{\leq time}[p])$	$\varphi_4 = A(\forall(id_1 \in R, id_2 \in R)U_{\leq 40}[p_{18}])$	False
	PM5	$\varphi_5 = AF_{\leq time}(d_1 \in R \rightarrow p_i)$	$\varphi_5 = AF_{\leq 15}(id_1 \in R \rightarrow p_5)$	True

Table 2. Performance Metrics

6.3 Experiments

In this part, we further do some experiments to compare the existing models in state space (i.e., the number of states and arcs), the construction time of SRG, and the soundness verification result. These experiments use the following benchmarks:

B1 is a system for merchants to handle customer complaints. It is required to conduct a questionnaire survey and evaluate the complaints within a specified time [53].

B2 is a system for merchants to handle orders. It represents a simple order processing, with four participants: the customer, the Front-office Service, The Production department and the Invoicing [54].

B3 is an example of a company seeking monthly write-ups from a selected set of employees for publication in its newsletter or on its website [55].

B4 is an online shopping system, and merchants will determine whether there is a malicious purchase based on the customer's credibility [56].

B5 is an electronic document management system. The system presented is used to manage medical certificates, check deadlines and validate documents [36].

B6 is a case of a loan application process where the staff decides whether to approve the loan based on the lender's information [19].

B7 is a job interview system. Human resources select candidates from among applicants [13][3].

B8 is our motivating example.

For each benchmark, we first use WFD-net, WFT-net, and TWFT-net to model these benchmarks in our tool and then respectively obtain their SRG. Furthermore, the soundness verification result is detected based on SRG. Each benchmark is tested 10 times, and the result of running time is their average.

Table 3 is the result of our experiments. It shows each model's state scale, construction time, and soundness results. It can be found from the experimental results in the table that since BM1, BM4, BM6, and BM8 did not consider the time elements, there are errors in the process model. Specifically, the reason for the error of BM4 is also related to the control flow in the model. After a buyer purchases a

[3] https://dbis.ipd.kit.edu/research_2134.php

BM	Model							SRG						Sound			Time
	WFD-net					WFT-net	TWFT-net	WFD-net		WFT-net		TWFT-net		WFD -net	WFT -net	TWFT -net	
	$\|T\|$	$\|P\|$	$\|F\|$	$\|D\|$	$\|G\|$	$\|R\|$	$\|DI(t)\|$	State	Arcs	State	Arcs	State	Arcs				
BM1	11	11	24	4	0	3	11	25	49	79	135	72	122	Yes	Yes	No	22.54
BM2	12	13	26	6	2	3	12	18	21	30	35	30	35	Yes	Yes	Yes	29.581
BM3	16	16	34	8	4	3	16	47	54	99	116	139	162	Yes	Yes	Yes	31.715
BM4	17	16	36	14	6	2	17	44	65	87	130	87	130	No	No	No	29.067
BM5	10	10	20	5	2	2	10	13	12	25	24	25	24	Yes	Yes	Yes	28.881
BM6	12	12	25	7	4	3	12	21	22	61	66	51	55	Yes	Yes	No	23.643
BM7	12	10	24	10	6	3	12	16	15	46	45	46	45	Yes	Yes	Yes	26.831
BM8	20	19	41	13	4	4	20	49	50	121	141	107	120	Yes	Yes	No	34.138

Table 3. Experimental results

product, the seller needs to verify the buyer's payment voucher and credit level. If the buyer's credit level is low, there may be a possibility of malicious purchases, and the seller rejects the buyer's purchase request, resulting in the transaction failure. Compared to WFD-net, which only considers abstract data elements, WFT-net considers the value of concrete data items without considering the time elements. TWFT-net considers the control flow, concrete data item values, and time elements, making it more accurate in detecting defects in the process model.

7 CONCLUSION

This paper gives a modeling method of a TWFT-net, which makes up for the lack of conventional modeling methods to describe the TBPMS. This model simulates TBPMS in actual scenarios by adding time elements in transitions. We extend the traditional CTL model checking operator, give a timed database computation tree logic model checking method, propose the corresponding algorithm, develop the related tool, and realize the soundness verification of a TBPMS. Our future work will mainly focus on three parts: (1) The TWFT-net does not involve resource scheduling issues. How to effectively and correctly utilize resources is a critical concern for future research; (2) We will optimize the TWFT-net to avoid the problem of state space explosion during the modeling process; and (3) Some tasks in the hospital have higher priority over other tasks, such as the severity of the patient's illness. Considering priority issues in the process model is also the main task of future research.

Acknowledgements

This paper was supported by the National Nature Science Foundation of China (Nos. 62172299, 62032019), the Space Optoelectronic Measurement and Perception Lab of Beijing Institute of Control Engineering (No. LabSOMP-2023-03) and the CCF-Huawei Populus Grove Fund (No. CCF-HuaweiFM202305).

REFERENCES

[1] Lapeña, R.—Pérez, F.—Pastor, Ó.—Cetina, C.: Leveraging Execution Traces to Enhance Traceability Links Recovery in BPMN Models. Information and Software Technology, Vol. 146, 2022, Art. No. 106873, doi: 10.1016/j.infsof.2022.106873.

[2] Li, Y.—Ding, Y.—Guo, Y.—Cui, H.—Gao, H.—Zhou, Z.—Zhang, N. A.—Zhu, S.—Chen, F.: An Integrated Decision Model with Reliability to Support Transport Safety System Analysis. Reliability Engineering & System Safety, Vol. 239, 2023, Art. No. 109540, doi: 10.1016/j.ress.2023.109540.

[3] van der Aalst, W. M. P.—De Masellis, R.—Di Francescomarino, C.—Ghidini, C.—Kourani, H.: Discovering Hybrid Process Models with Bounds on Time and Complexity: When to Be Formal and When Not? Information Systems, Vol. 116, 2023, Art. No. 102214, doi: 10.1016/j.is.2023.102214.

[4] Zhao, F.—Xiang, D.—Liu, G.—Jiang, C.: Behavioral Consistency Measurement Between Extended WFD-Nets. Information Systems, Vol. 119, 2023, Art. No. 102274, doi: 10.1016/j.is.2023.102274.

[5] Liu, G.: Petri Nets: Theoretical Models and Analysis Methods for Concurrent Systems. Springer Nature Singapore, 2022, doi: 10.1007/978-981-19-6309-4.

[6] Lonetti, F.—Bertolino, A.—Di Giandomenico, F.: Model-Based Security Testing in IoT Systems: A Rapid Review. Information and Software Technology, 2023, Art. No. 107326, doi: 10.1016/j.infsof.2023.107326.

[7] Zhou, J.—Reniers, G.—Cozzani, V.: A Petri-Net Approach for Firefighting Force Allocation Analysis of Fire Emergency Response with Backups. Reliability Engineering & System Safety, Vol. 229, 2023, Art. No. 108847, doi: 10.1016/j.ress.2022.108847.

[8] Estañol, M.—Sancho, M. R.—Teniente, E.: Ensuring the Semantic Correctness of a BAUML Artifact-Centric BPM. Information and Software Technology, Vol. 93, 2018, pp. 147–162, doi: 10.1016/j.infsof.2017.09.003.

[9] 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.

[10] Käppel, M.—Schönig, S.—Jablonski, S.: Leveraging Small Sample Learning for Business Process Management. Information and Software Technology, Vol. 132, 2021, Art. No. 106472, doi: 10.1016/j.infsof.2020.106472.

[11] Combi, C.—Oliboni, B.—Weske, M.—Zerbato, F.: Conceptual Modeling of Inter-Dependencies Between Processes and Data. Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC '18), 2018, pp. 110–119, doi: 10.1145/3167132.3167141.

[12] Xiang, D.—Liu, G.—Yan, C.—Jiang, C.: Detecting Data-Flow Errors Based on Petri Nets with Data Operations. IEEE/CAA Journal of Automatica Sinica, Vol. 5, 2018, No. 1, pp. 251–260, doi: 10.1109/JAS.2017.7510766.

[13] 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, https://www.ronpub.com/ojis-2014v1i2n01_stackelberg.html.

[14] Vogel, T.—Carwehl, M.—Rodrigues, G. N.—Grunske, L.: A Property Specification Pattern Catalog for Real-Time System Verification with UPPAAL. Information and Software Technology, Vol. 154, 2023, Art. No. 107100, doi: 10.1016/j.infsof.2022.107100.

[15] Pakonen, A.—Buzhinsky, I.—Björkman, K.: Model Checking Reveals Design Issues Leading to Spurious Actuation of Nuclear Instrumentation and Control Systems. Reliability Engineering & System Safety, Vol. 205, 2021, Art. No. 107237, doi: 10.1016/j.ress.2020.107237.

[16] Sbaï, Z.—Missaoui, A.—Barkaoui, K.—Ayed, R. B.: On the Verification of Business Processes by Model Checking Techniques. 2010 2nd International Conference on Software Technology and Engineering, IEEE, Vol. 1, 2010, pp. 97–103, doi: 10.1109/ICSTE.2010.5608905.

[17] Mönnich, H.—Raczkowsky, J.—Wörn, H.: Model Checking for Robotic Guided Surgery. In: Kostkova, P. (Ed.): Electronic Healthcare (eHealth 2009). Springer, Berlin, Heidelberg, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 27, 2010, pp. 1–4, doi: 10.1007/978-3-642-11745-9_1.

[18] 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.

[19] Trecka, N.—van der Aalst, W.—Sidorova, N.: Workflow Completion Patterns. 2009 IEEE International Conference on Automation Science and Engineering, 2009, pp. 7–12, doi: 10.1109/COASE.2009.5234170.

[20] Popova-Zeugmann, L.: Time and Petri Nets. Springer, 2013, doi: 10.1007/978-3-642-41115-1.

[21] Dotoli, M.—Fanti, M. P.—Iacobellis, G.—Martino, L.—Moretti, A. M.—Ukovich, W.: Modeling and Management of a Hospital Department via Petri Nets. 2010 IEEE Workshop on Health Care Management (WHCM), 2010, pp. 1–6, doi: 10.1109/WHCM.2010.5441248.

[22] He, L.—Liu, G.: Prioritized Time-Point-Interval Petri Nets Modelling Multi-Processor Real-Time Systems and TCTL$_x$. IEEE Transactions on Industrial Informatics.

[23] Bresolin, D.—Zavatteri, M.: Supervisory Control of Business Processes with Resources, Parallel and Mutually Exclusive Branches, Loops, and Uncertainty. Vol. 119, 2023, doi: 10.1016/j.is.2023.102288.

[24] Gouyon, D.—Pétin, J. F.—Cochard, T.—Devic, C.: Architecture Assessment for Safety Critical Plant Operation Using Reachability Analysis of Timed Automata. Reliability Engineering & System Safety, Vol. 199, 2020, Art. No. 106923, doi: 10.1016/j.ress.2020.106923.

[25] 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.

[26] Idel Mahjoub, Y.—Chakir El-Alaoui, E. H.—Nait-Sidi-Moh, A.: Modeling and Developing a Conflict-Aware Scheduling in Urban Transportation Networks. Future Generation Computer Systems, Vol. 107, 2020, pp. 1026–1036, doi: 10.1016/j.future.2018.04.022.

[27] Alur, R.—Dill, D. L.: A Theory of Timed Automata. Theoretical Computer Science, Vol. 126, 1994, No. 2, pp. 183–235, doi: 10.1016/0304-3975(94)90010-8.

[28] Van der Aalst, W. M. P.: Challenges in Business Process Management: Verification of Business Processes Using Petri Nets. Bulletin of the European Association for Theoretical Computer Science (EATCS), Vol. 80, 2003, No. 32, pp. 174–199.

[29] Geiger, M.—Harrer, S.—Lenhard, J.—Wirtz, G.: BPMN 2.0: The State of Support and Implementation. Future Generation Computer Systems, Vol. 80, 2018, pp. 250–262, doi: 10.1016/j.future.2017.01.006.

[30] Valderas, P.—Torres, V.—Serral, E.: Modelling and Executing IoT-Enhanced Business Processes Through BPMN and Microservices. Journal of Systems and Software, Vol. 184, 2022, Art. No. 111139, doi: 10.1016/j.jss.2021.111139.

[31] Felli, P.—de Leoni, M.—Montali, M.: Soundness Verification of Decision-Aware Process Models with Variable-to-Variable Conditions. 2019 19th International Conference on Application of Concurrency to System Design (ACSD), 2019, pp. 82–91, doi: 10.1109/ACSD.2019.00013.

[32] Batoulis, K.—Weske, M.: A Tool for Checking Soundness of Decision-Aware Business Processes. In: Clarisó, R., Leopold, H., Mendling, J., van der Aalst, W., Kumar, A., Pentland, B., Weske, M. (Eds.): Proceedings of the BPM Demo Track and BPM Dissertation Award Co-Located with 15th International Conference on Business Process Management (BPM 2017). CEUR Workshop Proceedings, Vol. 1920, 2017, pp. 1–5, https://ceur-ws.org/Vol-1920/BPM_2017_paper_184.pdf.

[33] 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.

[34] Morimoto, S.: A Survey of Formal Verification for Business Process Modeling. In: Bubak, M., van Albada, G. D., Dongarra, J., Sloot, P. M. A. (Eds.): Computational Science – ICCS 2008. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5102, 2008, pp. 514–522, doi: 10.1007/978-3-540-69387-1_58.

[35] Wang, J.—Wang, J.: Real-Time Adaptive Allocation of Emergency Department Resources and Performance Simulation Based on Stochastic Timed Petri Nets. IEEE Transactions on Computational Social Systems, Vol. 10, 2023, No. 4, pp. 1986–1996, doi: 10.1109/TCSS.2023.3266501.

[36] Ramos, D. S.—Rocha, F.—Soares, M. d. S.: A Bottom Up Approach for Modeling Business Process Using Time Petri Nets. Proceedings of the XVIII Brazilian Symposium on Information Systems (SBSI '22), 2022, doi: 10.1145/3535511.3535539.

[37] Nguyen Thanh, T.—Le Thanh, N.—Hoang Thi Thanh, H.: Formalization of Business Processes and Business Rules Model Using Colored Petri Nets. Proceedings

of the 5th International Conference on Future Networks and Distributed Systems (ICFNDS '21), 2021, pp. 42–47, doi: 10.1145/3508072.3508080.

[38] Zhu, A.—Sala, A.—Wang, J.: Colored Petri Nets Based Patient Flow Modeling and Optimal Staffing in Emergency Healthcare. 2022 International Conference on Cyber-Physical Social Intelligence (ICCSI), 2022, pp. 686–691, doi: 10.1109/ICCSI55536.2022.9970562.

[39] Song, W.—Ma, X.—Cheung, S. C.—Hu, H.—Lü, J. J.: Preserving Data Flow Correctness in Process Adaptation. 2010 IEEE International Conference on Services Computing, 2010, pp. 9–16, doi: 10.1109/SCC.2010.24.

[40] 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, pp. 333–363, doi: 10.1007/s00165-010-0161-4.

[41] Liu, W.—Du, Y.—Yan, C.: Soundness Preservation in Composed Logical Time Workflow Nets. Enterprise Information Systems, Vol. 6, 2012, No. 1, pp. 95–113, doi: 10.1080/17517575.2011.617472.

[42] Barkaoui, K.—Ayed, R. B.—Sbaï, Z.: Workflow Soundness Verification Based on Structure Theory of Petri Nets. International Journal of Computing and Information Sciences, Vol. 5, 2007, No. 1, pp. 51–61.

[43] 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 (in Chinese).

[44] He, L.—Liu, G.—Zhou, M.: Petri-Net-Based Model Checking for Privacy-Critical Multiagent Systems. IEEE Transactions on Computational Social Systems, Vol. 10, 2022, No. 2, pp. 563–576, doi: 10.1109/TCSS.2022.3164052.

[45] Song, J.—Xiang, D.—Liu, G.—He, L.: Guard-Function-Constraint-Based Refinement Method to Generate Dynamic Behaviors of Workflow Net with Table. Computing and Informatics, Vol. 41, 2022, No. 4, pp. 1025–1053, doi: 10.31577/cai_2022_4_1025.

[46] 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.

[47] Liu, G.—Jiang, C.—Zhou, M.: Time-Soundness of Time Petri Nets Modelling Time-Critical Systems. ACM Transactions on Cyber-Physical Systems, Vol. 2, 2018, No. 2, Art. No. 11, doi: 10.1145/3185502.

[48] Clarke, E. M.—Emerson, E. A.: Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic. In: Grumberg, O., Veith, H. (Eds.): 25 Years of Model Checking: History, Achievements, Perspectives. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5000, 2008, pp. 196–215, doi: 10.1007/978-3-540-69850-0_12.

[49] Baier, C.—Katoen, J. P.: Principles of Model Checking (Representation and Mind Series). The MIT Press, 2008.

[50] Lohmann, N.: Compliance by Design for Artifact-Centric Business Processes. Information Systems, Vol. 38, 2013, No. 4, pp. 606–618, doi: 10.1016/j.is.2012.07.003.

[51] Clarke, E. M.: Model Checking. In: Ramesh, S., Sivakumar, G. (Eds.): Foundations of Software Technology and Theoretical Computer Science (FSTTCS 1997). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1346, 1997, pp. 54–56, doi: 10.1007/BFb0058022.

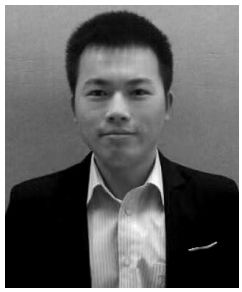
[52] Liu, G.: Some Complexity Results for the Soundness Problem of Workflow Nets. IEEE Transactions on Services Computing, Vol. 7, 2013, No. 2, pp. 322–328, doi: 10.1109/TSC.2013.36.

[53] van der Aalst, W. M. P.: Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. In: van der Aalst, W., Desel, J., Oberweis, A. (Eds.): Business Process Management: Models, Techniques, and Empirical Studies. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1806, 2002, pp. 161–183, doi: 10.1007/3-540-45594-9_11.

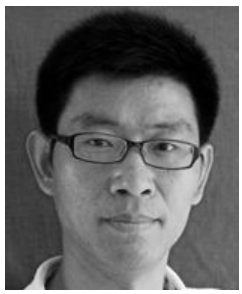
[54] Rachdi, A.—En-Nouaary, A.—Dahchour, M.: DataFlow Analysis in BPMN Models. Proceedings of the 19th International Conference on Enterprise Information Systems - Volume 2: ICEIS, 2017, pp. 229–237, doi: 10.5220/0006271202290237.

[55] 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.

[56] Wang, P.—Liu, W.—Du, Y.: Business Process Modeling and Analysis Based on Logical Data Petri Net. Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems (CIMS), Vol. 23, 2017, No. 5, pp. 921–930, doi: 10.13196/j.cims.2017.05.001 (in Chinese).



Jian Song 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 his 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, control-flow, and data-flow.



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.