IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 28, 2017, accepted February 6, 2018, date of publication February 16, 2018, date of current version March 16, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2806884

Verifying the Correctness of Workflow Systems Based on Workflow Net With Data Constraints

YAQIONG HE, GUANJUN LIU, (Member, IEEE), DONGMING XIANG, JIAQUAN SUN, CHUNGANG YAN, AND CHANGJUN JIANG

[1]Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 201804, China
[2]Shanghai Electronic Transactions and Information Service Collaborative Innovation Center, Tongji University, Shanghai 201804, China
[3]Department of Computer Science, Tongji University, Shanghai 201804, China

Corresponding authors: Guanjun Liu (liuguanjun@tongji.edu.cn) and Changjun Jiang (cjjiang@tongji.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1001804, in part by the Shanghai Science and Technology Innovation Action Plan Project under Grant 16511100900, and in part by the National Natural Science Foundation of China under Grant 61572360.

ABSTRACT The correctness verification is very important for workflow systems. It is closely related with both control-flows and data-flows. Workflow nets with data (WFD-nets) are a kind of formal model that can reflect some logical structures of workflow systems, e.g., choice and concurrency, and represent some operations on data, e.g., read, write, and delete. However, these data operations are conceptual in WFD-nets and only characterize the logical relation between two operations, e.g., whether a write and a read are concurrently operating on a data. They do not consider the functional requirements about data (i.e., data constraints). Thus, some data errors cannot be found via WFD-nets. In order to solve this problem, we propose Workflow nets with Data Constraints (WFDC-nets) and define four levels of soundness to describe different correctness requirements. Based on the reachability graphs of WFDC-nets, we verify the soundness. The related algorithms are proposed and a tool is developed to show the effectiveness and usefulness of our method.

INDEX TERMS Workflow systems, data constraints, Petri nets, soundness.

I. INTRODUCTION

The design of workflow systems becomes more complicated due to a large amount of business logics and activities [1], [2], which results in a more difficult analysis and verification of systems' correctness. There are many studies about the correctness verification of workflow systems such as model checking [3], [4], formal specifications [5] and theorem proving techniques [6]. As a prominent formal model, workflow nets (WF-nets) that are a class of Petri nets are suitable to represent the business logics of workflow systems [7]. The soundness of WF-nets is an important property which guarantees that a system can always terminate once it runs. In other words, soundness guarantees that a system has neither deadlock nor livelock [8], [9]. Sometimes, the concept of soundness becomes looser according to different system requirements such as weak soundness, relaxed soundness, k-soundness and generalized soundness [10]–[13]. However, these definitions of soundness ignore the data operated in the execution process of a system.

In practical application (e.g., e-commerce transaction system and financial management system), data plays an important role. There exists the case that the business process of a system is sound from the view of aspect of control-flow but is not correct if data-flow is considered, e.g., missing data, inconsistent data and conflict data [14]. These data errors bring a big challenge for system design. Many studies focus on data-flows in workflow systems. Sadiq *et al.* [14] define several data abnormalities that may lead to an incorrect execution of a workflow system. Some model-checking-based methods are given in [3], [15], and [16].. Sun *et al.* [15] provide a data-flow perspective to detect data-flow errors, which includes data-flow specification and data-flow analysis. Xiang *et al.* [17] check the data inconsistency in concurrent systems by Petri nets with data operations. Du *et al.* [18] propose a subclass of logic Petri nets (LPNs) to model and detect the indeterminacy of passing data in e-commerce systems. Artifact systems are proposed in [19] and [20] as data-centric models to verify some desirable data properties. There are also some methods that analyze the correctness of data-flows based on the soundness of extended WF-nets. Trcka *et al.* [21] propose a workflow net with data (WFD-net) to detect data-flow errors, which adds data elements, guards

and data operations (e.g., read, write and delete) to a WF-net. The soundness of WFD-net is further studied in [22], [23], and [24] that extends the classical soundness to must-/may-soundness with respect to different data refinements. Wang *et al.* [25] propose a more refined workflow net with transition conditions (WTC-net) based on a WFD-net to further analyze the refined data.

Although these studies provide different methods to detect the data-flow errors, they do not consider the values of data. In other words, some errors of control-/data-flows can be reflected only if the values of data are taken into account. For example, a payment requirement from a shopper is delivered to a third-party cashier which is bond to a merchant in an e-commerce transaction system [26]. If the merchant does not verify the payment notification from the third-party cashier, malicious shoppers may falsify the total price of goods before the payment. As a consequence, the merchant loses money. In this transaction system, there is no deadlock or livelock (i.e., it is sound). However, its business process lacks an activity of checking whether the amount of money in the payment is equal to the total price of goods before the payment is done. Therefore, the correctness of the business process of the system can be verified only if the data constraints are considered in the model.

In order to solve the above problem, we define a Workflow Net with Data Constraints (WFDC-net). It not only formalizes the abstract data operations, but also considers data constraints. Furthermore, we propose four levels of soundness named as soundness, control-soundness, data-soundness and non-soundness, and use them to reflect different correctness requirements. We describe the related algorithms and develop a tool to check them.

The remaining paper is organized as follows. Section II introduces some concepts used in this paper. Section III gives a motivated example of a car management system. Section IV defines WFDC-nets. Section V constructs the reachability graph of a WFDC-net. Section VI formalizes and verifies four levels of soundness based on the reachability graph. Section VII introduces our tool. The last section concludes this paper.

II. BASIC CONCEPTS

A Petri net [7], [27] is a triple $N = (P, T, F)$, where P and T are disjoint sets of places and transitions, respectively; and $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation.

For a node $x \in P \cup T$, ${}^\bullet x = \{y \mid (y, x) \in F\}$ denotes the *preset* of x, and $x^\bullet = \{y \mid (x, y) \in F\}$ denotes its *postset*. A *marking* is a mapping $M: P \to \mathbb{N}$, where $\mathbb{N} = \{0, 1, 2, \cdots\}$ is the set of non-negative integers. A marking is usually represented by a multiset. For example, $M = [p_1, 2p_2]$ is a marking, where $M(p_1) = 1, M(p_2) = 2$ and $\forall p \in P \backslash \{p_1, p_2\} : M(p) = 0$.

A transition t is enabled at a marking M if $M \geq {}^\bullet t$, denoted as $M[t\rangle$. After firing t, a new marking M' is generated, where $M' = M - {}^\bullet t + t^\bullet$. It is denoted as $M[t\rangle M'$. If a marking M'' is generated after firing a transition sequence σ at M, it is denoted as $M[\sigma\rangle M''$. The set of all markings that are reachable from M_0 is denoted by $R(M_0)$.

A workflow net is a special Petri net, which can model the business process of a workflow system. It has only one initial place and one terminal place. It requires a strong connection if one transition and two arcs are added to connect from the initial place to the terminal place.

Definition 1 (WF-Net [9]): A Petri net $N = (P, T, F)$ is a workflow net (WF-net) if

(1) *it has a source place i and a sink place o such that ${}^\bullet i = \emptyset \wedge o^\bullet = \emptyset$; and*

(2) *if a transition $\varepsilon \notin T$ is added into N, then we get a new Petri net N^* where ${}^\bullet i = \varepsilon \wedge o^\bullet = \varepsilon$, and N^* is strongly connected.*

Soundness is an important property of workflow nets that guarantees that a system has no deadlock, livelock or dead transition.

Definition 2 (Soundness of WF-Net [9]): Let $N=(P,T,F)$ be a WF-net. $[i]$ and $[o]$ denote the initial marking and terminal marking, respectively. N is sound if it satisfies

(1) $\forall M \in R([i]) : [o] \in R(M)$;

(2) $\forall M \in R([i]) : M \geq [o] \Rightarrow M = [o]$; *and*

(3) $\forall t \in T, \exists M \in R([i]) : M[t\rangle$.

In fact, the soundness of WF-nets only focuses on the correctness of control-flows, but does not consider the errors of data-flows. Even though a workflow system is sound, it does not necessarily satisfy the correctness of data-flows. A WFD-net extends a workflow net with conceptual data operations on refined data including read, write and delete. The concept of refinement was originally defined in [28] and later extended in system verifications [29]. It is a conceptual notion that a concrete specification can be substituted for an abstract one if its behavior is consistent with the abstract one. To introduce WFD-nets, we first introduce some notations associated with them. These notations come from in [24].

$D = \{d_1, \cdots, d_{|D|}\}$ is a set of data elements. $\Pi = \{{\pi_i}^d | i \in \mathbb{N}_k, d \in D\}$ is a set of propositions [30], where $\mathbb{N}_k = \{1, 2, \cdots, k\}$, and ${\pi_i}^d$ denotes an atomic propositional formula related to data d. For example, ${\pi_1}^{d_1}$, ${\pi_1}^{d_2}$ and ${\pi_2}^{d_2}$ are three propositional formulas. ${\pi_1}^{d_1}$ and ${\pi_1}^{d_2}$ have the same propositional function with different variables (i.e., data d_1 and d_2). ${\pi_1}^{d_2}$ and ${\pi_2}^{d_2}$ have different propositional functions with the same variable (i.e., data d_2). G_Π is a set of guards and every guard is a compound propositional formula.

Based on these notations, a WFD-net is defined to model some conceptual data operations.

Definition 3 (WFD-Net [24]): A 9-tuple $WD = (P, T, F, D, Rd, Wt, De, grd, G_\Pi)$ is a workflow net with data (WFD-net), where

(1) *(P, T, F) is a WF-net;*

(2) *D is a finite set of data elements;*

(3) *$Rd: T \to 2^D$ is a label function of reading data;*

(4) *$Wt: T \to 2^D$ is a label function of writing data;*

(5) *$De: T \to 2^D$ is a label function of deleting data; and*

(6) *grd*:$T \rightarrow G_\Pi$ *is a guard function of assigning a guard to each transition.*

III. A MOTIVATING EXAMPLE

Our work is motivated by a car management system. In a university, a car of every staff is authorized to have one (and only one) permit-card that records some information such as the plate number of the car. Only when a car has a permit-card, it can be permitted to enter the campus of the university. When a user logs in the system, an enquiry is needed to verify if s/he has registered her/his car or not. If the user has registered her/his car, s/he can modify some information before the permit-card has been made. Note that the process of approving and making a permit-card is omitted in our model for simplifications. If s/he has not registered her/his car, s/he needs to input the related information (e.g., the plate number of the car and her/his license) into the system. If the related car information is legal (e.g., the plate number has not been registered before), it allows the user to go forward. Otherwise, the information is required to input again. When s/he goes to the next step, s/he needs to upload some documents' copies such as driving license. In the similar way, if the documents are legal, the system accepts the application, and the user can log out. Otherwise, the user is required to resubmit these documents. Due to the fact that the modifying procedure is similar with the process of registration, we do not introduce it again.

Fig. 1 is a WFD-net of the motivated example. There are data elements, conceptual data operations and guards in it. The data set is $D = \{reg, cid, doc\}$, where *reg*, *cid* and *doc* represent the registering record, car plate number and document materials, respectively. $\Pi = \{Exist(reg), isUnique(cid), isLegal(doc)\}$ and $G_\Pi = \{Exist(reg), \neg Exist(reg), isUnique(cid), \neg isUnique(cid), isLegal(doc), \neg isLegal(doc)\}$. *Rd*, *Wt* and *De* represent read, write and delete operations, respectively. However, these refined data and conceptual data operations in WFD-nets do not reflect the values of data, *e.g.*, whether the car information is correct after a write operation is done. It is necessary to verify the car information and its document materials.

The car management system has some constraints, e.g., it requires that a plate number can be registered only once and then correspond to only one permit-card. In this workflow system, if these constraints on data items are not verified, they easily result in some abnormal data. For example, a user has registered the plate number of a car, and a new user first goes to register the plate number of another car. Later, the new user modifies the new plate number into the previous one. Because the modification process lacks the step of checking if the modified information is legal, the new user's modification can also be approved, i.e., the previous plate number are registered again.

This problem exists not only in the car management system, but also in other workflow systems, such as e-commerce transaction systems, financial management systems and delivery systems. The dissatisfaction of data

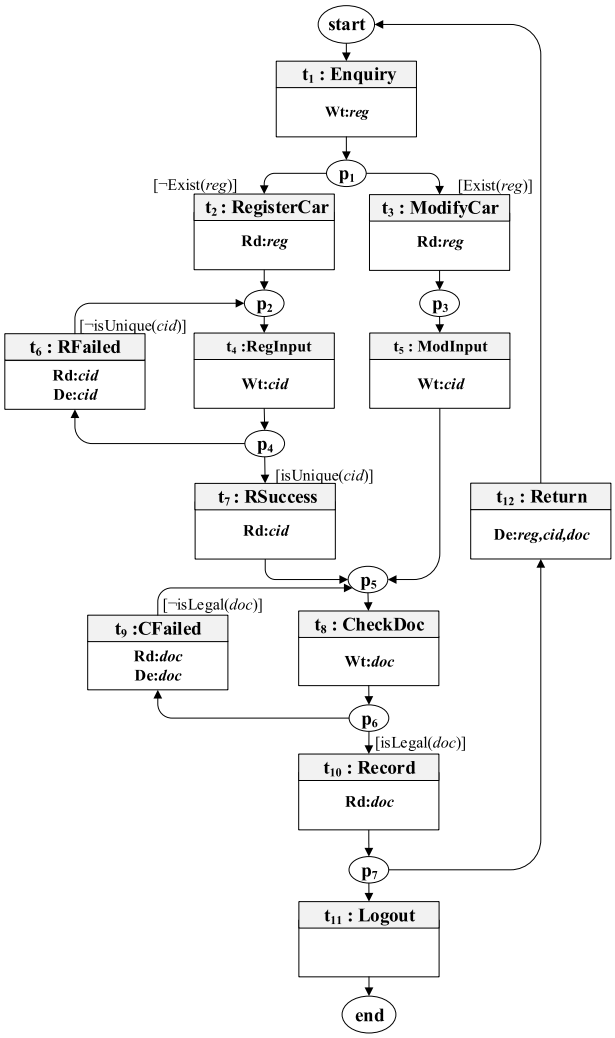


FIGURE 1. A WFD-net of the car management system.

constraints can potentially cause some fund losses [31], [32]. The traditional verification methods only check whether the business logics of workflow systems are correct or not. Therefore, although the soundness of the WFD-net modeling a workflow system can guarantee that the business process of the system is deadlock-free and livelock-free, it cannot indicate that its business process is correct, just as shown in our motivation example. This paper does not only focus on a correct termination of workflow processes, but also verifies some data constraints, *e.g.*, whether documents are legal and car information is not equal to others in database. In the following sections, we will give our formal model, extend the soundness properties and verify them.

IV. WORKFLOW NET WITH DATA CONSTRAINTS

In order to solve the above problem, we propose a WFDC-net that is obtained by adding a set of data constraints into a WFD-net. Fig. 3 shows the modeling framework of our WFDC-net.

In WFDC-nets, we consider both routing paths and constraints of data elements, and a data item is allowed to

be associated with different constraints. Assume that $\Psi = \{{\psi_i}^d | i \in \mathbb{N}_k, d \in D\}$ is a set of atomic propositional formulas, where $\mathbb{N}_k = \{1, 2, \cdots, k\}$ is a finite set of non-negative integers. $\Pi \subseteq \Psi$ is a subset of Ψ which includes all atomic propositional formulas in guards. $\Phi \subseteq \Psi$ represents a data constraint set. For convenience, we formalize $\Pi = \{{\pi_m}^d | m \in \mathbb{N}_{k_1}, d \in D\}$ and $\Phi = \{{\varphi_n}^d | n \in \mathbb{N}_{k_2}, d \in D\}$, where $k_1 \leq k$ and $k_2 \leq k$. Thus, for any $d \in D$, if ${\psi_i}^d = {\pi_m}^d$ and ${\psi_i}^d = {\varphi_n}^d$, then ${\pi_m}^d = {\varphi_n}^d$.

For example, in the car management system, $\Pi = \{{\pi_1}^{reg} : Exist(reg), {\pi_2}^{cid} : isUnique(cid), {\pi_3}^{doc} : isLegal(doc)\}$. The data constraints include: (1) the value of data *cid* is unique; and (2) the value of data *doc* is legal. Thus, $\Phi = \{{\varphi_1}^{cid} : isUnique(cid), {\varphi_2}^{doc} : isLegal(doc)\}$. Their union set is denoted as $\Psi = \{{\psi_1}^{reg} : Exist(reg), {\psi_2}^{cid} : isUnique(cid), {\psi_3}^{doc} : isLegal(doc)\}$. For readability, we use $isUnique(cid)$ to substitute a $\pi^{cid} : isUnique(cid)$ and $isUnique(cid)_\varphi$ to substitute a constraint $\varphi^{cid} : isUnique(cid)$. Thus, $\Pi = \{ Exist(reg), isUnique(cid), isLegal(doc)\}$ and $\Phi = \{ isUnique(cid)_\varphi, isLegal(doc)_\varphi\}$.

During the execution of a workflow system, data constraints are attached to different related data elements, and using them we can check whether the data satisfies the data constraints.

In order to describe the execution of a workflow system with some data constraints, we further define a constraint pattern based on Φ.

Definition 4 (Constraint Pattern Pa(Φ)): A triple Pa(Φ) $= (D, \Phi, \Gamma_D)$ is a constraint pattern, where

(1) *D is the data set;*

(2) *Φ is the data constraint set; and*

(3) *$\Gamma_D : D \rightarrow 2^\Phi$ is a mapping function that that maps each data in D to a set of constraints.*

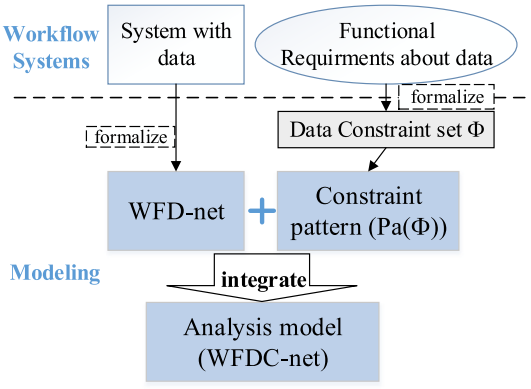


FIGURE 2. A WFDC-net of the car management system.

For example in Fig. 2, $\Phi = \{ isUnique(cid)_\varphi, isLegal(doc)_\varphi\}$ where $cid, doc \in D$. $\Gamma_D(cid) = isUnique(cid)_\varphi$ and $\Gamma_D(doc) = isLegal(doc)_\varphi$. Once a constraint pattern is given, the verification can be conducted to check whether a workflow system satisfies some data constraints.

Based on WFD-net and data constraint pattern, a WFDC-net is defined as follows.

Definition 5 (WFDC-Net): A 2-tuple $N = (WD, Pa(\Phi))$ is a workflow net with data constraints (WFDC-net), where

(1) *WD is a WFD-net;*

(2) *Pa(Φ) is a constraint pattern.*

As shown in Fig. 2, it is the WFDC-net modeling our motivated example, which formalizes the set of constraints Φ and adds constraint pattern *Pa*(Φ) to the WFD-net in Fig. 1.

V. REACHABILITY

Since a WFDC-net models both control-flows and data-flows of a workflow system, its states should include markings and data information. Furthermore, the data constraints are also considered in the states of WFDC-nets. In this paper, a state of WFDC-net is called a configuration.

In order to introduce configuration, we first formalize some notations. $\rho_D{:}D \rightarrow \{\top, \bot\}$ is an assigning function, where $\top$ represents a defined value and $\bot$ means an undefined value. An assigning function $\rho_\Pi : \Pi \rightarrow \{TRUE, FALSE, \bot\}$ denotes that an atomic propositional formula $\pi \in \Pi$ is assigned to *TRUE*, *FALSE* or $\bot$ (undefined). A mapping function $\ell_\Pi : \Pi \rightarrow D$ represents a relationship between all propositional formulas in Π and data elements in *D*. It is assumed that $\rho_\Pi(\pi) = \bot$ if $\exists d \in D : \ell_\Pi(\pi) = d \wedge \rho_D(d) = \bot$. An assigning function $\rho_\Phi : \Phi \rightarrow \{TRUE, FALSE, \bot\}$ denotes that a constraint of data *d* is assigned to *TRUE* (satisfied), *FALSE* (not satisfied) or $\bot$ (undefined). A mapping function $\ell_\Phi : \Phi \rightarrow D$ represents a relationship between all data constraints in Φ and data elements in *D*. Similarly, if $\exists d \in D : \Gamma_D(d) \neq \emptyset \wedge \ell_\Phi(\varphi) = d \wedge \rho_D(d) = \bot$, then $\rho_\Phi(\varphi) = \bot$. Furthermore, $\rho_G : G_\Pi \rightarrow \{TRUE, FALSE, \bot\}$ means that the evaluation of guards is *TRUE*, *FALSE* or $\bot$ (undefined). A mapping function $\ell_G : G \rightarrow 2^\Pi$ assigns an element in 2^Π for guard in *G*.

A triple $\rho = (\rho_D, \rho_\Phi, \rho_\Pi)$ is called a data state, and we use $\mathcal{P}$ to represent the set of all data states.

Definition 6 (Configuration): Let $N = (WD, Pa(\Phi))$ be a WFDC-net and $WD = (P, T, F, D, Rd, Wt, De, grd, G_\Pi)$. $c = \langle M, \rho\rangle = \langle M, \rho_D, \rho_\Phi, \rho_\Pi\rangle$ is a configuration of N, where

(1) *M is a marking of WD;*

(2) *$\rho_D{:}\ D \rightarrow \{\top, \bot\}$ is a mapping function that evaluates the data element as $\top$ (a defined value) or $\bot$ (an undefined value) ;*

(3) *$\rho_\Phi{:}\ \Phi \rightarrow \{TRUE, FALSE, \bot\}$ assigns TRUE (satisfied), FALSE(not satisfied) or $\bot$ (an undefined value) to each $\varphi \in \Phi$; and*

(4) *$\rho_\Pi{:}\ \Pi \rightarrow \{TRUE, FALSE, \bot\}$ assigns TRUE, FALSE or $\bot$ (an undefined value) to each $\pi \in \Pi$.*

The initial configuration of N is $c_0 = \langle[i], \rho_D, \rho_\Phi, \rho_\Pi\rangle$ satisfying $\forall d \in D, \forall\varphi \in \Phi, \forall\pi \in \Pi : \rho_D(d) = \bot \wedge \rho_\Phi(\varphi) = \bot \wedge \rho_\Pi(\pi) = \bot$, and the terminal configuration set is $C_f = \{\langle[o], \rho\rangle | \rho \in \mathcal{P}\}$.

For example, the initial configuration of the WFDC-net in Fig. 2 is

$$c_0 = \langle [start], \{\rho_D(reg) = \bot, \rho_D(cid) = \bot, \rho_D(doc) = \bot\}, \{\rho_\Phi(isUnique(cid)_\varphi) = \bot, \rho_\Phi(isLegal(doc)_\varphi) = \bot\}, \{\rho_\Pi(Exist(reg)) = \bot, \rho_\Pi(isUnique(cid)) = \bot, \rho_\Pi(isLegal(doc)) = \bot\}\rangle.$$

It is simplified as $c_0 = \langle [start], \{\bot, \bot, \bot\}, \{\bot, \bot\}, \{\bot, \bot, \bot\}\rangle$. In general, a WFDC-net has one unique initial configuration but possibly more terminal configurations due to different data states.

In order to analyze a WFDC-net, we give its enabling and firing rules.

Definition 7 (Enabling Rule): Let $N = (WD, Pa(\Phi))$ be a WFDC-net and $WD = (P, T, F, D, Rd, Wt, De, grd, G_\Pi)$. A transition $t \in T$ is enabled at the configuration $c = \langle M, \rho_D, \rho_\Phi, \rho_\Pi\rangle$, denoted by $c[t\rangle$ if

(1) $M[t\rangle$;
(2) $\forall d \in Rd(t) \cup De(t) : \rho_D(d) = \top$;
(3) $\forall \pi \in \ell_G(grd(t)) : \rho_\Pi(\pi) \neq \bot$; *and*
(4) $\rho_G(grd(t)) = TRUE$.

In the WFDC-net of Fig. 2, the transition t_7 has a guard $grd(t_7) = [isUnique(cid)]$, where $\ell(isUnique(cid)) = \{cid\}$. $c = \langle [p_4], \{\top, \top, \bot\}, \{TRUE, \bot\}, \{TRUE, TRUE, \bot\}\rangle$ is a configuration. t_7 is enabled at the marking $[p_4]$, and $\rho_\Pi(isUnique(cid)) = TRUE$. We can find that $\rho_G(grd(t_7)) = TRUE$. Thus, t_7 is enabled at the configuration c.

Definition 8 (Firing Rule): Let $N = (WD, Pa(\Phi))$ be a WFDC-net and $WD = (P, T, F, D, Rd, Wt, De, grd, G_\Pi)$. A set of configurations C' is generated after firing an enabled transition $t \in T$ at the configuration $c = \langle M, \rho_D, \rho_\Phi, \rho_\Pi\rangle$, where

$$\begin{aligned} C' = \{\langle M', \rho_D', \rho_\Phi', \rho_\Pi'\rangle \mid & M[t\rangle M' \\ & \wedge(\forall d \in Wt(t) : \rho_D'(d) = \top) \\ & \wedge(\forall d \in De(t) : \rho_D'(d) = \bot) \\ & \wedge(\forall d \in D \setminus (Wt(t) \cup De(t)) : \rho_D'(d) = \rho_D(d)) \\ & \wedge(\forall \varphi \in \Phi : \ell_\Phi(\varphi) \in Wt(t) \\ & \Rightarrow \rho_\Phi'(\varphi) \in \{TRUE, FALSE\}) \\ & \wedge(\forall \varphi \in \Phi : \ell_\Phi(\varphi) \in De(t) \Rightarrow \rho_\Phi'(\varphi) = \bot) \\ & \wedge(\forall \varphi \in \Phi : \ell_\Phi(\varphi) \in D \setminus (Wt(t) \cup De(t)) \\ & \Rightarrow \rho_\Phi'(\varphi) = \rho_\Phi(\varphi)) \\ & \wedge(\forall \pi \in \Pi : (\forall d \in \ell_\Pi(\pi) \cap Wt(t) : \Gamma_D(d) = \emptyset) \\ & \Rightarrow \rho_\Pi'(\pi) \in \{TRUE, FALSE\}) \\ & \wedge(\forall \pi \in \Pi : (\forall d \in \ell_\Pi(\pi) \cap Wt(t) : \Gamma_D(d) \neq \emptyset) \\ & \Rightarrow \rho_\Pi'(\pi) = \omega) \\ & \wedge(\forall \pi \in \Pi : \ell_\Pi(\pi) \in De(t) \Rightarrow \rho_\Pi'(\pi) = \bot) \\ & \wedge(\forall \pi \in \Pi : \ell_\Pi(\pi) \in D \setminus (Wt(t) \cup De(t)) \\ & \Rightarrow \rho_\Pi'(\pi) = \rho_\Pi(\pi))\} \end{aligned}$$

where

$$\omega \in \begin{cases} \{TRUE\}, & if \exists \varphi \in \Gamma_D(d) : \pi = \varphi \wedge \rho_\Phi'(\varphi) = TRUE; \\ \{FALSE\}, & if \exists \varphi \in \Gamma_D(d) : \pi = \varphi \wedge \rho_\Phi'(\varphi) = FALSE; \\ \{TRUE, FALSE\}, & if \exists \varphi \in \Gamma_D(d) : \pi \neq \varphi. \end{cases}$$

It is denoted by $c[t\rangle C'$.

After firing an enabled transition t at a configuration c, a set of new configurations are generated. Their markings and data states not only satisfy the firing rules of WFD-nets, but also consider the transformation of data constraints.

For example of $c_0 = \langle [start], \{\bot, \bot, \bot\}, \{\bot, \bot\}, \{\bot, \bot, \bot\}\rangle$ in Fig. 2, when the enabled transition t_1 is fired, data *reg* is written into the system. According to Definition 8, we know that $\rho_D'(d) = \top$ and $\rho_\Pi'(Exist(reg)) \in \{TRUE, FALSE\}$ since $\Gamma_D(reg) = \emptyset$. The generated $C' = \{c_1, c_2\}$, where $c_1 = \langle [p_1], \{\top, \bot, \bot\}, \{\bot, \bot\}, \{TRUE, \bot, \bot\}\rangle$ and $c_2 = \langle [p_1], \{\top, \bot, \bot\}, \{\bot, \bot\}, \{FALSE, \bot, \bot\}\rangle$.

For another example, the transition t_4 is enabled at the configuration $c_3 = \langle [p_2], \{\top, \bot, \bot\}, \{\bot, \bot\}, \{FALSE, \bot, \bot\}\rangle$. According to Definition 8, data *cid* is written into system after firing t_4, $\rho_D'(cid) = \top$. Since $\Gamma_D(cid) = \{isUnique(cid)_\varphi\}$, we can get $\rho_\Phi'(isUnique(cid)_\varphi) \in \{TRUE, FALSE\}$. There exists a $\pi = isUnique(cid)$ which is equal to the data constraint $isUnique(cid)_\varphi$, then $\rho_\Pi'(isUnique(cid)) = TRUE$ when $\rho_\Phi'(isUnique(cid)_\varphi) = TRUE$ and $\rho_\Pi'(isUnique(cid)) = FALSE$ when $\rho_\Phi'(isUnique(cid)_\varphi) = FALSE$, i.e., a configuration set $C'' = \{c_4, c_5\}$ is generated, where $c_4 = \langle [p_4], \{\top, \top, \bot\}, \{TRUE, \bot\}, \{FALSE, TRUE, \bot\}\rangle$, and $c_5 = \langle [p_4], \{\top, \top, \bot\}, \{FALSE, \bot\}, \{FALSE, FALSE, \bot\}\rangle$.

Definition 9 (Reachability): Let $N = (WD, Pa(\Phi))$ be a WFDC-net and $WD = (P, T, F, D, Rd, Wt, De, grd, G_\Pi)$. c, c' and c'' are configurations of N. C, C' and C'' are configuration sets of N.

(1) *There is a **may-step** from c to c', denoted by $c \rightarrow_{may} c'$, if $\exists t \in T$: $c[t\rangle c'$;*

(2) *c'' is **may-reachable** from c if there is a sequence of configuration $c_0, \cdots, c_i, \cdots, c_n$ such that $c_i \rightarrow_{may} c_{i+1}$, where $0 \leq i < n$. It is denoted by $c \xrightarrow{*}_{may} c''$ with $c_0 = c$ and $c_n = c''$;*

(3) *There is a **must-step** from c to C, denoted by $c \rightarrow_{must} C$, if $\exists t \in T$: $c[t\rangle c' \wedge c' \in C$. Furthermore, a must-step exists from C to C', denoted by $C \rightarrow_{must} C'$, if $C' = \bigcup_{c \in C} C_c$ where $c \rightarrow_{must} C_c$ or $C_c = \{c\}$ if c is a dead configuration; and*

(4) *C'' is **must-reachable** from C if there is a sequence of configuration sets $C_0, \cdots, C_i, \cdots, C_n$ of N such that $C_i \rightarrow_{must} C_{i+1}$, where $0 \leq i < n$. It is denoted by $C \xrightarrow{*}_{must} C''$ with $C_0 = C$ and $C_n = C''$.*

The definition of reachability refers to [23]. May-reachability considers one execution path from a configuration to another one. Must-reachability considers all execution paths from a configuration to all successor configurations of

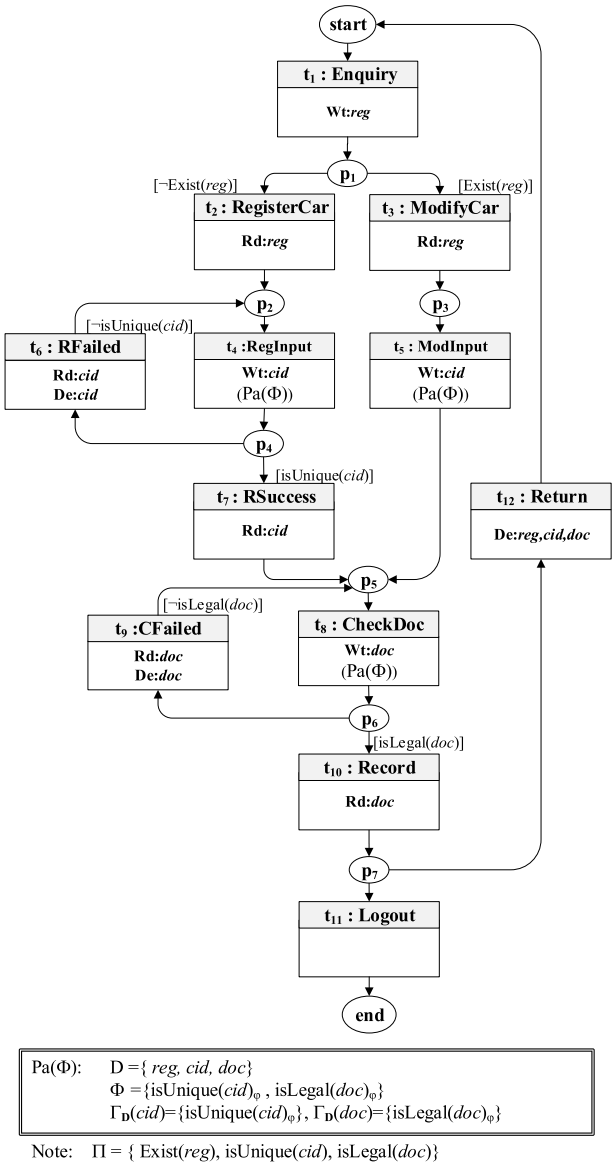


FIGURE 3. WFDC-net (workflow net with data constraints).

it. Due to the fact that each successor is may-reachable, must-reachability also generates different may-paths. In this paper, the set of may-reachable configurations from c_0 is denoted by $R(c_0)$.

For example, two configurations c_1 and c_4 are may-reachable from the initial configuration c_0 in Fig. 4, denoted as $c_0 \rightarrow_{may} c_1$ and $c_0 \stackrel{*}{\rightarrow}_{may} c_4$. Two configuration sets $\{c_1, c_2\}$ and $\{c_4, c_5, c_{12}, c_{13}\}$ are must-reachable from c_0, denoted as $c_0 \rightarrow_{must} \{c_1, c_2\}$ and $c_0 \stackrel{*}{\rightarrow}_{must} \{c_4, c_5, c_{12}, c_{13}\}$, respectively. Meanwhile, the configuration set $\{c_4, c_5, c_{12}, c_{13}\}$ is also must-reachable from $\{c_1, c_2\}$, which is denoted as $\{c_1, c_2\} \stackrel{*}{\rightarrow}_{must} \{c_4, c_5, c_{12}, c_{13}\}$.

Based on the firing rules of WFDC-net, we define a configuration graph with data constraints (CDC-graph).

Definition 10 (CDC-Graph): Let $N = (WD, Pa(\Phi))$ be a WFDC-net. $G = (\mathcal{C}, \mathcal{E}, \mathcal{S})$ is a configuration graph with data constraints of N, where

(1) $\mathcal{C} = R(c_0)$;

(2) $\mathcal{E} = \{(c, C') | \ c \in \mathcal{C} \wedge C' \in 2^{R(c_0)} \wedge (\exists t \in T : c[t\rangle C')\}$; *and*

(3) $\mathcal{S} : \mathcal{E} \rightarrow T$ *is a label function that* $S(c, C') = t$ *if* $c[t\rangle C' \wedge (c, C') \in \mathcal{E}$.

Fig. 4 shows a CDC-graph of the WFDC-net in Fig. 2, e.g., $e_0 = (c_0, \{c_1, c_2\}) \in \mathcal{E}$ and $\mathcal{S}(e_0) = t_1$. In this graph, there are three terminal configurations, which are represented by double-line circles. A branched arrow with multiple heads is used to represent an edge $e \in \mathcal{E}$ between a configuration and its successor configurations.

Algorithm 1 Constructing CDC-Graph

Require: WFDC-net N ;
Ensure: CDC-graph G;

1: Let c_0 be a root node, and mark it "*new*";
2: **while** there exists a "*new*" node **do**
3: Arbitrarily choose a "*new*" node, on behalf of c;
4: **if** $\forall t \in T : \neg c[t\rangle$ **then**
5: **if** $M = [end]$ **then**
6: Change the mark of c to "*end*"; **goto** *step* 2;
7: **else**
8: Change the mark of c to "*leaf*"; **goto** *step* 2;
9: **end if**
10: **else**
11: **for** each $t \in T$ that satisfies $c[t\rangle$ **do**
12: According to firing rules, calculate every $c' \in C'$ that satisfies $c[t\rangle\ C'$;
13: **if** there exists a node that has the same configuration as c' in the directed trace from c_0 to c **then**
14: Generate a directed arrow pointing to this node and label the arrow with "t";
15: **else**
16: **if** $|C'|$=1 **then**
17: Record c'; generate a directed arrow pointing to c' and label it with "t"; mark c' with "*new*";
18: **else**
19: Record every $c' \in C'$; generate a branched arrow pointing to each c' and label it with "t"; mark every c' with "*new*";
20: **end if**
21: **end if**
22: **end for**
23: **end if**
24: Change the mark of c to "*old*";
25: **end while**

Based on Definition 10, we propose an algorithm to construct the CDC-graph of a WFDC-net. As shown in Algorithm 1, we use the breadth-first method to implement this algorithm. According to the firing rule in Definition 8,

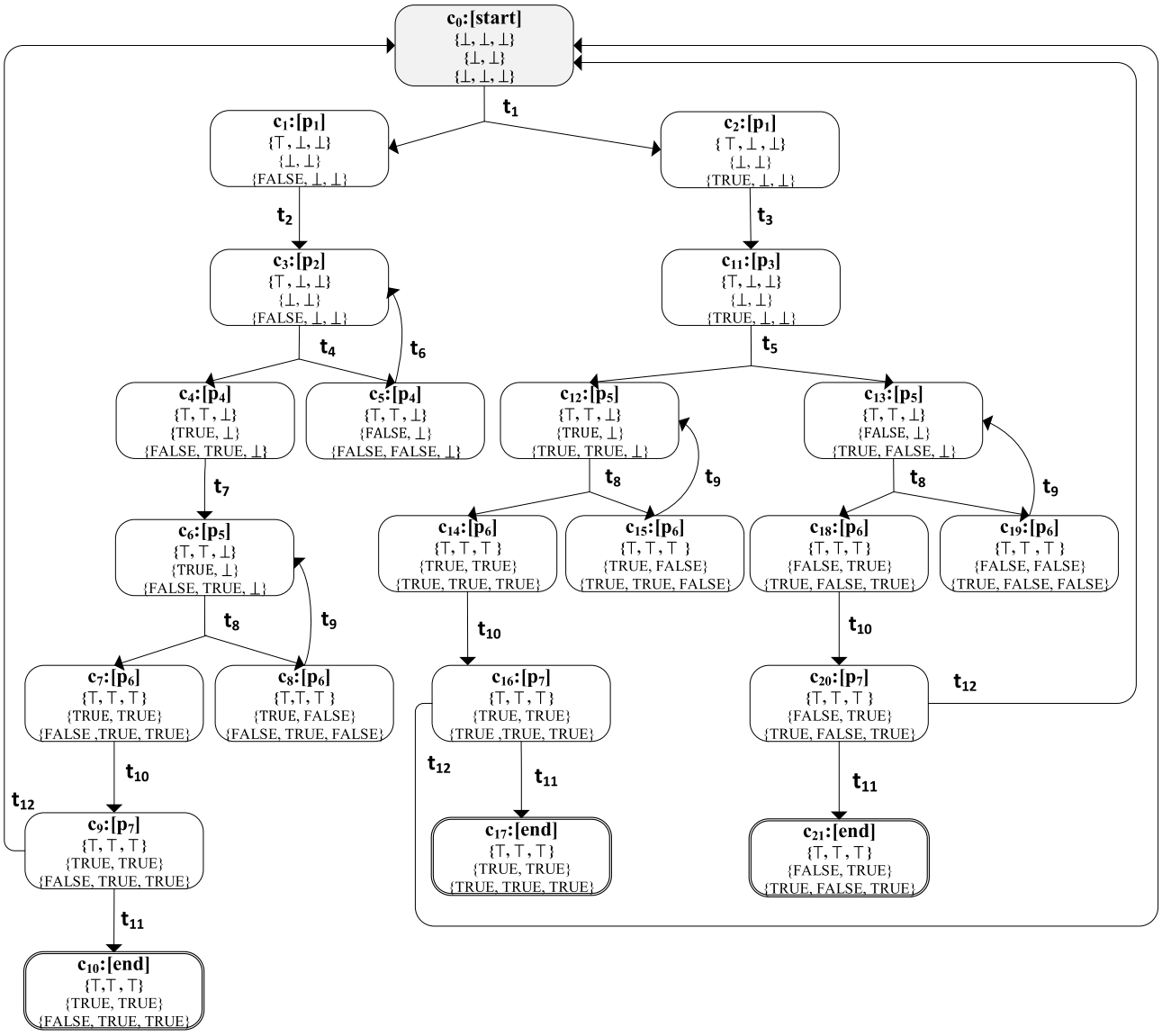


FIGURE 4. A CDC-graph of the WFDC-net in Fig.2.

new nodes (*i.e.*, configurations) and edges are iteratively generated and added into the CDC-graph.

The CDC-graph of a WFDC-net reflects all running information of a workflow system, *e.g.*, control-flows and data-flows with data constraints. Therefore, we can check the correctness of workflow systems based on their CDC-graph. In the following section, we extend the classical soundness with data constraints, and propose algorithms to verity it.

VI. HIERARCHICAL SOUNDNESS

The classical soundness property of WF-nets focuses on the correctness of business logics in the control-flows of workflow systems, and the analyzing result divides workflow systems into two categories: sound or non-sound (see in Fig. 5(a)). Based on WF-nets, WFD-nets consider the influence of data elements to the soundness verification and propose must-/may-soundness (see in Fig. 5(b)). These soundness properties only guarantee a proper termination of workflow systems. However, they ignore the satisfaction of some data constraints, *e.g.*, whether the car information is unique in a database. In fact, data constraints are closely related with the reliability and the correctness of systems. Therefore, we propose the hierarchical soundness of WFDC-nets which considers the following two aspects.

(1) Logical correctness. It guarantees that the terminal configurations can be reached in different data refinements. Meanwhile, there are no deadlocks, livelocks or dead transitions in all configurations.

(2) Satisfaction of data constraints. It guarantees that every terminal configuration and every path of configurations satisfy data constraints.

With respect to the above considerations, we propose four levels of soundness, *i.e.*, soundness, control-soundness,

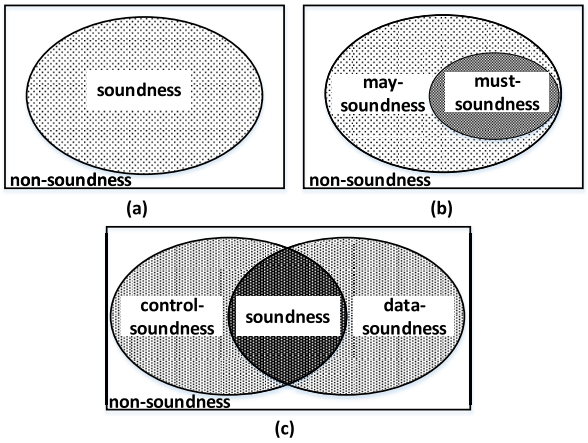


FIGURE 5. (a) The soundness of WF-net; (b) the soundness of WFD-net; (c) the soundness of WFDC-net.

data-soundness and non-soundness. Their relationships are shown in Fig. 5(c).

In order to formalize these soundness of WFDC-nets, we first discuss must-/may-termination and normal/abnormal termination of CDC-graphs with respect to the business logics and data-flows of workflow systems.

A. MUST-/MAY-TERMINATION

Based on the definition of reachability, we define must-termination and may-termination to ensure the control-flow correctness.

Definition 11 (Must-/May-Termination): Let $N = (WD, Pa(\Phi))$ *be a WFDC-net and* $WD = (P, T, F, D, Rd, Wt, De, grd, G_\Pi)$. $G = (\mathcal{C}, \mathcal{E}, \mathcal{S})$ *is a CDC-graph of* N, *where* c_0 *is the initial configuration of* N. c, c' *are configurations of* N. C, C_f *are sets of terminal configurations of* N. N *satisfies* ***must-termination*** *if* G *satisfies*

(1) $\forall c \in R(c_0) : \exists C \subseteq C_f \wedge c \xrightarrow{*}_{must} C$;

(2) $\forall c \in R(c_0) : M \geq [o] \Rightarrow M = [o]$; *and*

(3) $\forall t \in T : \exists\, c \in R(c_0)$: $c[t\rangle$.

N *satisfies* ***may-termination*** *if one of the above three conditions does not hold.*

The may-termination of WFDC-nets shows that there exist some paths that can reach terminal configurations but not every path is required to reach a terminal configuration. The must-termination represents that all paths from the initial configuration can reach the terminal configurations.

According to Definition 11, we can check whether every path has a terminal configuration, whether a path has an end configuration c that belongs to a terminal configuration set C_f and whether every transition is live. The specific method is given as Algorithm 2.

B. NORMAL/ABNORMAL TERMINATION

By using Algorithm 2, we can find that the CDC-graph in Fig. 4 satisfies must-termination. This shows that the business logics of the car management system can lead to deadlock-free and livelock-free. However, we cannot conclude that its workflow process is completely error-free at this time. We need to further verify whether its data constraints are satisfied via its CDC-graph, *i.e.*, a normal data that satisfies its constraints can reach the terminal configuration, but an abnormal data which does not satisfy its constraints is not allowed to. A correct WFDC-net must guarantee that the abnormal data can be stopped during execution. To this end, we propose the normal/abnormal termination of WFDC-net.

Definition 12 (Normal and Abnormal Termination): Let $N = (WD, Pa(\Phi))$ *be a WFDC-net and* $WD = (P, T, F, D, Rd, Wt, De, grd, G_\Pi)$. $G = (\mathcal{C}, \mathcal{E}, \mathcal{S})$ *is a CDC-graph of* N, *where* C_f *is a set of terminal configurations of* N. C, C' *are sets of configurations.* N *satisfies* ***normal termination*** *if for any* $d \in D$, G *satisfies:*

(1) $\forall c \in R(c_0), \forall \varphi \in \Gamma_D(d) : c \xrightarrow{*}_{must} C \wedge C \subseteq C_f \Rightarrow$

- $\forall c' \in C : \rho_\Phi{}'(\varphi) \neq FALSE$;
- $\exists c'' \in C : \rho_\Phi{}''(\varphi) = TRUE$; *and*

(2) $\forall c \in R(c_0), \forall \varphi \in \Gamma_D(d) : \rho_\Phi(\varphi) = FALSE \Rightarrow \exists C' \in 2^{R(c_0)} : c \xrightarrow{*}_{must} C' \wedge C' \nsubseteq C_f \wedge (\forall c' \in C', \rho_\Phi{}'(\varphi) = \perp)$.

Algorithm 2 Must-/May-Termination Analysis

Require: CDC-graph G; Transition set T;
Ensure: $Must(G)$; $sign_{dead}$; $sign_{lock}$; T_{fire};
/* $Must(G)$=TRUE *represents must-termination.* */
1: **Initialize** $sign = 0$; $sign_{dead} = 0$; $sign_{lock} = 0$; $T_{fire} = \emptyset$;
2: **for** $C = R(c_0); C \neq \emptyset; C = C - \{c\}$ **do**
3: Arbitrarily choose a $c \in C$;
4: Traverse all c' that satisfies $c \xrightarrow{*}_{may} c'$;
5: **if** $\exists\, c'$ marked with "*leaf*" **then**
6: **if** c' is marked with "*dead*" **then**
7: **goto** *step* 16;
8: **else**
9: mark c' with "*dead*"; $sign_{dead} = sign_{dead} + 1$;
/* There is a deadlock. */
10: **end if**
11: **else if** $\exists\, c'$ marked with "*end*" **then**
12: **goto** *step* 16;
13: **else**
14: $sign_{lock} = sign_{lock} + 1$;
/* There is no termination. */
15: **end if**
16: **if** $\exists t \in T$ satisfies $c[t\rangle$ **then**
17: For all t satisfies $c[t\rangle : T_{fire} = T_{fire} \cup \{t\}$;
18: **end if**
19: **end for**
20: $sign = sign_{dead} + sign_{lock}$;
21: **if** $sign = 0 \wedge T_{fire} = T$ **then**
22: $Must(G) = TRUE$;
23: **else**
24: $Must(G) = FALSE$;
25: **end if**

N satisfies ***abnormal termination*** *if G does not satisfy the above conditions.*

Algorithm 3 specifies the check method of Definition 12. To analyze the normal and abnormal termination, we need to verify every configuration of a CDC-graph from two aspects: data constraint values *TRUE* (satisfied) and *FALSE* (not satisfied). For a data $d \in D$ and its related constraint $\varphi \in \Gamma_D(d)$, if there exists a configuration c with $\rho_\Phi(\varphi) = TRUE$, then there must be a *TRUE* or $\perp$ of φ at the terminal configuration of every path from it, i.e., a data with a satisfied constraint will successfully reach the terminal configuration or be deleted for some reasons. If there exists a configuration c' with $\rho_\Phi(\varphi) = FALSE$, then there must be a $\perp$ of φ in every configuration of a must-reachable configuration set C from c', and there must be a *TRUE* or $\perp$ of φ in all terminal configurations from C, i.e., a data with a dissatisfied constraint will be deleted during execution and then either be replaced by a satisfied one or not. Furthermore, in all terminal configurations of paths from c and c', there exists at least one *TRUE* of φ, i.e., there must be at least one efficient operating result after execution.

For our motivated example, if users input a normal data into the car management system in Fig. 2, they successfully get the car permit-card, or they log out with nothing. If users input an abnormal data into the system, the data must be deleted during their executions and users must be guided to re-input a correct one or log out directly.

We can observe from Fig. 4 that the terminal configuration c_{21} does not satisfy Definition 12(1) since $\rho_\Phi(isUnique(cid)_\varphi) = FALSE$, which means the value of data *cid* does not satisfy its data constraint $isUnique(cid)_\varphi$, and it lacks a verification of *cid* after *cid* is written into this system.

C. HIERARCHICAL SOUNDNESS

Based on must-/may-termination and normal/abnormal termination above, we define four levels of soundness including soundness, control-soundness, data-soundness and non-soundness.

Definition 13 (Hierarchical Soundness): Let $N = (WD, Pa(\Phi))$ *be a WFDC-net,* $G = (\mathcal{C}, \mathcal{E}, \mathcal{S})$ *be a CDC-graph of* N. N *is*

(1) ***sound*** *if G satisfies must- and normal termination;*

(2) ***control-sound*** *if G satisfies must-termination;*

(3) ***data-sound*** *if G satisfies normal termination; or*

(4) ***non-sound*** *if G satisfies may- and abnormal termination.*

Soundness requires a completely proper termination, i.e., all execution paths can reach the terminal configurations under all data constraints from the initial configuration.

Control-soundness means that all execution paths can reach the terminal configurations, regardless of data constraints.

Data-soundness shows that all data elements satisfy data constraints and there are no abnormal data in the terminal configurations.

Algorithm 3 Normal-Termination Analysis

Require: CDC-graph G; Constraint pattern $Pa(\Phi)$;
Ensure: $Normal(G)$;Φ_T; Φ_F;

Initialization: array $\Phi_T[\,]$; array $\Phi_F[\,]$;
for Φ; $\Phi \neq \emptyset$; $\Phi = \Phi - \{\varphi\}$ **do**
  Arbitrarily choose a data constraint $\varphi \in \Phi$;
  for $C_T = R(c_0)$;$C_T \neq \emptyset$;$C_T = C_T - \{c\}$ **do**
    /* Verify data with a satisfied constraint. */
    Arbitrarily choose a $c \in C_T$;
    if $\rho_\Phi(\varphi) = TRUE$ **then**
      Traverse every $c_f \in C_f$ that satisfies $c \xrightarrow{*}_{must} C_f$;
      if $\exists \rho_{\Phi f}(\varphi) = FALSE$ **then**
        $\Phi_T[\varphi] = 1$; **goto** *step* 19;
      else
        if $\forall c_f \in C_f : \rho_{\Phi f}(\varphi) = \perp$ **then**
          $\Phi_T[\varphi] = 1$; **goto** *step* 19;
        else
          $\Phi_T[\varphi] = 0$;
        end if
      end if
    end if
  end for
  for $C_F = R(c_0)$;$C_F \neq \emptyset$;$C_F = C_F - \{c\}$ **do**
    /* Verify data with a dissatisfied constraint. */
    Arbitrarily choose a data a $c \in C_F$;
    if $\rho_\Phi(\varphi) = FALSE$ **then**
      Traverse every C that satisfies $c \xrightarrow{*}_{must} C$;
      if $\exists C$ that satisfies $\forall c' \in C : \rho_\Phi{}'(\varphi) = \perp$ **then**
        if $C \nsubseteq C_f$ **then**
          Find C' satisfies $C \xrightarrow{*}_{must} C' \wedge C' \subseteq C_f$;
          if $\nexists c'' \in C' : \rho_\Phi{}''(\varphi) = FALSE$ **then**
            if $\exists c''' \in C' : \rho_\Phi{}'''(\varphi) = TRUE$ **then**
              $\Phi_F[\varphi] = 0$;
            else
              $\Phi_F[\varphi] = 1$; **goto** *step* 2;
            end if
          else
            $\Phi_F[\varphi] = 1$; **goto** *step* 2;
          end if
        else
          $\Phi_F[\varphi] = 1$; **goto** *step* 2;
        end if
      else
        $\Phi_F[\varphi] = 1$; **goto** *step* 2;
      end if
    end if
  end for
end for
if $(\bigvee_{\varphi\in\Phi} \Phi_T[\varphi] = 0) \wedge (\bigvee_{\varphi\in\Phi} \Phi_F[\varphi] = 0)$ **then**
  $Normal(G) = TRUE$;
else
  $Normal(G) = FALSE$;
end if

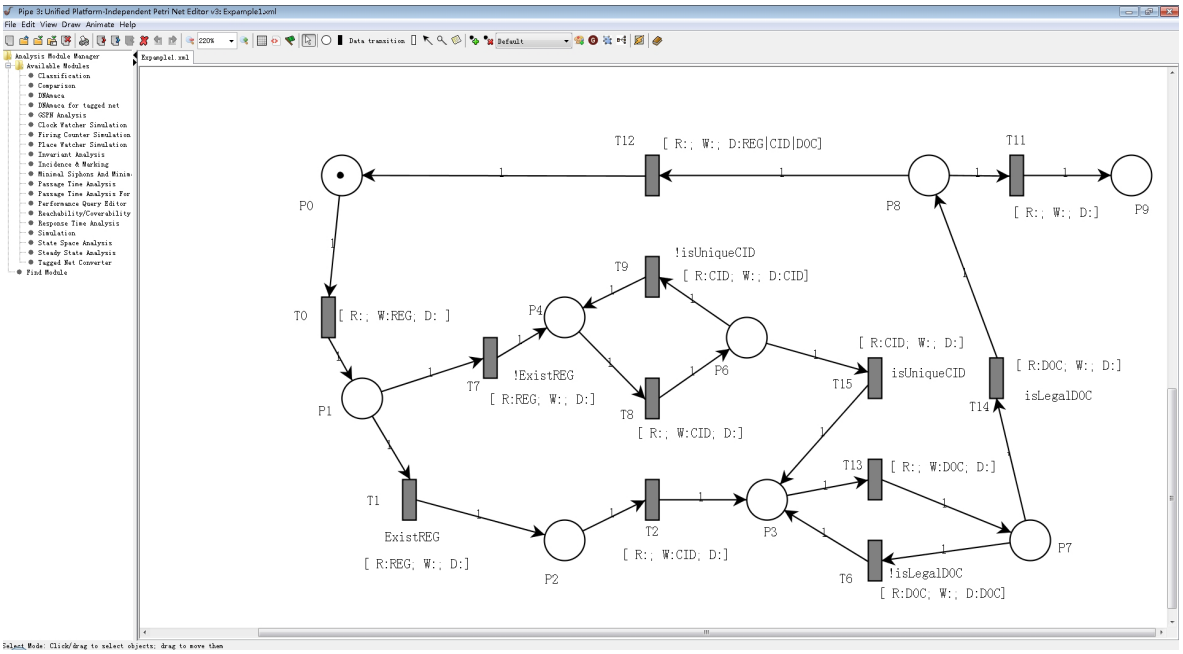


FIGURE 6. A WFDC-net in LDCC-tool.

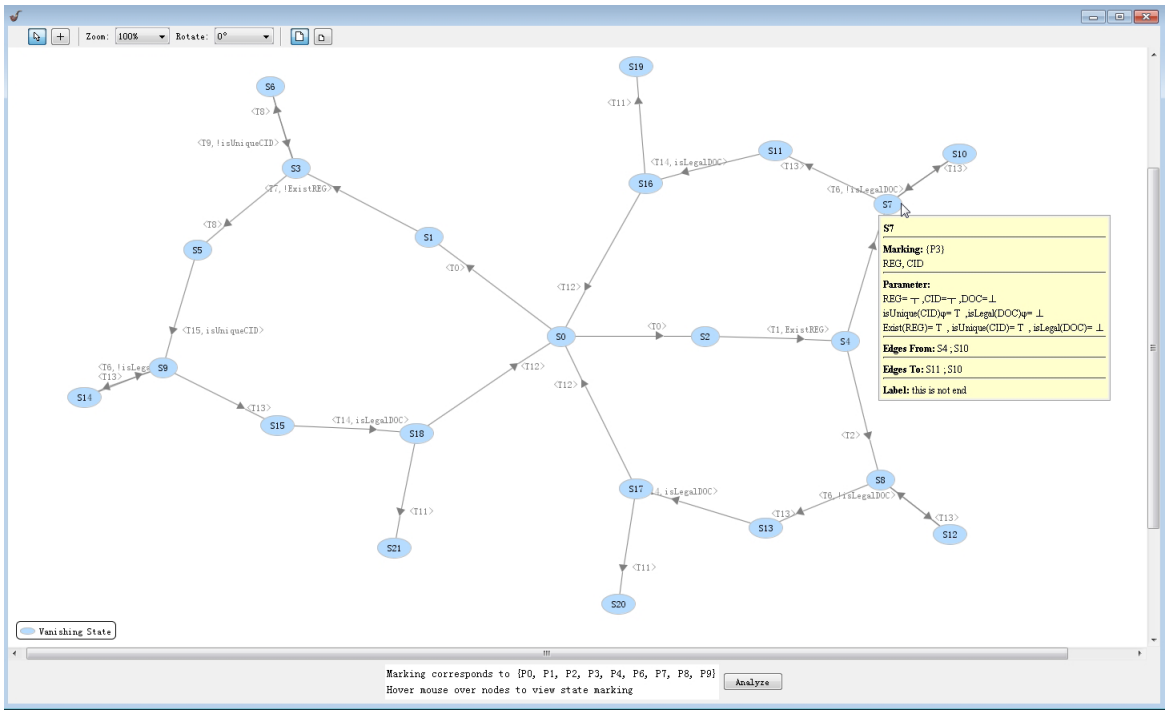


FIGURE 7. A CDC-graph in LDCC-tool.

Non-soundness indicates the incorrectness in both control-flows and data-flows of a workflow system.

VII. AN ANALYZING TOOL OF WFDC-NET

Based on our algorithms, we develop a tool called LDCC-tool (Logic and Data Constraints Check tool) to detect the errors in control-flows and data-flows with data constrains.

LDCC-tool is developed based on PIPE (Platform Independent Petri Net Editor) [33], which is an open source tool of Petri net. Using LDCC-tool, we can create not only WFDC-nets but also WFD-nets, and produce CDC-graphs. Furthermore, LDCC-tool can automatically detect the errors in control-flows and data incorrectness, and give an analysis result of hierarchical soundness. By using LDCC-tool, we can

import and export a WFDC-net, as shown in Fig. 6. Besides, we can edit and modify its data elements, guards and data constraints. The CDC-graph of the WFDC-net is constructed in Fig. 7, where every node represents a configuration and its detailed information is hidden in it.

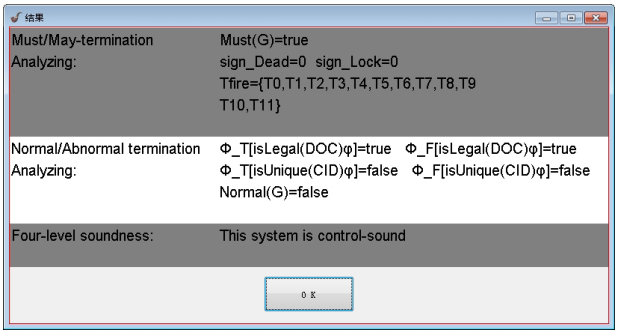


FIGURE 8. The analysis result of hierarchical soundness in LDCC-tool.

The analysis result of the car management system is shown in Fig. 8. $Must(G) = true$ means that it has no deadlock, livelock or dead transition, i.e., it is control-sound. From the result, $\Phi_T[isUnique(cid)_\varphi] = false$ indicates that there is at least one path such that cid with a satisfied constraint cannot reach the terminal configuration, $\Phi_F[isUnique(cid)_\varphi] = false$ indicates that there is at least one path such that cid with a dissatisfied constraint can reach the terminal configuration, and $Normal(G) = false$ means this system does not satisfy all data constraints, i.e., it is not data-sound and lacks verification of data cid. Obviously, this workflow system is sound when we take the same control mechanism to the modifying procedure as the registering procedure.

VIII. CONCLUSION

The correctness is very important for the workflow systems. It depends on both control-flows and data-flows. On one hand, a correct workflow system requires a proper termination, *e.g.*, no deadlocks, livelocks or dead transitions. On the other hand, the system should satisfy some data constraints. Considering the above two points, we propose Workflow Net with Data Constraints (WFDC-net) to model workflow systems in which data constraints are represented by propositional formulas. We also define different levels of soundness to reflect different correctness requirements. Based on the reachability graph of WFDC-net, we develop the related tool to automatically verify them. The implementation of LDCC-tool proves that our method can effectively check the correctness of workflow systems.

In the future, we plan to do the following work:

(1) Take some control mechanisms to guarantee the soundness.

(2) Use the unfolding techniques[34]–[36] to reduce the state space explosion problem.

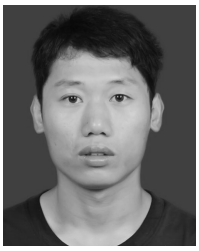
REFERENCES

[1] M. Dumas, W. M. P. van der Aalst, and A. H. ter Hofstede, *Process Aware Information Systems: Bridging People and Software Through Process Technology*. Hoboken, NJ, USA: Wiley, 2005.
[2] K. M. van Hee, N. Sidorova, and J. M. van der Werf, "Business process modeling using Petri nets," in *Transactions on Petri Nets and Other Models of Concurrency VII*. Berlin, Germany: Springer, 2013, pp. 116–161.
[3] E. M. Clarke, O. Grumberg, and D. E. Long, "Model checking and abstraction," *ACM Trans. Program. Lang. Syst.*, vol. 16, no. 5, pp. 1512–1542, 1994.
[4] M. J. Ibáñez, J. Fabra, P. Álvarez, and J. Ezpeleta, "Model checking analysis of semantically annotated business processes," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 42, no. 4, pp. 854–867, Jul. 2012.
[5] A. Speck, S. Feja, S. Witt, and E. Pulvermüller, and M. Schulz, "Formalizing business process specifications," *Comput. Sci. Inf. Syst.*, vol. 18, no. 2, pp. 427–446, 2011.
[6] D. Xu and K. E. Nygard, "Threat-driven modeling and verification of secure software using aspect-oriented Petri nets," *IEEE Trans. Softw. Eng.*, vol. 32, no. 4, pp. 265–278, Apr. 2006.
[7] W. M. P. van der Aalst, "The application of Petri nets to workflow management," *J. Circuits, Syst., Comput.*, vol. 8, no. 1, pp. 21–66, 1998.
[8] W. M. P. van der Aalst *et al.*, "Soundness of workflow nets: Classification, decidability, and analysis," *Formal Aspects Comput.*, vol. 23, no. 3, pp. 333–363, 2011.
[9] W. M. P. van der Aalst, "Verification of workflow nets," in *Proc. Int. Conf. Appl. Theory Petri Nets*, 1997, pp. 407–426.
[10] A. Martens, "Analyzing web service based business processes," in *Proc. Int. Conf. Fundam. Approaches Softw. Eng.*, 2005, pp. 19–33.
[11] J. Dehnert and P. Rittgen, "Relaxed soundness of business processes," in *Advanced Information Systems Engineering*. Berlin, Germany: Springer, 2001, pp. 157–170.
[12] K. Van Hee, N. Sidorova, and M. Voorhoeve, "Soundness and separability of workflow nets in the stepwise refinement approach," in *Proc. Int. Conf. Appl. Theory Petri Nets*, 2003, pp. 337–356.
[13] K. Van Hee, N. Sidorova, and M. Voorhoeve, "Generalised soundness of workflow nets is decidable," in *Applications and Theory of Petri Nets*. Berlin, Germany: Springer, 2004, pp. 197–215.
[14] S. Sadiq, M. Orlowska, W. Sadiq, and C. Foulger, "Data flow and validation in workflow modelling," in *Proc. 15th Austral. Database Conf.*, vol. 27. 2004, pp. 207–214.
[15] S. X. Sun, J. L. Zhao, J. F. Nunamaker, and O. R. L. Sheng, "Formulating the data-flow perspective for business process management," *Inf. Syst. Res.*, vol. 17, no. 4, pp. 374–391, 2006.
[16] A. Awad, G. Decker, and N. Lohmann, "Diagnosing and repairing data anomalies in process models," in *Proc. Int. Conf. Bus. Process Manage.*, 2009, pp. 5–16.
[17] D. Xiang, G. Liu, C. Yan, and C. Jiang, "Checking the inconsistent data in concurrent systems by Petri nets with data operations," in *Proc. IEEE 22nd Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2016, pp. 501–508.
[18] Y. Du, L. Qi, and M. Zhou, "Analysis and application of logical Petri nets to E-commerce systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 4, pp. 468–481, Apr. 2014.
[19] D. Cohn and R. Hull, "Business artifacts: A data-centric approach to modeling business operations and processes," *IEEE Data Eng. Bull.*, vol. 32, no. 3, pp. 3–9, Sep. 2009.
[20] E. Damaggio, A. Deutsch, and V. Vianu, "Artifact systems with data dependencies and arithmetic," *ACM Trans. Database Syst.*, vol. 37, no. 3, 2012, Art. no. 22.
[21] N. Trčka, W. M. P. van der Aalst, and N. Sidorova, "Analyzing control-flow and data-flow in workflow processes in a unified way," *Comput. Sci. Rep.*, vol. 0831, pp. 1–24, Jan. 2008.
[22] N. Trčka, W. M. P. van der Aalst, and N. Sidorova, "Data-flow anti-patterns: Discovering data-flow errors in workflows," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.*, 2009, pp. 425–439.
[23] N. Sidorova, C. Stahl, and N. Trčka, "Workflow soundness revisited: Checking correctness in the presence of data while staying conceptual," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.*, 2010, pp. 530–544.
[24] N. Sidorova, C. Stahl, and N. Trčka, "Soundness verification for conceptual workflow nets with data: Early detection of errors with the most precision possible," *Inf. Syst.*, vol. 36, no. 7, pp. 1026–1043, 2011.
[25] Z. Wang, J. Wang, X. Zhu, and L. Wen, "Verification of workflow nets with transition conditions," *J. Zhejiang Univ.-Sci. C*, vol. 13, no. 7, pp. 483–509, 2012.

[26] R. Wang, S. Chen, X. Wang, and S. Qadeer, "How to shop for free online security analysis of cashier-as-a-service based Web stores," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2011, pp. 465–480.
[27] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.
[28] O.-J. Dahl, *Structured programming*. San Diego, CA, USA: Academic, 1972.
[29] G. Smith and J. Derrick, "Verifying data refinements using a model checker," *Formal Aspects Comput.*, vol. 18, no. 3, pp. 264–287, 2006.
[30] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[31] (Feb. 21, 2012). *One Yuan Gate Event of Taobao*. [Online]. Available: https://baike.baidu.com/view/6419081.htm
[32] (Nov. 11, 2017). *Vulnerability of the Cambridge Satchel Company Website*. [Online]. Available: http://baijiahao.baidu.com/s?id=1583727274643974605&wfr=spider&for=pc
[33] N. J. Dingle, W. J. Knottenbelt, and T. Suto, "PIPE2: A tool for the performance evaluation of generalised stochastic Petri nets," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 4, pp. 34–39, 2009.
[34] K. L. McMillan, "Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits," in *Proc. Int. Conf. Comput. Aided Verification*, 1992, pp. 164–177.
[35] J. Esparza, S. Römer, and W. Vogler, "An improvement of McMillan's unfolding algorithm," *Formal Methods Syst. Des.*, vol. 20, no. 3, pp. 285–310, 2002.
[36] G. Liu, W. Reisig, C. Jiang, and M. Zhou, "A branching-process-based method to check soundness of workflow systems," *IEEE Access*, vol. 4, pp. 4104–4118, 2016.

YAQIONG HE received the degree from Tongji University, China, in 2012, where she is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tongji University. Her research interests include model checking, Petri net, business process management, and workflow systems.

GUANJUN LIU (M'16) received the Ph.D. degree in computer software and theory from Tongji University, Shanghai, China, in 2011. He was a Post-Doctoral Research Fellow with the Singapore University of Technology and Design, Singapore, from 2011 to 2013 and with the Humboldt University zu Berlin, Germany, from 2013 to 2014, supported by the Alexander von Humboldt Foundation. He is currently an Associate Professor with the Department of Computer Science and Technology, Tongji University. He has authored over 60 papers including 12 ones in IEEE/ACM TRANSACTIONS and one book *Liveness of Petri Nets and Its Application* (Tongji University Press, 2017). His research interests include Petri net theory, model checking, Web service, workflow, discrete event systems, and information security.



DONGMING XIANG received the bachelor's and M.S. degrees from the University of Jinan, China, in 2010 and2013, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tongji University. His research interests include model checking, Petri net, business process management, and service computing.

JIAQUAN SUN received the bachelor's degree from the University of Zhejiang Technology, China, in 2016. He is currently pursuing the M.S. degree with the Department of Computer Science and Technology, Tongji University. His research interests include model checking, Petri net, and business process management.

CHUNGANG YAN received the Ph.D. degree from Tongji University, Shanghai, China, in 2006. She is currently a Professor with the Department of Computer Science and Technology, Tongji University. She has authored or co-authored over 100 papers in domestic and international academic journals and conference proceedings. Her current research interests include concurrent model and algorithm, Petri net theory, formal verification of software, trusty theory on software process.

CHANGJUN JIANG received the Ph.D. degree from the Institute of Automation, Chinese Academy of Science, Beijing, China, in 1995. He is currently the Leader with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai, China. He is an IET Fellow and an Honorary Professor with Brunel University London. He has authored or co-authored over 300 papers in journals and conference proceedings. He has led over 30 projects. His research interests include concurrency theory, Petri nets, formal verification of software, cluster, grid technology, intelligent transportation systems, and service-oriented computing. He was a recipient of one international prize and seven prizes in the field of science and technology.