Journal of Systems Engineering and Electronics
Vol. 35, No. 5, October 2024, pp.1219 – 1230

A general Boolean semantic modelling approach for complex and intelligent industrial systems in the framework of DES

XU Changyi[1], WANG Yun[1], DUAN Yiman[2], and ZHANG Chao[2,*]

1. Key Laboratory of Intelligent Control and Optimization for Industrial Equipment of Ministry of Education, School of Control Science and Engineering, Dalian University of Technology, Dalian 116024, China; 2. State Key Laboratory of Fluid Power and Mechatronic Systems, School of Mechanical Engineering, Zhejiang University, Hangzhou 310027, China

Abstract: Discrete event system (DES) models promote system engineering, including system design, verification, and assessment. The advancement in manufacturing technology has endowed us to fabricate complex industrial systems. Consequently, the adoption of advanced modeling methodologies adept at handling complexity and scalability is imperative. Moreover, industrial systems are no longer quiescent, thus the intelligent operations of the systems should be dynamically specified in the model. In this paper, the composition of the subsystem behaviors is studied to generate the complexity and scalability of the global system model, and a Boolean semantic specifying algorithm is proposed for generating dynamic intelligent operations in the model. In traditional modeling approaches, the change or addition of specifications always necessitates the complete resubmission of the system model, a resource-consuming and error-prone process. Compared with traditional approaches, our approach has three remarkable advantages: (i) an established Boolean semantic can be fitful for all kinds of systems; (ii) there is no need to resubmit the system model whenever there is a change or addition of the operations; (iii) multiple specifying tasks can be easily achieved by continuously adding a new semantic. Thus, this general modeling approach has wide potential for future complex and intelligent industrial systems.

Keywords: industrial complex system, operation specifying, Boolean semantic, discrete event system (DES) theory, intelligent operation.

DOI: 10.23919/JSEE.2024.000066

1. Introduction

The model-based systems engineering illustrates the physical behaviors, intelligent operations, component cooperating relationships, and architectures of complex and intelligent industrial systems by formal methodology [1], which is always used in system design, behavior simulation and control, dependability assessment, and property verification [2–5]. Discrete event system (DES) theory [6,7] is commonly applied in model-based system engineering, and it has two advantages: the states intuitively illustrate the structure of the system, and the transitions intuitively illustrate the behaviors [8]. Modeling methods for DES, such as stochastic Petri nets [9], Markov chains [10] and automaton theory [11] are commonly used. The Markov stochastic processes are memoryless [12]: the next state only depends on the current state regardless of any past evolution [13,14]. The continuous time Markov chains (CTMC) express state transitions by relying on event occurrence rates [15,16]. In the process of modeling, the global system behavior needs to be built out of its individual components. Thus, the composition of the individual models in order to generate the system model is a required step. Moreover, additional behavioral specifications need to be modeled and composed into the global result. Petri nets can be composed by using the interconnection operation [17]. Automaton theory [18] delves into abstract machines and computational processes. It serves as a cornerstone in computer science [19], wielding significant influence across diverse domains such as formal languages, compiler design, artificial intelligence, and algorithm analysis. Nevertheless, there are three problems that restrict the wide applications and development of automaton theory in system engineering.

(i) With the advancement of modern manufacturing techniques, industrial systems have become increasingly complex [20]. Consequently, the system mock-up models have also grown in complexity, detail, and scale, often comprising thousands of states and transitions [21,22]. Thus, manual modeling work of engineers will be burdensome, the accuracy of the model will be lost, and the huge model will be less explainable for describing the system with numerous states and transitions.

Manuscript received May 31, 2023.
*Corresponding author.
This work was supported by the National Natural Science Foundation of China (U21B2074;52105070).

(ii) As human beings enter the era of intelligence, the application of intelligent controls and operations to industrial systems must be dynamic, given that these systems are not static entities. Therefore, the system model should be evolvable by dynamically specifying the operations [23,24]. In the traditional approaches, it is necessary to design a specification automaton that contains the information about intelligent operation, and then the system model will be successfully specified by composing this specification automaton with the system model [6,25–27]. As industrial systems and operations become increasingly large and complex [28], the implementation of specification automaton will be a huge challenge even for experienced system engineers. It is very difficult or even impossible for an engineer to artificially consider every detailed transiting trajectory among the thousands of states in the system model. Moreover, designing state transition functions to fully present the intelligent operations in the specification model is also difficult for engineers. Additionally, a change or an addition of intelligent operations will totally lead to a resubmission of the previous models. Consequently, there arises a need to reconsider the system model and redesign the specification model. This means that the traditional specifying approaches lack generality and compatibility, especially when facing multiple specifying tasks for one system or the same operation in different systems.

(iii) In traditional automaton theory [29], the composition of the system model and operating specification models results in a Cartesian product [30] of the states and transitions. Introducing an extra state in either the specification or system model will lead to a double growth of the state set in the target model [31]. As the system and specification models become huge, there is a risk of state explosion, and the target model will lose its function.

To solve above problems, in this work, Boolean semantically enhanced automaton is studied. In the component model, the Boolean symbol is used to associate with the states, allowing for a thorough illustration of the detailed behaviors of the components. To handle scalability, the composition of these new component models is developed to automatically generate the global system model. The Boolean labels are conjunct via this composition, so all states could be expressed by the Boolean information in the system model. To address the specifying problem of intelligent operations, easily constructed Boolean semantics are created to replace the skill-calling specification automaton. Boolean logic is used to identify the state labels and the event sequence is used to illustrate the operation information, thereby driving the model evolution. The proposed Boolean semantics specification algorithm can dynamically generate intelligent operations in the model. Due to the robust expressive power of Boolean semantics, the main contributions of this paper include the following three aspects: (i) Scalability. By continuously adding new Boolean semantics, various specification tasks can be easily accomplished. (ii) Agility. This approach exhibits strong compatibility, requiring no alteration to the design of the Boolean semantics specification automaton whenever the model changes. (iii) Universality. An established Boolean semantic can be fitful for all kinds of systems.

This paper is arranged as follows: Section 1 is the introduction, which states the problem and motivation; Section 2 is the introduction of the traditional specifying solution of the automaton in DES theory, representing the state of the art in this field and it is also compared with our proposed approach; Section 3 is the algorithms of the Boolean semantic specifying method; Section 4 is the case of study; Section 5 is the conclusion and perspectives.

2. Traditional approach

In this section, the traditional approach of the system model generation and the operation specifying process is introduced. For example, an industrial system is constructed by two transceivers, and each transceiver has “idle”, “send”, and “receive” functions. An easy intelligent operation is set: “within these two transceivers, the one that first receives should first send”. There are three steps:

Step 1 The modeling of the transceivers and the generation of the system model by the composition of the component models;

Step 2 The design of the specification automaton and the composition operation of the specification automaton and the system model;

Step 3 The discussion of this approach.

2.1 Component modeling

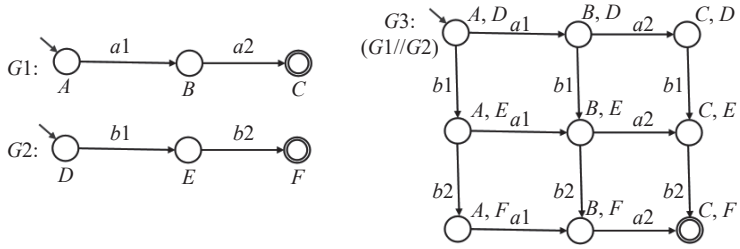
The behaviors of the components (transceivers in this system) are modeled by the form of automaton [6]. A deterministic finite automaton [32] is of five tuples denoted by

$$G = (Q;\ E;\ f;\ x_0;\ X_m) \tag{1}$$

where Q is the state set, E is the set of transitions, and f is the transition function $Q\times E \rightarrow Q$. For example, e is assumed to be a transition event and $(q_1, q_2)\in Q$, $e\in E$, so $f(q_1, e)=q_2$ means that state q_1 transits to state q_2 by transition e. x_0 is the initial state, and X_m is the set of marked states.

The modeling of the transceivers is depicted in Fig. 1(a).

The models are $G1=(Q1;\ E1;\ f1;\ x_{01};\ X_{m1})=(\{A,\ B,\ C\};\ \{a1,\ a2\};\ f1;\ A;\ C)$ and $G2=(Q2;\ E2;\ f2;\ x_{02};\ X_{m2})=(\{D,\ E,\ F\};\ \{b1,\ b2\};\ f2;\ D;\ F)$. States A and D are the idle states; B and E are the sending behavior states; C and F are the receiving behavior states; transitions $a1$ and $b1$ are system commands to start the sending function; $a2$ and $b2$ are system commands to start the receiving function.



(a) Component models (b) System model

Fig. 1 Modeling of the transeivers and the system model

To generate the system model, these two automata are composed, as shown in Fig. 1(b). We use "//" to denote the composition operation between two automata and "×" to denote the Cartesian product operation of the subset, so the composition can be defined:

$$G1//G2=A_c(Q1\times Q2;E1\cup E2;f;\ x_{01}\times x_{02};X_{m1}\times X_{m2}) \tag{2}$$

where A_c is the extraction function of the accessible parts in the brackets.

The transition function f is

$$f((x,y),e)=\begin{cases}(f_1(x,e),f_2(y,e)), & e\in E_1\cap E_2\\ (f_1(x,e),y), & e\in E_1\backslash(E_1\cap E_2)\\ (x,f_2(y,e)), & e\in E_2\backslash(E_1\cap E_2)\\ \text{undefined}, & \text{otherwise}\end{cases} \tag{3}$$

where the symbol "\" denotes the complement set operation; x and y are one of the states of $Q1$ and $Q2$ respectively. The explanation of (3) has three layers, in the case where there is an intersection in $E1$ and $E2$: (i) If the transition event e belongs to the intersection, the transition function f can be represented using transition from $f1$ and $f2$; (ii) If e belongs to the complement set of the intersection with $E1$ as the complete set, the transition function f can be represented using transition from $f1$; (iii) If e belongs to the complement set of the intersection with $E2$ as the complete set, the transition function f can be represented using transition from $f2$. Therefore, the system model (denoted by $G3$) can be obtained:

$$\begin{aligned}&G3=G1//G2=\\ &A_c\ (Q1\times Q2;E1\cup E2;\\ &f3;\ x_{01}\times x_{02};X_{m1}\times X_{m2})\end{aligned} \tag{4}$$

where $Q1\cdot Q2=\{(A,\ D);\ (B,\ D);\ (C,\ D);\ (A,\ E);\ (B,\ E);\ (C,\ E);\ (A,\ F);\ (B,\ F);\ (C,\ F)\}$; $E1\cup E2=\{a1,\ a2,\ b1,\ b2\}$; $x_{01}\cdot x_{02}=(A,\ D)$; $X_{m1}\cdot X_{m2}=(C,\ F)$.

According to (3), the composition result function $f3$ is generated from local transition function $f1$ and $f2$. Based on the Cartesian product, each global state in $G3$ is generated from one local state in $G1$ and one local state in $G2$, while the state transition evolution is driven by local transitions from $G1$ and $G2$. For example, in $G3$, the state evolution $f3((B,\ D),\ b1)=(B,\ E)$ is driven by $b1$ ($b1\in E2$, and $f2(D,\ b1)=E$). This global automaton $G3$ can describe the whole industrial system behavior of subsystem $G1$ and subsystem $G2$.

2.2 Specification automaton to specify intelligent operations

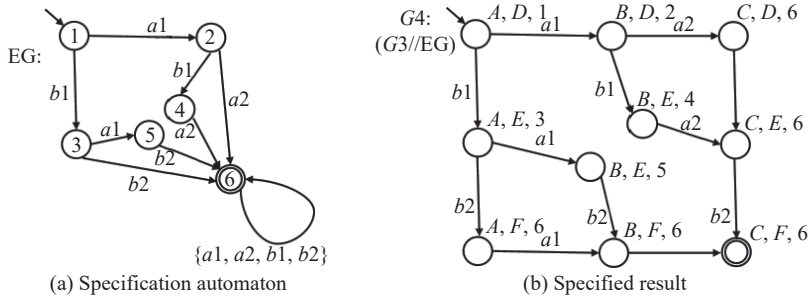
Based on DES theory, a specification automaton that contains intelligent operation information is designed. By further composing the specification automaton and the system model, intelligent operations can be successfully specified into the model. The specification automaton is defined as $EG=(Q;\ E;\ f;\ x_0;\ X_m)$ [6].

In model $G3$ (Fig. 1(b)), four trims pass through state $(B,\ E)$, namely $\{a1a2;\ b1b2;\ a1b2;\ b1a2\}$. Based on the intelligent operation principle "first send, first receive", trims $a1b2$ and $b1a2$ are legal trims, while trims $b1b2$ and $a1a2$ are illegal. Thus, $G3$ should be operated to be "state splitting" at state $(B,\ E)$, signifying a state similar to state $(B,\ E)$ should be added and the event sequence should be replanned: one state is passing through trim $b1a2$ while the other state is passing through $a1b2$ so that illegal trims can be avoided.

In Fig. 2(a), the specification automaton EG is designed:

$$EG=(Q_{EG};\ E_{EG};\ f_{EG};\ x_{0EG};\ X_{mEG}) \tag{5}$$

where $Q_{EG}=\{1,\ 2,\ 3,\ 4,\ 5,\ 6\}$; $E_{EG}=\{a1,\ a2,\ b1,\ b2\}$; $x_{0EG}=1$; $X_{mEG}=6$.



(a) Specification automaton (b) Specified result

Fig. 2 Specification automation to specify the principle of "first send, first receive" for intelligent operations

The initial state 1 passes through $a1b1$ to state 4, and then state 4 has only an output transition $a2$. Thus, we can ensure that if component $G1$ first sends ($a1$) and following component $G2$ sends ($b1$), our system can only execute the operation that component $G1$ receives ($a2$), thereby satisfying the operation principle "first send, first receive". It is the same principle for state 5; if component $G2$ first sends, the specification can guarantee that $G2$ is the first to receive. To achieve the above target function, the EG should be composed of the system automaton $G3$, and then the specified automaton becomes $G4=G3//\text{EG}$, where the composing operation is the same as the composition operation in Subsection 2.1. The reason is that in EG, $f_{\text{EG}}(2, b1)=4$ and $f_{\text{EG}}(3, a1)=5$; in $G3$, $f_3((B, D), b1)=(B, E)$ and $f_3((A, E), a1)=(B, E)$.

According to the transition function of (3), $f_{G4}(((B, D), 2), b1)=(f_{G3}((B, D), b1), f_{\text{EG}}(2, b1))=(B, E, 4)$, $f_{G4}(((A, E), 3), a1)=(f_{G3}((A, E), a1), f_{\text{EG}}(3, a1))=(B, E, 5)$.

As $G3//\text{EG}$ is depicted in Fig. 2(b), state (B, E) is a Cartesian product with state 4 and state 5. State (B, E) is divided into two new states $(B, E, 4)$ and $(B, E, 5)$, ensuring that their transition trims adhere to the intelligent operation principle of "first send, first receive".

2.3 Discussion

The composition of component models can generate the system model, providing an alternative solution for automated system model generation. The establishment of the specification is difficult to handle, demanding meticulous consideration of the intricate transitioning data in the system model and accurate planning of new event sequences based on intelligent operations. Given the potential complexity of both the system model and intelligent operations, which may encompass thousands of states, this specifying task seems to be impossible for a proficient engineer. It requires dynamically specifying the model whenever the intelligent operations are set during the lifecycle of the industrial system. Moreover, this specifying approach seriously lacks compatibility. Any alteration or resubmission of the system or the operation calls for a total change of previous work, rendering the specification non-reusable. Hence, there is a pressing need for an easily manageable and universally applicable specification approach, which is expected to foster the adoption of DES.

3. Boolean semantic specifying approach

This section introduces the Boolean semantic specifying method, which consists of five steps. To fully express the component behaviors, Boolean labels are applied to the automaton model in Subsection 3.1. The component models are composed to generate the system model; thus, the Boolean label conjunction through composition is studied in Subsection 3.2. As each state is linked with a Boolean label in the system model, the states can be identified by the property introduced in Subsection 3.3. In Subsection 3.4, according to the transition function, the transitions in the system model can be identified with the help of identified states. In Subsection 3.5, the Boolean semantic specifying solution that can be processed by the identified transitions and states is studied.

3.1 Boolean label-enhanced automaton for component modeling

In Subsection 2.1, the modeling of component behaviors is introduced, and here, this modeling solution is enhanced by the addition of Boolean labels. The Boolean labels provides a comprehensive representation of component behaviors, serving as a valuable tool for state and event sequence identification in the specification of intelligent operations.

A set of Boolean labels (denoted by AP, for the meaning of atomic proposition) and a link function that links the state names with Boolean labels (denoted by L, for the meaning of linking) are added to the definition of automaton.

$$G=(Q;\ E;\ f;\ x_0;\ X_m;\ \text{AP};\ L) \tag{6}$$

where Q, E, f, x_0, X_m are the same as defined in Subsection 2.1; AP is the set of Boolean labels, for example, we

can use a Boolean letter WORK to represent that the system is working, and its negation ¬WORK to indicate that the system is not working, AP=$\{x_0, x_1, x_2, \cdots, x_n\}$; L: $Q \to 2^{AP}$ is the link function linking the state names with Boolean labels, and 2^{AP} is the power set of AP. The conjunction of 2^{AP} elements is denoted by the form of logic "and" (symbol: $\wedge$), $2^{AP}=\{\varnothing, x_0, x_1, x_2, x_0 \wedge x_1, x_0 \wedge x_2, x_1 \wedge x_2, x_0 \wedge x_1 \wedge x_2, \cdots\}$.

However, although the state names are different, two or more states can share the same Boolean label, provided that these states have the same characteristics. For example, in Fig. 1(a), although state 2 and state 3 are in the "send" and "receive" states, respectively, they can share the same label, "WORK", and "idle" state A should be linked with the label "¬WORK", when it is only considered that the system is working or stopped.

For the send-receive industrial system (in Fig. 1), the automata can be defined: $G1 = (Q1; E1; f1; x_{01}; X_{m1}; \text{AP1}; L1)$ and $G2 = (Q2; E2; f2; x_{02}; X_{m2}; \text{AP2}; L2)$. As described, each local component has three working modes: "idle", "send", and "receive". To establish the AP set, for each local component, three basic Boolean elements are needed: AP1={idle1, send1, receive1} and AP2={idle2, send2, reveive2}. With the help of the link function $L1$: $Q1 \to 2^{AP1}$ and $L2$: $Q2 \to 2^{AP2}$, the state can be associated with these Boolean labels. What should be emphasized here is that as each AP element is a Boolean value, there exist two expressions of each label: "true" or "negation". Therefore, by this link function, if we want to associate one state with one true AP element, we can directly label it; if we want to associate one state with one negation AP element, we can use "¬" marked before AP to label it (we denote the negation logic AP element with symbol "¬").

Due to the function and the performance of the components, in *G1* and *G2*, the labeling result is $L1(A)$=idle1; $L1(B)$=send1; $L1(C)$=receive1; $L2(D)$=idle2; $L2(E)$=send2; $L2(F)$=receive2.

3.2 Boolean label conjunction via automata composition

It is introduced in Subsection 2.1 that the component models are composed to automatically generate the system model. Therefore, the study of Boolean label conjunction via automata composition can help obtain the labels of all the states in the system model. This step is important to treat the scalability in our approach.

For two automata $G1 = (Q1; E1; f1; x_{01}; X_{m1}; \text{AP1}; L1)$ and $G2 = (Q2; E2; f2; x_{02}; X_{m2}; \text{AP2}; L2)$. The composition of them is

$$G1//G2 = A_c\,(Q1 \times Q2; E1 \cup E2; f;\ x_{01} \times x_{02};\ X_{m1} \times X_{m2},\ \text{AP1} \cup \text{AP2},\ L1 \wedge L2) \tag{7}$$

where AP1 ∪ AP2 means the union of two atomic proposition sets, such that the composition of the transceiver models is AP1 ∪ AP2= {idle1, send1, receive1, idle2, send2, reveive2}. The link function in automaton $G1//G2$ is connected with the "and" logic with two local link functions: $L1$: $Q1 \to 2^{AP1}$ and $L2$: $Q2 \to 2^{AP2}$. The global link function is then defined as

$$L1 \wedge L2 : Q1 \times Q2 \to 2^{AP1 \cup AP2}. \tag{8}$$

To be explained in more detail, we assume two states x and y, where $x \in Q1$ and $y \in Q2$. We use (x, y) to denote the product state from x and y belonging to the system model $(x, y) \in Q1 \times Q2$, and we use $L(x)$, $L(y)$, $L(x, y)$ to denote state AP labels, so we can achieve

$$L(x,y) = L(x) \wedge L(y). \tag{9}$$

The label function $L1 \wedge L2$ of the global automata $G1//G2$ is shown in Table 1.

Table 1 Composed AP via automata

State name	Composed AP	State name	Composed AP
A, *D*	idle1 ∧ idle2	*B*, *F*	send1 ∧ receive2
A, *E*	idle1 ∧ send2	*C*, *D*	receive1 ∧ idle2
A, *F*	idle1 ∧ receive2	*C*, *E*	receive1 ∧ send2
B, *D*	send1 ∧ idle2	*C*, *F*	receive1 ∧ receive2
B, *E*	send1 ∧ send2	—	—

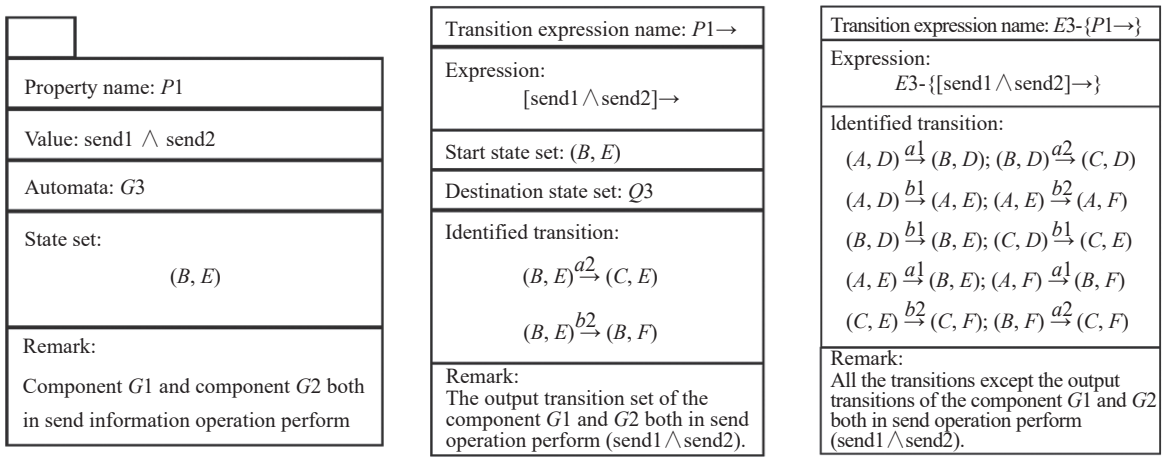
3.3 State identification

As all the states in the system model are already linked with Boolean labels, it is possible to identify the states using a Boolean property. If the Boolean labels satisfy the property in logic, the states are identified.

To clearly identify the states, as shown in Fig. 3, an arrow "→" is applied to denote a certain transition between two states. On the left of "→" is the starting state, on the right side is the destination state, and on the top is the transition name. A box marked on the left top is the "state identification box" to contain the identifying

information. This box will be a pre-processed document that is conveniently prepared for our further specification design.

Here, we present instructions of the transition identification box. Depicted in Fig. 3, as (B, E) is identified by P1 in the state identification box; in the middle part, according to the transition function $f3$ in $G3$ and (12), the existing transitions starting from (B, E) are presented in the transition identification box, namely $P1\rightarrow$; on the right side is the complement set operation $E3\text{-}\{P1\rightarrow\}$, which includes all the transitions except when both $G1$ and $G2$ are in a sending state.



(a) State indentification box (b) Transition indentification: $P1\rightarrow$ (c) Transition indentification: $E3\text{-}\{P1\rightarrow\}$

Fig. 3 State identification and transition indentification associated with the state name (B, E)

In the state identification box, from top to bottom, there are five floors, written in bold: property name; property value (the property value consists of several Boolean values from the set of AP); automata (which automaton the property is operated on); state set (the states who satisfy this property); Remark for this property.

For the purpose of state identification, according to the system management requirement context, the identification property will be established in the form of AP. When there is a need to extract several states that share certain similar significance from the universe set, primarily it should define a formal identification property to describe this significance.

The state identification property uses the Boolean labels in the component models to describe "what kind of states are considered according to intelligent operations". The Boolean labels are connected with the conjunction "and" logic (symbol "$\wedge$"), disjunction "or" logic (symbol "$\vee$"), and also applied with the negation ("$\neg$") logic. Various logic combinations are able to establish all complex Boolean algebra, ensuring the establishment of various state-identifying properties. For example, considering global model $G3$, the property value "idle1$\vee$idle2" can express "at least one of the transceivers is idle".

Given an automaton $G = (Q; E; f; x_0; X_m; \text{AP}; L)$, an identified state set IS(Φi), whose state label $L(x)$ satisfies a property value Φi, is

$$\text{IS}(\Phi\text{i}) = \{x \mid x \in Q,\ L(x) \vDash \Phi\text{i}\} \tag{10}$$

IS(Φi) is a subset of automaton state set Q. If no state label satisfies the property, IS(Φi) will be an empty set. Here, the symbol "$\vDash$" is applied to denote $L(x)$ is a sufficient condition of Φi.

The identification judgment formula compares whether the state label is a sufficient condition of the identification property. Sufficient condition judgment is available in most logic analysis tools, such as reduced ordered binary decision diagrams (BDDs) [33]. Here is a short example using BDD order "imp" (means "imply") to judge the sufficient condition. As shown in the state identification box of Fig. 3, by scanning Table 1 and using "imp" to illustrate (10) in BDDs, when inputting the state label of (B, E) and the property $P1$= send1$\wedge$send2, the judgment result presents "T" (means "true"). This implies that $L3(B, E) \vDash$ P1 and the wanted state is (B, E).

3.4 Transition identification

As the states are successfully identified, the identification of the transitions is also achievable according to the transition function of the automaton.

According to the transition function of $G3$, the transition function corresponding to transition $a1$ is $f3((A, D), a1) = (B, D)$, so the arrow format should be

$(A, D) \xrightarrow{al} (B, D)$.

Furthermore, the arrow can also denote a set of identified transitions between two identified state sets. There are three forms of the identified transitions, and these forms are explained by (11), (12), and (13). Given an automaton $G = (Q; E; f; x_0; X_m; \mathrm{AP}; L)$ and two state identifying properties Φi and Φj. Based on Φi and Φj(10), resulting identified state sets are IS(Φi) $\subseteq Q$ and IS(Φj) $\subseteq Q$. Based on the transition function f in G, the identified transitions are

$$\mathrm{IS}(\Phi\mathrm{i}) \rightarrow \mathrm{IS}(\Phi\mathrm{j}) = \begin{cases} \exists x \in E \,|\, f(q,x) = q', \ q \in \mathrm{IS}(\Phi\mathrm{i}); \\ \qquad q' \in \mathrm{IS}(\Phi\mathrm{j}) \\ \varnothing, \ \text{else} \end{cases} \tag{11}$$

$$\mathrm{IS}(\Phi\mathrm{i}) \rightarrow = \begin{cases} \exists x \in E \,|\, f(q,x) = q', \ q \in \mathrm{IS}(\Phi\mathrm{i}); q' \in Q \\ \varnothing, \ \text{else} \end{cases} \tag{12}$$

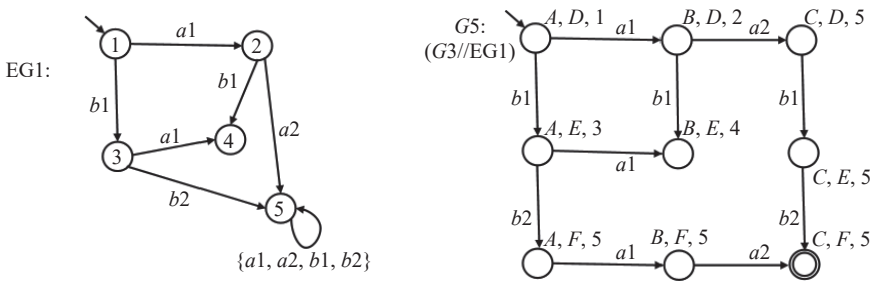
$$\rightarrow \mathrm{IS}(\Phi\mathrm{j}) = \begin{cases} \exists x \in E \,\big|\, f(q,x) = q', \ q \in Q; q' \in \mathrm{IS}(\Phi\mathrm{j}) \\ \varnothing, \ \text{else} \end{cases} \tag{13}$$

Here, (11) represents the transitions identified by traversing from the state q in IS(Φi) as the starting state to the state q' in IS(Φj) as the ending state. Equation (12) expresses the transitions identified by traversing from the state q in IS(Φi) as the starting state to the state q' in the state set Q as the ending state. Equation (13), on the other hand, represents the transitions identified by traversing from the state q in the state set Q as the starting state to the state q' in IS(Φj) as the ending state.

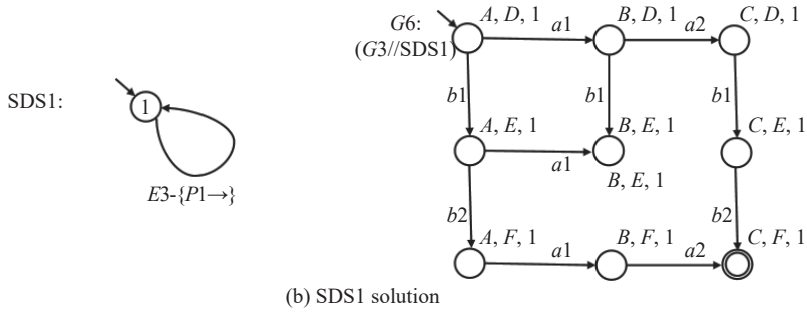
A transition identification box is used to hold these transitions, depicted in the transition identification box of Fig. 3. To be distinguished with the state identification box, there is no "small box marked on the left top" in the outward appearance. There are six floors in this box from top to bottom: the transition expression name; the transition expression; start and destination state sets; identified transition is from the definition of "x" in (11), (12), (13), and they are shown in the form of certain start state, destination state, and transition name with symbol "→"; the bottom floor is the remark of the transitions. Moreover, to operate on the transition set, two set operations are applied: complement set operation (operation symbol "–") and union set operation (operation symbol "+").

3.5 Boolean semantic specifying solution

To implement our wanted specifying approach, a novel specification boolean semantic direct expressing specification automaton (denoted by SDS) is created. In this SDS, the Boolean semantic is used to effectively and easily express intelligent operations, replacing the complex states/transition plan in the specification automaton with the traditional approach, and there is no need to consider the complex details of the system model. To parse the Boolean semantics, the mentioned states and transitions are pre-processed by identification boxes, a process that can be automatically treated by computers. As shown in Fig. 4, if the operation is set "To guarantee safety and these two transceivers are both sent, the whole system must be stopped (no further performance)", SDS1 is easier to establish than the traditional specification model EG1. To establish EG1, by the traditional approach, it must first consider the trajectory sequences from initial states to state (B, E). This indicates that when two transceivers are both in the sending state, the state (B, E) should become a blocking state. Finally, the other transitions respect the same transiting detail of system model $G3$. If there are a large number of states and transitions in the system model, the workload for planning trajectory sequences will be extremely substantial. However, when establishing SDS1, only one Boolean semantic "$E3$-$\{P1\rightarrow\}$" is needed, this means that all the transitions are allowed except when $G1$ and $G2$ are both in the sending state (supported by the identification boxes in Fig. 3). There is no need to consider the complex detail of $G3$. A notable highlight is the robust adaptability of SDS1, extending its applicability to any other system incorporating such intelligent operations. On the right side of Fig. 4, it can be seen that these two specifications have the same result (on the right side of Fig. 4).



(a) Traditional specification EG1



(b) SDS1 solution

Fig. 4 Comparison of traditional specification and SDS solution

SDS is formally defined as a Boolean enhanced automaton. SDS= (Q_{SDS}; IT_{SDS}; f_{SDS}; x_{0SDS}; x_{mSDS}, S_{SDS}), Q_{SDS} is the state set, IT_{SDS} is the identified transition as introduced in the identified transition box and $IT_{SDS} \subseteq E$, x_{0SDS} is the initial state, x_{mSDS} is the marked state and S_{SDS} stands for the Boolean semantics of the transitions.

The composing operation of SDS and system model G is

$$GS = A_c(Q_G \times Q_{SDS};\ E_{GS};\ f_{GS};\ x_{0G} \times x_{0SDS};\ x_{mG} \times x_{mSDS};\ AP_{GS};\ L_{GS}) \tag{14}$$

where $Q_G \times Q_{SDS}$ is the state set of the Cartesian product result; E_{GS} is the transitions set; $x_{0G} \times x_{0SDS}$ and $x_{mG} \times x_{mSDS}$ are the initial state and marked state, respectively. In some situations, there is no need to consider the marked state, so the marked state can be dismissed while the specifying function is also achievable; AP_{GS} is the AP set in GS, which remains the same as the AP set of automaton G, $AP_{GS} = AP_G$. Because there are no new labels created in SDS, after the composition, the label of GS is its original state in G. It is formally explained by

$$L_{GS}(x', y') = L_G(x) \tag{15}$$

where $x, x' \in Q_G$, $y' \in Q_{SDS}$, $(x', y') \in Q_{GS}$.

$$f_{GS}((x,y),e) = \begin{cases} (x',y'), & \text{condition(1)} \\ (x,y'), & \text{condition(2)} \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{16}$$

where condition (1) is $E_{SDS} \subseteq E_G$, $e \in E_G \cap E_{SDS}$, $(x, x') \in Q_G$, $f_G(x, e)=x'$ and $(y, y') \in Q_{SDS}$, $f_{SDS}(y, e)=y'$. Condition (2) is $E_{SDS} \not\subset E_G$, $e \in Q_{SDS}$ but $e \notin Q_G$, $(y, y') \in Q_{SDS}$, $f_{SDS}(y, e)=y'$.

For condition (1), "e" is the intersection of E_G and E_{SDS}. The identified transition is from the system model G, mapping to SDS. The identified transition is required not only to have the same name but also to transit in the same way as in G. Assuming in G that there are two identified states x and x': according to $L_G(x) \vDash \Phi 1$ and $L_G(x') \vDash \Phi 2$ and the transition function $f_G(x, e)=x'$, an identified transition "e" in G is about the transition from state x to state x': $x \xrightarrow{e} x'$; moreover, transition e is applied into the specification automaton SDS by the form: $y \xrightarrow{x \xrightarrow{e} x'} y'$, so based on the transition function f_{GS}, in the resulting automaton GS, the state transition is $(x,y) \xrightarrow{e} (x,y')$.

For condition (2), "e" is not an identified transition, and "e" is an additional transition compared to E_G, it will follow the normal composing principle introduced in (3).

To illustrate an intelligent operation, the Boolean semantic is established by a combination of "→" and property value "Pi" stands for the identified transitions, while transition set operations "+" or "−" can be used.

Decipted in Fig. 4 SDS1, the transition function of SDS1 is explained: $f_{SDS1}(1, (E3\text{-}(P1\rightarrow)))=1$ and the identified transitions in $G3$ is all the transitions except $(B,E) \xrightarrow{a2} (C,E)$ and $(B,E) \xrightarrow{b2} (B,F)$, so the transition function is assumed to be $f_{G3}((B, E), (E3\text{-}(P1\rightarrow))) \neq (C, E) \neq (B, F)$, based on (16), the transition function in $G6$ is $f_{G6}((B, E, 1), (E3\text{-}(P1\rightarrow))) \neq (C, E, 1) \neq (B, F, 1)$. This proves that there is no output transition from state $(B, E, 1)$ to $(C, E, 1)$ or to $(B, F, 1)$, as shown in $G6$ of Fig. 4, satisfying this operation requirement: "the system must be stopped, when two components are sending at the same time".

4. Case of study

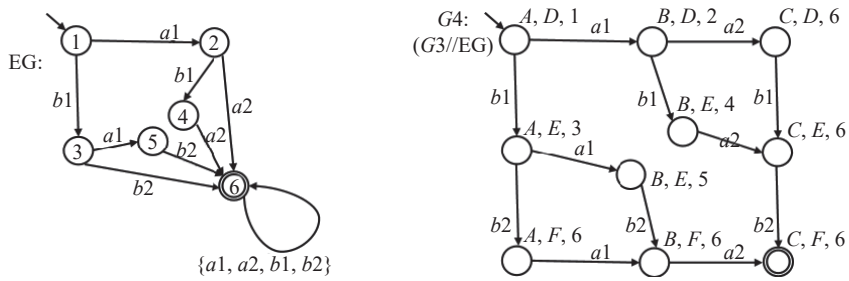
In industrial systems, there are always several cooperating subsystems, and these subsystems have their own complex behaviours, such as the transceivers in the example. Therefore, a global model that can describe all behaviours of the industrial system is needed. In addition, the complex intelligent operations are multiplied and dynamically set on to the system. The modelling solution is as follows. First, a component model is established to describe the component behaviours. Second, the global system model can be automatically generated by composing the component models. Third, SDS is established based on the operations, and the related state and transition identification boxes can be automatically processed. Finally, by composing the SDS and system model, the system model evolutes in accordance with intelligent

operations.

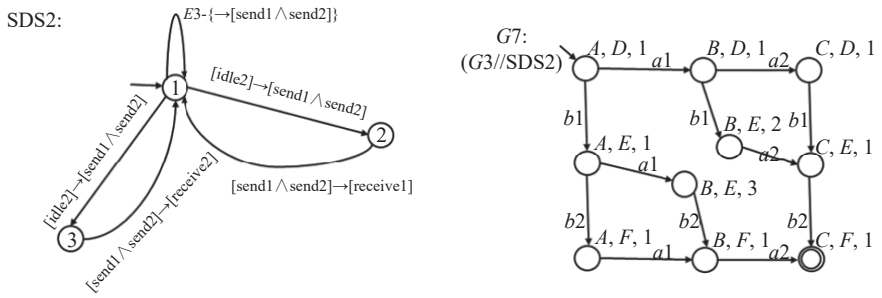
This part studies the specifying operation of the "first send, first receive" for system $G3$ (shown in Fig. 1). As $G3$ is already generated by the form of Boolean-enhanced automata composition in Subsection 3.2, this section only focuses on the specifying task.

Faced with this operation, the Boolean semantic should convey either "subsystem $G1$ first sends and then $G1$ is ordered to receive first" or "subsystem $G2$ first sends and then $G2$ is instructed to receive first".

The respecting specification automaton SDS2 is presented in Fig. 5 and synchronized with the global model $G3$. We obtain the result $G7$. Because the management requirement context remains the same, we can obtain the same state splitting result as the normally used specification automaton compared to $G4$ in Fig. 5. In specification automaton SDS2, the key solved problem is as follows: when two subsystems are both in sending state, we should identify that if $G1$ first sends, then $G1$ should be specified to receive first, and the same principle that if $G2$ first sends, then $G2$ should receive first. Through comparing EG and SDS2, we can summarize the advantages of our modeling approach: SDS2 is applicable to all systems incorporating the principle of "first send, first receive" while EG is only suitable for individual systems. Furthermore, when there are changes or additions to the operations, SDS2 does not require model resubmission. However, EG necessitates re-planning trajectory paths, leading to lower efficiency and difficulty in implementation. Therefore, this universe modeling approach holds significant potential and can be applied to future complex intelligent industrial systems.



(a) Traditional specification EG



(b) SDS2 solution

Fig. 5 Case of study

Depicted in SDS2 of Fig. 5, the initial state 1 has a self-loop transition $E3$-{→[send1∧send2]}. By this Boolean semantic, we can see that in this state, all the transitions of $E3$ will remain unchanged except the input transitions resulting in the states "$G1$ and $G2$ are both in the sending state". From state 1 to state 2 is the transition [idle2]→[send1∧send2], which means it considers the transition from $G2$ in idle performance to $G1$ and $G2$ both in send performmance, indicating that $G1$ first sends. From state 2 to state 1 is the transition [send1∧send2]→[receive1]. This implies that the only allowed transition from state 2 to state 1 is from two systems both in send performance to $G1$ in receive performance, indicating that $G1$ first receives. By this design, when transitioning from state 1 to state 2 and then back to state 1, the specification automaton will ensure that when facing $G1$ and $G2$ both in send performmance, the only guaranteed transition trajectory is that if $G1$ first sends, it will also be the first to receive. The result in $G7$ is $(B,D,1) \xrightarrow{b1} (B,E,2) \xrightarrow{a2} (C,E,1)$. The same principle applies to state 3, ensuring that if $G2$ first sends and then $G2$ first receives by specifying that the result in $G7$ is $(A,E,1) \xrightarrow{a1} (B,E,3) \xrightarrow{b2} (B,F,1)$.

Fig. 6 is the state identification boxes that serve the transition boxes based on (10). Fig. 7 presents the transition identification boxes, which serve the semantics based

on (11), (12), and (13). The identification boxes of states and transitions can be automatically processed by a spooler, and the only manual task is designing the Boolean semantics.

Property name: $P1$	Propery name: $P2$	Propery name: $P3$	Propery name: $P4$	Propery name: $P5$
Value: send1∧send2	Value: receive2	Value: receive1	Value: idle2	Value: idle1
Automata: $G3$	Automata: $G3$	Automata: $G3$	Automata: $G3$	Automata: $G3$
State set: (B, E)	State set: $(A, F), (B, F), (C, F)$	State set: $(C, D), (C, E), (C, F)$	State set: $(A, D), (B, D), (C, D)$	State set: $(A, D), (A, E), (A, F)$
Remark: Component $G1$ and component $G2$ are in both send operation perform.	Remark: Component $G2$ is in receive operation perform, component $G1$ is in any possible operation perform.	Remark: Component $G1$ is in receive operation perform, component $G2$ is in any possible operation perform.	Remark: Component $G2$ is in idle operation perform, component $G1$ is in any possible operation perform.	Remark: Component $G1$ is in idle operation perform, component $G1$ is in any possible operation perform.

Fig. 6 State identification

Transition expression name: $\rightarrow P1$	Transifion expression name: $E3\text{-}\{\rightarrow P1\}$	Transition expression name: $P5\rightarrow P1$
Expression: $\rightarrow$\{send1∧send2\}	Expression: $E3\text{-}\{\rightarrow$\{send1∧send2\}\}	Expression: \{idlel\}$\rightarrow$\{send1∧send2\}
Start slate set: $Q3$	Identified transition: $(A, D) \xrightarrow{a1} (B, D)$; $(B, D) \xrightarrow{a2} (C, E)$ $(A, D) \xrightarrow{b1} (A, E)$; $(A, E) \xrightarrow{b2} (A, F)$ $(C, D) \xrightarrow{b1} (C, E)$; $(B, E) \xrightarrow{a2} (C, E)$ $(B, E) \xrightarrow{b2} (B, F)$; $(B, E) \xrightarrow{b2} (C, F)$ $(A, F) \xrightarrow{a1} (B, F)$; $(B, F) \xrightarrow{a2} (C, F)$	Start state set: $(A, D), (A, E), (A, F)$
Destination state set: (B, E)	Remark: All the transitions except the input transition of the component $G1$ and $G2$ are both in send operation perform \{send1∧send2\}.	Destination state set: (B, E)
Identified transition: $(A, E) \xrightarrow{a1} (B, E)$ $(B, D) \xrightarrow{b1} (B, E)$		Identified transition: $(A, E) \xrightarrow{a1} (B, E)$
Remark: The input transition set of the component $G1$ and $G2$ are both in send operation perform \{send1∧send2\}.		Remark: The transition set: the component $G1$ in idle perform\{idlel\} evolution to component $G1$ and $G2$ both in send operation perform \{send1∧send2\}.

Transifion expression name: $P4\rightarrow P1$	Transition expression name: $P1\rightarrow P2$	Transition expression name: $P1\rightarrow P3$
Expression: \{idle2\}$\rightarrow$\{send1∧send2\}	Expression: \{send1∧send2\}$\rightarrow$\{receive2\}	Expression: \{send1∧send2\}$\rightarrow$\{receivel
Start state set: $(A, D), (B, D), (C, D)$	Start state set: (B, E)	Start state set: (B, E)
Destination state set: (B, E)	Destination state ser: $(A, F), (B, F), (C, F)$	Destination state set: $(C, D), (C, E), (C, F)$
Identified transition: $(B, D) \xrightarrow{b1} (B, E)$	Identified transition: $(B, E) \xrightarrow{b2} (B, F)$	Idenfified transition: $(B, E) \xrightarrow{a2}$ (C,E)
Remark: The transition set: the component $G2$ in idle perform\{idle2\} evolution to component $G1$ and $G2$ both in send operation perform \{send1∧send2\}.	Remark: The transition set: the component $G1$ and $G2$ in send perform \{send1∧send2\} evolution to component $G2$ in receive perform \{receive2\}.	Remark: The transition set: the component $G1$ and $G2$ in send perform\{send1∧send2\} evolution to component $G1$ in receive perform \{receivel\}.

Fig. 7 Transition identification

5. Conclusions

For industrial systems, model-based system engineering is applied for system analysis, simulation, control, and property verification. Various intelligent operations require a stochastic specifying approach based on the original system model, making it a complex, tough, and erratic task. Our research is able to be simple and uniform by designing the Boolean semantics directly on the basis of the operations.

To ensure scalability, our approach starts from the treatment of component behaviours, necessitating only basic information of each component. Thanks to the composition formulas in our approach, the global system model becomes attainable, resolving the problem of "super large-scale systems model implementation impossible by manual".

In terms of agility, facing the problem that various specifying demands will lead to erratic implementation work, an interesting point in our approach is that no mat-

ter how changeable the system model is, one intelligent operation will always remain to be the unique specification model. It has the strong compatibility that there is no need to change the Boolean semantic specification automaton design whenever the model is changed or even a new industrial system is given. If two or more operations are added to the system, the system engineers only need to design each SDS based on each operation and automatically compose them with the system model; thus, the model is multiply specified.

Concerning universality, the application of our proposed approach will be considerable. Thanks to the large scope of DES framework applied objects, if one industrial system happens in performance and reasonable events, the model implementation and evolution is achievable through our approach. Furthermore, due to the powerful expression function of Boolean semantics, this approach is nearly able to specify all kinds of operations.

References

[1] AKUNDI A, ANKOBIAH W. Mapping industry workforce needs to academic curricula—a workforce development effort in model-based systems engineering. Systems Engineering, 2024. https://doi.org/10.1002/sys.21745.

[2] YAN R, DUNNETT S J, JACKSON L M. Model-based research for aiding decision-making during the design and operation of multi-load automated guided vehicle systems. Reliability Engineering & System Safety, 2022, 219: 108264.

[3] MADNI A M, SIEVERS M. Model-based systems engineering: motivation, current status, and research opportunities. Systems Engineering, 2018, 21(3): 172–190.

[4] CLAUDIA P, VALENTIN R, TUNC A. A framework for verifying dynamic probabilistic risk assessment models. Reliability Engineering & System Safety, 2020, 203: 107099.

[5] PAKONEN A, BUZHINSKY I, BJORKMAN K. Model checking reveals design issues leading to spurious actuation of nuclear instrumentation and control systems. Reliability Engineering & System Safety, 2021, 205: 107237.

[6] CASSANDRAS C G, CHRISTOS G, STEPHANE L. System and models, in introduction to discrete event systems. Berlin: Springer Science & Business Media, 2009.

[7] JOSEPH H S, ELIZABETH A S, FRANCESCA M F, et al. Accident precursors, near misses, and warning signs: critical review and formal definitions within the framework of discrete event systems. Reliability Engineering & System Safety, 2013, 114: 148–154.

[8] LIU Y, LIU Q Z, XIE C Y, et al. Reliability assessment for multi-state systems with state transition dependency. Reliability Engineering & System Safety, 2019, 188: 276–288.

[9] CASSANDRAS C G, LAFORTUNE S. Introduction to discrete event systems. Cham: Springer, 2021.

[10] WOODS E J, KANNAN D, SHARPE D J. et al. Analysing ill-conditioned Markov chains. Philosophical Transactions of the Royal Society A-Mathematical Physical and Engineering Sciences, 2023, 381(2250): 20220245.

[11] TANG Y H, MOOR T. Compositional non-blockingness verification of finite automata with prioritised events. Discrete Event Dynamic Systems-Theory and Applications, 2024, 34(1): 125–161.

[12] YUAN X H, JIAN J J, CHAI Z, et al. Markov chain signal generation based on single magnetic tunnel junction. IEEE Electron Device Letters, 2023, 44(12): 1963–1966.

[13] SERFOZO R. Basics of applied stochastic processes. Cham: Springer Science & Business Media, 2009.

[14] STROOCK D W. An introduction to Markov processes. Cham: Springer Science & Business Media, 2013.

[15] GAGNIUC P A. Markov chains: from theory to implementation and experimentation. New York: John Wiley & Sons, 2017.

[16] SERFOZO R F. An equivalence between continuous and discrete time markov decision processes. Operations Research, 1979, 27(3): 616–620.

[17] EVERDIJ M H C, KLOMPSTRA M B, BLOM H A P, et al. Compositional specification of a multi-agent system by stochastically and dynamically coloured petri nets. Stochastic Hybrid Systems: Theory and Safety Critical Applications. 2006, 337: 325−350.

[18] ROTH O. Type automata. Proc. of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, 2023: 1537−1540.

[19] JORDAAN S, TIMM N, MARSHALL L. Formal methods: foundations and applications. Proc. of the 26th Brazilian Symposium, 2023: 131−140.

[20] DESTOUET C, TLAHIG H, BETTAYEB B, et al. Flexible job shop scheduling problem under industry 5. 0: a survey on human reintegration, environmental consideration and resilience improvement. Journal of Manufacturing Systems, 2023, 67: 155–173.

[21] YIN S, LI X W, GAO H J, et al. Data-based techniques focused on modern industry: an overview. IEEE Trans. on Industrial Electronics, 2015, 62(1): 657–667.

[22] MAIDANA R G, PARHIZKAR T, GOMOLA A, et al. Supervised dynamic probabilistic risk assessment: review and comparison of methods. Reliability Engineering & System Safety, 2023, 230: 108889.

[23] DELANEY W. Dynamic models and discrete event simulation. Boca Raton: CRC Press, 2020.

[24] POWER D J. Specifying an expanded framework for classifying and describing decision support systems. Communications of the Association for Information Systems, 2004, 13(1): 158−166.

[25] WINTENBERG A, BLISCHKE M, LAFORTUNE S, et al. A general language-based framework for specifying and verifying notions of opacity. Discrete Event Dynamic Systems-Theory and Applications. 2022, 32(2): 253−289.

[26] BOUSSIF A, GHAZEL M, BASILIO J C. Intermittent fault diagnosability of discrete event systems: an overview of automaton-based approaches. Discrete Event Dynamic Systems-Theory and Applications, 2021, 31(1): 59−102.

[27] DESGEORGES L, PIRIOU P Y, LEMATTRE T, et al. Formalism and semantics of PyCATSHOO: a simulator of distributed stochastic hybrid automata. Reliability Engineering & System Safety, 2021, 208: 107384.

[28] QIU S H, CUI X P, PING Z W, et al. Deep learning techniques in intelligent fault diagnosis and prognosis for industrial systems: a review. Sensors, 2023, 23(3): 1305.

[29] ANTONIO T. Discrete-event system theory: an introduction. Singapore: World Scientific Publishing Company, 1995.

[30] AGESEN O. The cartesian product algorithm: simple and precise type inference of parametric polymorphism. Proc. of the 9th European Conference, Object-Orienteel Programming, 1995: 2−26.

[31] XU C Y. Operational dependability model generation. Lyon: University of Lyon, 2020.
[32] BAIER C, KIEFER S, KLEIN J, et al. Markov chains and unambiguous automata. Journal of Computer and System Sciences, 2023, 136: 113–134.
[33] WEAVER S, FRANCO J, SCHLIPF J. Extending existential quantification in conjunctions of BDDs. Journal on Satisfiability, Boolean Modeling and Computation, 2006, 1(2): 89−110.

Biographies

XU Changyi was born in 1989. He received his B.S. degree in electronic information science and technology from Jilin University, Changchun, China, in 2012, M.S. degree from Chinese Academy of Sciences, Changchun, China, in 2015, and Ph.D. degree in automatic from Institut National des Sciences Appliquees de Lyon, Lyon, France, in 2021. He is currently an associate professor with the School of Control Science and Engineering, Dalian University of Technology, Dalian, China. His research interests include systems engineering, electronic technology, aero-engine, artificial intelligence, control theory and practice, and systems reliability.
E-mail: ChangyiXU@dlut.edu.cn

WANG Yun was born in 2000. She received her B.S. degree from Hebei University of Technology in 2021. She is currently pursuing her M.S. degree with the School of Control Science and Engineering, Dalian University of Technology, Dalian, China. Her research interests include systems engineering, electronic technology, and control theory and practice.
E-mail: wangyun@mail.dlut.edu.cn

DUAN Yiman was born in 1996. She received her B.S. degree in mechanical design, manufacture, and automation from Yanshan University, Qinhuangdao, China, in 2019, and M.S. degree in petroleum and natural gas engineering from Yanshan University, Qinhuangdao, China, in 2022. She is currently working toward her Ph.D. degree in the College of Mechanical Engineering, Zhejiang University, Hangzhou, China. Her research interests include systems engineering, control theory and practice, and high-performance mechatronic equipment.
E-mail: ymduan@zju.edu.cn

ZHANG Chao was born in 1996. He received his B.S. and M.S. degrees from Northwestern Polytechnical University in 2012 and 2015, respectively. He received his Ph.D. degree from the University of Lyon in 2019. Currently, he is a professor at the Institute of Mechatronics and Control Engineering, Zhejiang University, Hangzhou, China. His research interests include systems engineering, control theory and practice, and high-performance mechatronic equipment.
E-mail: chao.zhang@zju.edu.cn