MODELING AND OPTIMAL SUPERVISION OF MULTI-SENSORS AND MULTI-STATES DISCRETE-EVENT SYSTEMS WITH STRUCTURED LANGUAGES

Shunping Ji
College of Mechanical and Electrical Engineering
Sanjiang University, 210012,
Nanjing, China
Emails: jishunping@qq.com

Submitted: Aug. 17, 2016 ***Accepted: Nov. 3, 2016*** ***Published: Dec. 1, 2016***

Abstract- In a discrete-event system, more than one sensor will create more than one event. Several events may happen at the same time parallelly. Events in different parallel branches are irrelevant and it will lead states explosions if mixing them together and enumerating all possible arrangements. A system with parallel sequences has multi-threads and can be dealt with as a whole. In this paper, after analyzing the process of discrete-event systems, the new model is established as a 5-tuple set, which is called a generator with multi-threads. The new generator will generate structured languages which are in accordance with the sequences of the process. The controllability of the structured languages is defined and the theorem about the existence of the supervisor for given languages is presented and proved. For given specifications, the method to solve the supervisor is presented. With the new model, the states of the total system will not explode when two or more systems synchronizing. From two examples shown in Section VI, the new model and supervisor theory for multi-sensors and multi-states discrete-event system are more convenient and natural than that of the traditional theories.

Index terms*: Discrete-event system, multi-sensors and multi-states, structured languages, supervisory control.*

I. INTRODUCTION

A discrete-event system (DES) is a system in which events are discrete in time. In DES, however, a state has duration in time and different states transit to others with the happening of events. A practical manufacturing system often contains many sensors and actuators, which make the system complex [1-3]. Manufacturing system is a kind of DES. A signal from one sensor will create an event which will drive the DES to go ahead. When a DES has multi-sensors, signals from more than one sensor will create more events at the same time. Then the DES will have multi-sensors and multi-states (MSMS).

This paper focuses on the DES with MSMS. In MSMS DES, a system could be at multiple states at the same time and could have multiple impending events at the same time.

Many research studies have been carried out on discrete-event system (DES). Wonham [4] and Cassandras[5] introduced the theory of supervisory control of DES. Every discrete event is described by a letter and different letters construct an alphabet. A DES is usually modeled as a generator, which will generate languages, which are strings containing some letters from the alphabet. Language is used to represent the process of the DES. The supervisory control uses a supervisor to supervise the generator to avoid the undesired behaviors. Cieslak and Randy study the supervisory control of discrete-event processes with partial observations [6].

Some subsequent works on DES are focused on timed DES (e.g. Khatab 1998[7], Zhang 2013[8]). Timed DES has time dimension and make the system have time properties. The hierarchical control with modularity are effective ways to deal with a distributed system (e.g. Schmidt, Marchand and Gaudin 2006[9]; Leduc, Lawford and Dai 2006[10]; Fekri 2008[11]). For distributed systems such as flexible manufacturing systems (FMS) and communication networks, a modular control architecture can be more suitable than a monolithic centralized one. Takai [12] and Su [13] study the DES with concurrent events. All those achievements assume that the system could be only at one state at the same time and also only one event could happen at the same time, which is a simple way to deal with DES. When synchronizing different subsystems, the system will be exploded in states.

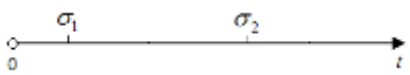
In recent years, state tree structures (STS) are used to describe DES (e.g. Wang 2016[14]). STS could remit the problem of states explosions, but modeling and mathematic descriptions are complex.

Petri net has been used to analyze DES for many years [15-17]. Branislav and Zhou [15] concentrate on Petri nets and their use in the modeling and control design for DES, which serve as a basis for extending to other tools and approaches such as supervisory control theory and other functions. When dealing with the DES, the reachable graphs of Petri nets will cause states explosions too. Another important issue about the modeling of DES is about the robotic models. Jayasiri and Mann [18] use supervisory control of fuzzy discrete event systems to control mobile robotics; Yao [19] uses infinitesimal perturbation analysis of stochastic flow models to analyze DES; Su [20] presents an abstract way to model the nondeterministic finite-state automata in DES. Chen, Kuo and Chen [21] present the modeling of hierarchical robotic discrete-event system. All those achievements present ways to deal with nondeterministic DES, but the problem of states explosions remains.

Most of those works pay more attentions to the single event and single state DES. Some works are related to the MSMS DES, but most of which are based on theory of Petri nets rather than automata.

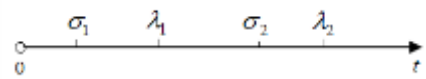
Let's investigate a small question of a DES. Let σ_1, σ_2, λ_1 and λ_2 be 4 events, event σ_2 happens after σ_1 and λ_2 happens after λ_1, shown as Figure1(a) and Figure1(b). There are no known relationships between σ_1 (or σ_2) and λ_1 (or λ_2). Then how to model this system will be a question. The traditional method is to enumerate all the possible arrangements of those four events. If not considering the situations that two events happen simultaneously, there will be 6 possible arrangements, two of which are shown in Figure1(c) and Figure1(d). If considering the simultaneity situations, there will be 13 possible arrangements.

In fact, events σ_1, σ_2 and events λ_1, λ_2 are two parallel groups. There are no known relations between σ_1 (or σ_2) and λ_1 (or λ_2). In order to model this kind of systems, a new method with multi-threads is introduced in this paper to describe a complex process and its supervisor in a better way.

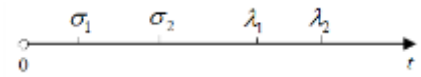


(a) Schedule of 2 related events

(b) Schedule of 2 related events



(c)Possible schedule of 4 events



(d)Another possible schedule of 4 events

Figure 1. Schedule of a DES

In this paper, we focus on the modeling and supervisor of MSMS DES. This paper proceeds as the following. In Section II, the concepts of single sequence, alternative structure and parallel structure are presented. With catenation and extended structure, processes could be illustrated by sequences of events. In Section III, modeling of MSMS DES is introduced and structured languages will be generated by the model. Section IV presents the controllability of the MSMS DES. Section V presents the optimal supervisor by automata control. Section VI formulates two examples with multi-sensors and multi-states. Section VII formulates an experiment of the parallel process plant. And finally in section VIII, the conclusions are drawn.

II. EVENT SEQUENCE AND PROCESS OF A DES

Events which happen in a determined order will construct an event sequence. Between every two events, system is at a state. Happening of an event, the system will transit from the previous state to the next state. A state has duration in time while an event happens instantly.

In DES, the system starts from initial states, and with the events happening it runs towards marked states; at last the system ends at marked states. This course of the system is called a process. In order to represent a process, the following definitions are presented.

Definition 1: (single sequence)

Let $\sigma_1, \sigma_2, \ldots, \sigma_n$ be a group of events of a system, and events $\sigma_1, \sigma_2, \ldots, \sigma_n$ happen in sequence from a state of the system to another state. Let σ_i (i =1,2,…, $n-1$) be an event, if the impending

event after σ_i is determined and single, the sequenced events construct a single sequence, which is denoted as a string $\sigma_1\sigma_2...\sigma_n$.

Definition 2: (alternative structure/ OR operation)

Let $str_1 = \sigma_1\sigma_2...\sigma_i$ and $str_2 = \tau_1\tau_2...\tau_j$ (i, j =1,2,… , σ_u and τ_v are events of a system, and u =1,2,…, i, v = 1,2,…, j) be two single sequences, if the impending event is only one of the two events σ_1 and τ_1, str_1 and str_2 construct an alternative structure. str_1 and str_2 are called alternative sequences (also called alternative branches) in the alternative structure. The whole activity of two strings could be represented by the OR operation: $str_1 + str_2$, in which the operator '+' means OR. And the result of the operation is either str_1 or str_2. From the definition, the operation is commutative and $str_1 + str_2 = str_2 + str_1$.

The two alternative sequences could be denoted as $str_1 + str_2 = (\sigma_1\sigma_2...\sigma_i, \tau_1\tau_2...\tau_i)$, in which the ',' means 'OR'.

Definition 3: (parallel structure/ AND operation)

Let $str_1 = \sigma_1\sigma_2...\sigma_i$ and $str_2 = \tau_1\tau_2...\tau_j$ (i, j =1,2,… , σ_u and τ_v are events of a system, and u =1,2,…, i, v = 1,2,…, j) be two single sequences, if the impending events are both event σ_1 and event τ_1, str_1 and str_2 construct a parallel structure. And str_1 and str_2 are called parallel sequences (also called parallel branches). The whole activity of two strings could be represented by the AND operation: $str_1 \| str_2$, in which the operator '||' means AND. And the result of the operation is both str_1 and str_2.

Definition 4: (catenation operation and extended structure)

To represent a more complex system, catenation operation could be used to catenate single sequences, alternative structures and parallel structures together. And alternative structures and parallel structures could be extended to more complex structures.

Each sequence in an alternative structure could be followed by an alternative structure or a parallel structure; each sequence in a parallel structure could be followed by an alternative structure or a parallel structure. That is to say, in an alternative structure or a parallel structure, other alternative structures or parallel structures could be embedded.

Example:

Let str_1, str_2, str_3, str_4, str_5 and str_6 be single sequences, $(str_1 \| (str_2\ (str_3\ ,\ str_4\ )\ str_5))\ str_6$ means that str_1 is parallel with $str_2\ (str_3\ ,\ str_4\ )\ str_5$ and then this parallel structure catenates str_6. The system is initialed at two states and ends at one marked state. The whole activity could also be illustrated as $\begin{pmatrix} str_1 \\ str_2 str_3 str_5, str_2 str_4 str_5 \end{pmatrix} str_6$.

Definition 5: (active sequence)

Let $\sigma_1\sigma_2...\sigma_n$ be a single sequence, the sequence $\sigma_1\sigma_2...\sigma_n$ is active if the system is at the state whose impending event is σ_i (i =1, 2, …, n).

Definition 6: (thread)

A thread is a control stream following an active single sequence [22]. A thread is included in a process. And a process could have multi-threads which follow different single sequences.

III. MODELING OF MULTI-SENSORS AND MULTI-STATES DISCRETE-EVENT SYSTEM

a. Modeling of MSMS DES

A process is generated by a DES system. Now we model a MSMS DES as a generator in a new way. The DES is modeled as a 5-tuple set. Then the languages generated by the generator will represent the process of a MSMS DES.

Let

$$G = (Q, \Sigma, \Delta, Q_o, Q_m) \quad (1)$$

Here Q is the state set, which is at most countable. $Q_o \subseteq Q$ is the initial state set. And $Q_m \subseteq Q$ is the set of marked states. Let $v_i = (s_{i1}, s_{i2}, \ldots, s_{im_i})^T$ be a vector of current states, $s_{ij} \in Q$ (i =0,1,2,…, j =1,2,…, m_i). The symbol m_i means the dimension of v_i, which is the number of current threads at v_i. At v_i, a state could be in different thread, so the situation that $s_{ij} = s_{ik}$ (j, k =1,2,…, m_i) is possible. The first state vector is $v_0 = (s_{01}, s_{02}, \ldots, s_{0m_i})^T$, and $s_{01}, s_{02}, \ldots, s_{0m_i}$ are elements of Q_o.

Σ is a finite alphabet of events, $\Sigma = \{\sigma_i \mid i = 0,1,2,3,\ldots\}$.

Δ is the set of transition function, $\Delta=\{\Delta_i,\ i=0,1,2,\ldots\}$. Δ_i is the transition to transit the system from v_i to v_{i+1}.Let $v_i=(s_{i1}, s_{i2}, \ldots, s_{im_i})^T$, let the impending events at s_{ij} be $\{\sigma_{i,j,1}, \sigma_{i,j,1}, \ldots, \sigma_{i,j,h}\}\subseteq\Sigma$, $j=1,2,\ldots, m_i$ and $h=1,2,3,\ldots$,

$\Delta_i:=(\delta_{i1}, \delta_{i2}, \ldots, \delta_{im_i})$, $i=0,1,2,\ldots$, $\delta_{ij}\in\{\sigma_{i,j,1}, \sigma_{i,j,2}, \ldots, \sigma_{i,j,h}\}\cup\{\varepsilon\}$, ε represents empty event, δ_{ij} represents the happening event at state s_{ij}. Then

$$v_{i+1}=v_i\,\Delta_i=\begin{Bmatrix} s_{i1}\delta_{i1} \\ s_{i2}\delta_{i2} \\ \cdots \\ s_{im_i}\delta_{im_i} \end{Bmatrix}=\begin{Bmatrix} s_{i+1,1} \\ s_{i+1,2} \\ \cdots \\ s_{i+1,m_{i+1}} \end{Bmatrix} \quad (2)$$

If $\delta_{ij}=\varepsilon$, then $s_{i+1,j}=s_{i,j}$, and the system still stays at the previous state. Of course, the system could be at many different states at the same time.

The number of the threads may be changed when the state vector transits from v_i to v_{i+1}. If $m_{i+1}=m_i$, the number of threads is not changed, which means no new threads produced after the transition. If $m_{i+1}>m_i$, new threads are produced. And if $m_{i+1}<m_i$, some of the old threads is ended. However, even the number of threads is not changed, there are also possible that some threads are produced and some ended.

Now let's investigate two typical situations: the beginning and end of parallel sequences. Let $\delta_{ij}\in\{\sigma_{i,j,1}, \sigma_{i,j,1}, \ldots, \sigma_{i,j,h}\}$ be the transition that transit from s_{ij} (at v_i) to parallel sequences, then $s_{ij}\,\delta_{ij}=\begin{Bmatrix} s_{i+1,1} \\ s_{i+1,2} \\ \cdots \\ s_{i+1,u} \end{Bmatrix}$, $u=1,2,\ldots$ and $s_{i+1,1}=s_{i+1,2}=\ldots=s_{i+1,u}$. Here transition δ_{ij} begins the new threads and makes the number of threads $u-1$ more than before.

At the end of parallel sequences, if all parallel sequences end asynchronously and a single sequence begins, then let s_{iu} be the last ended parallel sequence, $\begin{Bmatrix} s_{i,1} \\ s_{i,2} \\ \cdots \\ s_{i,u} \end{Bmatrix}\delta_{ij}=\{s_{i+1,p}\}, p\le m_{i+1}$, $s_{i,1}=s_{i,2}=\ldots=s_{i,u}$. If some parallel sequences end synchronously, the transition is similar.

b. Languages

According to the definition of 5-tuple set, G will be a generator that will generate strings from all initial states. The strings are constructed by letters in Σ. So every string represents a sequence of

the process. If the process has only one single sequence, the generator will generate a single string in which the letters are the names of events. We call this single string a single language on alphabet Σ.

If the process has more single sequences, which could be arranged as any structures of process, the generator will generate so-called structured languages on alphabet.

Definition 7: (structured language)

A generator G defined by the 5-tuple set generates strings from all initial states. Those single strings may construct alternative structures, parallel structures or extended structures which are the same as the sequence structures. And different structures could catenate with others or single string. After catenating, strings are linked to a whole, which is called structured language.

Remarks:

(1) The structures of a structured language are same as that of sequences.

(2) Catenations of single string and any structures are same as that of sequences.

(3) Languages, either single language or structured languages, represent the process of the system. Of course, single language is a basic situation of structured language.

The languages generated by G represent a process, in which events happen in series. The languages generated by G are called $L(G)$. And those $L(G)$ from the initial states to the marked states construct $L_m(G)$.

Lemma 1: (equivalent to a single thread way)

Let $L_m(G) = \left\{\begin{pmatrix} e_{11}e_{12} \\ e_{21}e_{22} \end{pmatrix}\right\}$ be a parallel structure, $\begin{bmatrix} e_i \\ e_j \end{bmatrix}$ represents that two events e_i and e_j happen simultaneously. $L_m(G)$ are equivalent to the expanded languages:

$$L_m^e(G) = \{ e_{11}e_{12}e_{21}e_{22},\ e_{21}e_{11}e_{12}e_{22},\ e_{21}e_{22}e_{11}e_{12},\ e_{11}e_{21}e_{12}e_{22},\ e_{11}e_{21}e_{22}e_{12},\ e_{21}e_{11}e_{22}e_{12},\ \begin{bmatrix} e_{11} \\ e_{21} \end{bmatrix} e_{12}e_{22},\ \begin{bmatrix} e_{11} \\ e_{21} \end{bmatrix} e_{22}e_{12},$$

$$e_{21}\begin{bmatrix} e_{11} \\ e_{22} \end{bmatrix} e_{12},\ e_{11}\begin{bmatrix} e_{12} \\ e_{21} \end{bmatrix} e_{22},\ e_{11}e_{21}\begin{bmatrix} e_{12} \\ e_{22} \end{bmatrix},\ e_{21}e_{11}\begin{bmatrix} e_{12} \\ e_{22} \end{bmatrix},\ \begin{bmatrix} e_{11} \\ e_{21} \end{bmatrix}\begin{bmatrix} e_{12} \\ e_{22} \end{bmatrix} \}. \quad (3)$$

Remarks:

(1) This lemma could be extended to the situation that in each parallel branch there are more events.

(2) It could be also extended to the situation that there are more parallel branches.

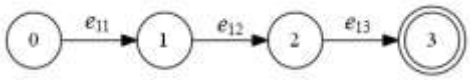
(3) From Lemma 1, the structured language will be more intuitionistic and succinct, especially when the events are more in each parallel branch.

Example: Languages generated by generators

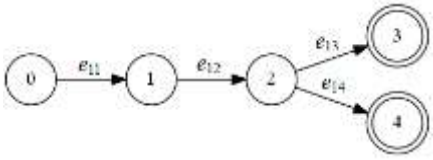
(i) $L_m(G) = \{e_{11}e_{12}e_{13}\}$, which represents a single sequence shown in Figure2(a). State 0 is the initial state and state 3 is the marked state with double circles.

(ii) $L_m(G) = \{e_{11}e_{12}e_{13}, e_{11}e_{12}e_{14}\}$, which represents languages with alternative branches, shown in Figure2(b). If system is at state 2, the happening event will be either e_{13} or e_{14}. But they couldn't happen simultaneously.

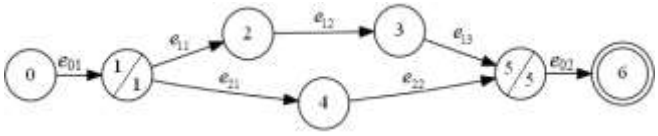
(iii) $L_m(G) = \left\{e_{01}\begin{pmatrix}e_{11}e_{12}e_{13}\\ e_{21}e_{22}\end{pmatrix}e_{02}\right\}$, which represents languages with parallel branches, shown in Figure2(c). From state 1 to state 5, there are two parallel branches: $e_{11}e_{12}e_{13}$ and $e_{21}e_{22}$. At the initial state, $v_0 = (0)$. State 1 is the beginning of the two branches. A line divides the circle to two halves, which means there will be two branches. After the happening of e_{01}, the system is at v_1 and $v_1 = (1,1)$, then the threads become two. Then two parallel branches go ahead independently. Before the last event, e_{13} or e_{22}, happens, the system is at (3, 5) or (5, 4). When the last event of e_{13} and e_{22} happens, the system is at state 5. State 5 is the end of two parallel branches. Also a line divides the circle to two halves, which means two branches end here. Only when all branches have completed could the system transit to the next state. In this example, event e_{02} must happen after events e_{13} and e_{22}.



(a) Language with a single sequence



(b) Languages with alternative branches



(c) Languages with parallel branches

Figure 2. Generators

If $L_m(G) = \left\{ \begin{pmatrix} e_{11}e_{12}e_{13} \\ e_{21}e_{22} \end{pmatrix} e_{02} \right\}$, the system will start from two initial states. If $L_m(G) = \left\{ e_{01} \begin{pmatrix} e_{11}e_{12}e_{13} \\ e_{21}e_{22} \end{pmatrix} \right\}$, the system will end at two marked states. If $L_m(G) = \left\{ \begin{pmatrix} e_{11}e_{12}e_{13} \\ e_{21}e_{22} \end{pmatrix} \right\}$, the system will operate as two separated subsystems from separate initial states to separate marked states.

In the analysis of DES, we often use a concept of the set of any possible languages, which is called Σ^*. Σ^* is a set of languages constructed by letters from $\{\varepsilon\} \cup \Sigma$. Here the languages with parallel branches are also on Σ^*.

IV. CONTROLLABILITY

a. Controllability of events

Controllability is a concept associated with the system. Whether events happen or not are decided by the system, then those events are controllable events. Otherwise, they are uncontrollable events.

For the alphabet Σ, we have the partition $\Sigma = \Sigma_c \cup \Sigma_u$. Σ_c is a subset of Σ, in which the events are controllable transitions; Σ_u is an uncontrollable event subset.

b. Controllability of a language

We follow the traditional symbols in the definition about controllability [4]. But the implication of those symbols may be extended to the multi-threads situation.

Definition 8: (Closure of a language).

For $s \in \Sigma^*$, we say $t \in \Sigma^*$ is a prefix of s, and write $t \leqslant s$, if $s = tu$ for some $u \in \Sigma^*$. If $K \subseteq \Sigma^*$, the (prefix) closure of K is the language $\overline{K}$ consisting of all prefixes of strings of K.

Example:

$L_1 = \{ e_{11}e_{12}e_{13}, e_{11}e_{12}e_{14} \}$, then $L_2 = \{ \varepsilon, e_{11}, e_{11}e_{12}, e_{11}e_{12}e_{13}, e_{11}e_{12}e_{14} \}$; $\overline{L_1} = \{ e_{01} \begin{pmatrix} e_{11}e_{12} \\ e_{22} \end{pmatrix} e_{02} \}$, then

$\overline{L_2} = \{ \varepsilon, e_{01}, e_{01} \begin{pmatrix} e_{11} \\ \varepsilon \end{pmatrix}, e_{01} \begin{pmatrix} e_{11}e_{12} \\ \varepsilon \end{pmatrix}, e_{01} \begin{pmatrix} \varepsilon \\ e_{22} \end{pmatrix}, e_{01} \begin{pmatrix} e_{11} \\ e_{22} \end{pmatrix}, e_{01} \begin{pmatrix} e_{11}e_{12} \\ e_{22} \end{pmatrix}, e_{01} \begin{pmatrix} e_{11}e_{12} \\ e_{22} \end{pmatrix} e_{02} \}$.

Remarks:

(1) L_1 represents a single thread process with two alternative sequences.

(2) L_2 includes two parallel sequences. When they are active, the number of threads of the system is two.

(3) The closure of a language is associated with the structure of the language.

Definition 9: (Controllability of a structured language)

Let K be a structured language and $K \subseteq \Sigma^*$, then K is controllable (with respect to G) if $(\forall\, t_i)\ t_i \in \overline{K}$, and let t_{i+1} be the language developed from t_i by happening of an impending uncontrollable event or some of the impending uncontrollable events simultaneously, then $t_{i+1} \in \overline{K}$.

The details of the language developing from t_i to t_{i+1} are like this:

(1)Let $t_i = \begin{pmatrix} \sigma_{i-1,1} \\ \sigma_{i-1,2} \\ \cdots \\ \sigma_{i-1,m_{i-1}} \end{pmatrix}$, $v_i = (s_{i1}, s_{i2}, \ldots, s_{im_i})^T$;

(2)Let the impending events at s_{ij} be $\Sigma_{ij} = \{\sigma_{i,j,1}, \sigma_{i,j,1}, \ldots, \sigma_{i,j,h}\} \subseteq \Sigma$, j =1,2,…, m_i and h =1,2,3,…, Σ_{ij} could be partitioned to controllable part Σ_{ij}^c and uncontrollable part Σ_{ij}^u; According to 5-tuple model, the happening event of state s_{ij} is δ_{ij} and $\delta_{ij} \in \Sigma_{ij}^c \cup \Sigma_{ij}^u \cup \{\varepsilon\}$;

(3) Let $\delta_{ij}^{ue} \in \Sigma_{ij}^u \cup \{\varepsilon\}$. In Definition 9, happening of an impending uncontrollable event means one and only one of δ_{ij}^{ue} (j =1,2,…, m_i) is not ε; happening of some of the impending uncontrollable events simultaneously means two or more events in δ_{ij}^{ue} are not ε.

(4) After the happening of δ_{ij}^{ue}, the language is changed from t_i to $t_{i+1} = \begin{pmatrix} \sigma_{i,1} \\ \sigma_{i,2} \\ \cdots \\ \sigma_{i,m_i} \end{pmatrix}$.

In other words, K is controllable if and only if no $L(G)$-string that is already a prefix of K, when followed by an uncontrollable event or some uncontrollable events in G, thereby exits from the prefixes of K: the prefix closure $\overline{K}$ is invariant under the occurrence in G of uncontrollable events.

Example:

Let $L_m(G)=\left\{\begin{pmatrix} e_{10}e_{12}, e_{10}e_{14} \\ e_{11}e_{13}, e_{11}e_{15} \end{pmatrix}\right\}$, and $\{ e_{10}, e_{12}, e_{14} \} \subseteq \Sigma_u$, $\{ e_{11}, e_{13}, e_{15} \} \subseteq \Sigma_c$, then the language $K_1=\left\{\begin{pmatrix} e_{10}e_{12} \\ e_{11}e_{13}, e_{11}e_{15} \end{pmatrix}\right\}$ is uncontrollable, for the happening of event e_{14} will make the system exit from $\overline{K_1}$.

And according to the definition, the language $K_2=\left\{\begin{pmatrix} e_{10}e_{12}, e_{10}e_{14} \\ e_{11}e_{13} \end{pmatrix}\right\}$ is controllable with respect to G.

Remarks:

(1) The abbreviated parts of languages of t_i and t_{i+1} are records of happened events.

(2) The happening of controllable events may exit from the prefixes of K, but it doesn't affect the controllability of the language K, for the system could forbid the controllable events.

(3) At a state v_i, happening of one or more controllable events companying with uncontrollable events is possible, but this doesn't interfere the controllability.

V. SUPERVISOR

a. Structure of the supervised MSMS DES

MTMS DES has been modeled as a generator. The languages generated by G is called $L(G)$. And those $L(G)$ from the initial state to the marked states construct $L_m(G)$, which represent a completely process of the system. If we want G generate the desired languages to fulfill the expected process, a supervisor is needed to permit the desired transitions and prohibit the undesired ones. Here we continue to adopt traditional supervisory frame, shown in Figure3.

Figure3. Structure of the supervisory MSMS DES

Definition 10: The marked behavior of S/G is $L_m(S/G) = L(S/G) \cap L_m(G)$; And S is nonblocking (for G) if $\overline{L_m}(S/G) = L(S/G)$.

In Figure3, S is the supervisor to supervise the generator G. It's convenient to presume that all the uncontrollable events are permitted naturally, for the system can't control the uncontrollable

events and just let them go. For the impending controllable events, the supervisor S could prohibit them to meet the need of specifications. Due to multi-threads, the supervisor S could prohibit one impending controllable event or more. The permitted impending controllable events and all impending uncontrollable events could happen simultaneously or respectively.

Definition 11: Let $L_1 \subseteq \Sigma^*$, $L_2 \subseteq \Sigma^*$, the meet of L_1 and L_2 is denoted as $L_1 \cap L_2$. Let $l_1 \in L_1$, $l_2 \in L_2$, result of $l_1 \cap l_2$ is the same sub-string from the left to right in l_1 and l_2. The operation of $L_1 \cap L_2$ is to match L_1 and L_2 to find the common substrings.

Examples:

(1) If L_1={ad, abc}, and L_2={ab, adc}, $L_1 \cap L_2$={ ad, ab};

(2) If $L_1 = \left\{ e_{01} \begin{pmatrix} e_{11} \\ e_{22} \end{pmatrix} \right\}$, $L_2 = \left\{ e_{01} \begin{pmatrix} e_{22} \\ e_{12} \end{pmatrix} \right\}$, $L_1 \cap L_2 = \left\{ e_{01} \begin{pmatrix} \varepsilon \\ e_{22} \end{pmatrix} \right\}$;

Remarks:

(1) From the example, the match of operation ‘ $\cap$ ’ includes the change of threads;

(2) If two parallel branches change their positions with each other, the language will be equivalent. In the above example (2), $L_2 = \left\{ e_{01} \begin{pmatrix} e_{22} \\ e_{12} \end{pmatrix} \right\}$ is equivalent to $\left\{ e_{01} \begin{pmatrix} e_{12} \\ e_{22} \end{pmatrix} \right\}$.

b. Optimal supervisor of MSMS DES

The supervisor will control the generator to generate a desired language, which must meet some conditions. Here we extend the traditional theory to the multi-threads situation.

Theorem 1: (Existence)

Let K be a structured language and $K \subseteq L_m(G)$, $K \neq \Phi$. There exists a nonblocking supervisory control S for G such that $L_m(S/G) = K$ if and only if

(i) K is controllable with respect to G, and

(ii) K is $L_m(G)$-closed: $K = \overline{K} \cap L_m(G)$.

Proof:

(if) We have $\overline{K} \subseteq \overline{L_m(G)} \subseteq L(G)$. Furthermore $(\varepsilon,\varepsilon,...,\varepsilon)^T \in \overline{K}$ since $K \neq \Phi$. For $(\forall\, t_i) t_i \in \overline{K}$, as we supposed before, $t_i = (\ldots \begin{matrix} \sigma_{i-1,1} \\ \sigma_{i-1,2} \\ \cdots \\ \sigma_{i-1,m_{i-1}} \end{matrix})$, $v_i = (s_{i1}, s_{i2}, \ldots, s_{im_i})^T$, the happening event of state s_{ij} is δ_{ij} and $\delta_{ij} \in \Sigma_{ij}^c \cup \Sigma_{ij}^u \cup \{\varepsilon\}$. Let t_{i+1}^1 be each possible language after t_i.

Define $S(t_i)$ according to $S(t_i) = \{ \begin{pmatrix} \delta_{i1} \\ \delta_{i2} \\ \cdots \\ \delta_{im_i} \end{pmatrix} \mid \delta_{ij} \in \Sigma_{ij}^c \cup \Sigma_{ij}^u \cup \{\varepsilon\} \mid t_{i+1}^1 \in \overline{K} \}$

We claim that $L(S/G) = \overline{K}$. First we show that $L(S/G) \subseteq \overline{K}$. Suppose $t_{i+1}^2 \in L(S/G)$, existing $t_i \in L(S/G)$, After happening of $(\delta_{i1}^1, \delta_{i2}^1, \ldots, \delta_{im_i}^1)^T \in S(t_i)$, the language t_i will be changed to t_{i+1}^2. From the definition of $S(t_i)$, t_{i+1}^1 is for each possible language, t_{i+1}^2 is one of the kinds of t_{i+1}^1, so $t_{i+1}^2 \in \overline{K}$. For the reverse inclusion, suppose $t_{i+1}^3 \in \overline{K}$. Let $t_i \in L(S/G)$, according to the definition of $S(t_i)$, $t_{i+1}^3 \in L(S/G)$. The claim is proved. Finally $L_m(S/G) = L(S/G) \cap L_m(G)$ (by definition)$= \overline{K} \cap L_m(G) = K$ (since K is $L_m(G)$-closed). And $\overline{L_m}(S/G) = L(S/G) = \overline{K}$, so S/G is nonblocking for G.

(only if) Let S be a supervisory control for G with $L_m(S/G) = K$. Assuming that S is nonblocking for G we have $L(S/G) = \overline{K}$, so $K = L(S/G) \cap L_m(G) = \overline{K} \cap L_m(G)$ i.e. K is $L_m(G)$-closed. To show that K is controllable, let $t_i^4 \in \overline{K}$, $\delta_{ij}^{ue} \in \Sigma_{ij}^u \cup \{\varepsilon\}$, events in $(\delta_{i1}^{ue}, \delta_{i2}^{ue}, \ldots, \delta_{im_i}^{ue})^T$ happen, and the language t_i^4 is changed to $t_{i+1}^4 \in L(G)$. Then $t_i^4 \in L(S/G)$ and $(\delta_{i1}^{ue}, \delta_{i2}^{ue}, \ldots, \delta_{im_i}^{ue})^T \in S(t_i)$. According to the definition of $S(t_i)$, So $t_{i+1} \in \overline{K}$, i.e. K is controllable with respect to G.

The generator G is to represent the regular process of the system. And the supervisor S is used to supervise the generator G to generate the desired languages, which will fulfill additional control goals. That control goals are represented by the specifications. Specifications in DES are languages which are also generated by a generator. The set of languages for specifications is E.

Lemma 2 :(Optimal supervisor with respect to E)

Let $E \subseteq \Sigma^*$ be $L_m(G)$-marked, and let $K = \sup C(E \cap L_m(G))$. If $K \neq \Phi$, there exists a nonblocking supervisory control (NSC) S for G such that $L_m(S/G) = K$.

Remarks:

(1)E is $L_m(G)$-marked : $E = \overline{E} \cap L_m(G)$, which means that E contains every one of its prefixes that belong to $L_m(G)$.

(2) supC($E \cap L_m(G)$) is the upper bound of $E \cap L_m(G)$, and supC($E \cap L_m(G)$) is controllable with respect to G. Then from Theorem 1, there exists a nonblocking supervisory control S for G such that $L_m(S/G) = K$.

(3) K = supC ($E \cap L_m(G)$) is the optimal result under the supervisor S with respect to E.

(4) If we compare $L_m(S/G)$ with $L_m(G)$, the function of the supervisor will be discovered. If $L_m(S/G) \subset L_m(G)$, some characters in the language are prohibited. That is to say some events are prohibited.

VI. EXAMPLES

a. Example 1: A parallel process plant

A Parallel process plant has two parallel branches, shown in Figure4. At state 0, the system will transit to state 1 and state 4 simultaneously. Then two branches will go respectively from state 1 to state 3 and from state 4 to state 6. The two parallel branches will meet together at last and the earlier arriver needs wait for the later. When the system is at state 3 and state 6, it will transit to state 0 simultaneously.

Multi-threads method is used to model the system as a generator named Plant. $Plant = (Q, \Sigma, \Delta, q_o, Q_m)$; $Q = \{0,1,2,3,4,5,6,7\}$, $q_o = \{0\}$, $Q_m = \{0\}$, $\Sigma = \{12,\ 13,\ 25,34,45,50\}$. The generator Plant is illustrated in Figure5(a). The specification here is to limit the system loops to be less than or equal to twice. So the event 14 will happen no more than twice. The third one will be prohibited by the supervisor S. The model of the specification is shown in Figure5(b). And the languages of the optimal supervised plant are $L_m(S/G)$ = supC ($L_m(G) \cap E$), shown as Figure5(c).

With the new method, parallel structure is allowed to establish the model. We can find the new method is more clear and efficient.

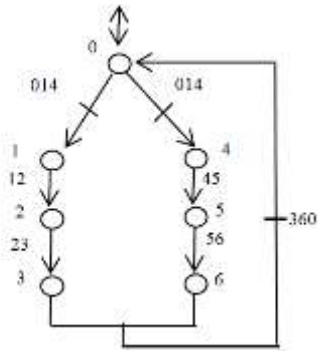
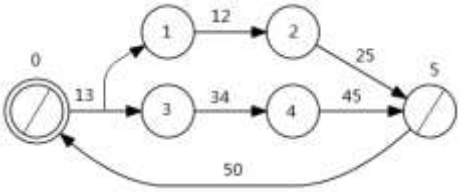
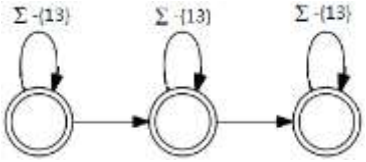


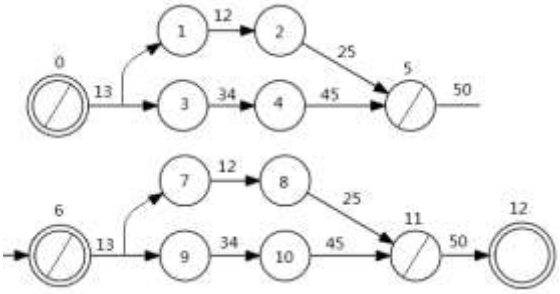
Figure 4. A parallel process plant



(a) The generator of plant with multi-threads



(b) The generator of specification



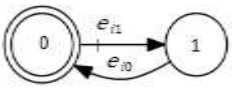
(c) The supervised generator of plant $L_m(S/G)$

Figure 5. Optimal supervisor control of plant

b. Example 2: Factories

Machines work together in a factory. This example will show some different links of machines. In this example, two machines will appear which are represented as MACHi (i could be 1 or 2), shown in Figure6(a). In this example, we use $\begin{bmatrix} e_i \\ e_j \end{bmatrix}$ to represent the situation that two events e_i and e_j happen simultaneously.

MACH1 operates to feed a buffer with capacity 1; MACH2 empties the buffer, shown in Figure6(b). After the buffer is emptied, the MACH1 could restart again.



(a) MACHi

(b) A simple factory (Two machines with a one-capacity buffer)

Figure 6. Simple factories

The factory is $L_m(G) = \left\{ \begin{pmatrix} (e_{11}e_{10})^* \\ (e_{31}e_{30})^* \end{pmatrix} \right\}$, in which $(e_{11}e_{10})^*$ means that $e_{11}e_{10}$ repeats random times. And $(e_{31}e_{30})^*$ is similar. Specification is from the buffer and E=$\left\{ \begin{pmatrix} \#\begin{bmatrix} e_{10} \\ \# \end{bmatrix}\begin{bmatrix} \varepsilon \\ e_{31} \end{bmatrix}\# \\ \# \end{pmatrix}^* \right\}$,in which # means any events in $\{\varepsilon\} \cup \Sigma$ except e_{10} and e_{31}. Then

$$L_m(S/G) = \sup C\,(L_m(G) \cap E) = \left\{ \begin{pmatrix} e_{11}e_{10}\begin{bmatrix} \varepsilon \\ e_{31} \end{bmatrix}e_{30} \end{pmatrix}^* \right\}. \qquad (4)$$

VII. EXPERIMENT

To show the supervisory theory, an experiment of the parallel process plant is introduced here. In industrial control system, signals from sensors are usually considered as inputs， which also are considered as events to drive the DES process. During the experiment, a programmable logic controller (PLC) is used as the controller. Sensors are linked to the input ports of PLC. Sensors will transfer digital signals to the controller. In PLC, variables, such as I0.0, I0.1, etc., are used to record the input information. For example, when the first port has input signal, the variable I0.0 in PLC will be “true” or “1”; when the first port has no input signal, the variable I0.0 will be “false” or “0” accordingly.

In the experiment, six sensors are used to illustrate multi-events. The control program in the controller is developed on the supervisory theory above. So the experiment can demonstrate the

validity of the supervisory theory. The control program in PLC is written in a PLC language named GRAPH, which belongs to sequential function chart (SFC).

The control program in PLC, shown in Figure7, is developed with the new supervisory theory. At state “initial”, a counter named “counter 1” is initialized to 0. Then every time the system enters “state 0”, the current value of “counter 1” will increase 1. The next states of “state 0” are “state 1” and “state 4”. The system will enter “state 1” and “state 4” simultaneously. “state 1”, “state 2” and “state 3” construct one branch of a parallel structure; “state 4”, “state5” and “state 6” construct another one. The process of each branch goes forward separately when driven by their own events. When the system is at “state 3” and “state 6”, the happening of “event 360” will make the two branches meet together and then the system transfers back to “state 0”. Then a new loop begins. Of course, the third loop will be prohibited by the supervisor, which is “counter 1”.

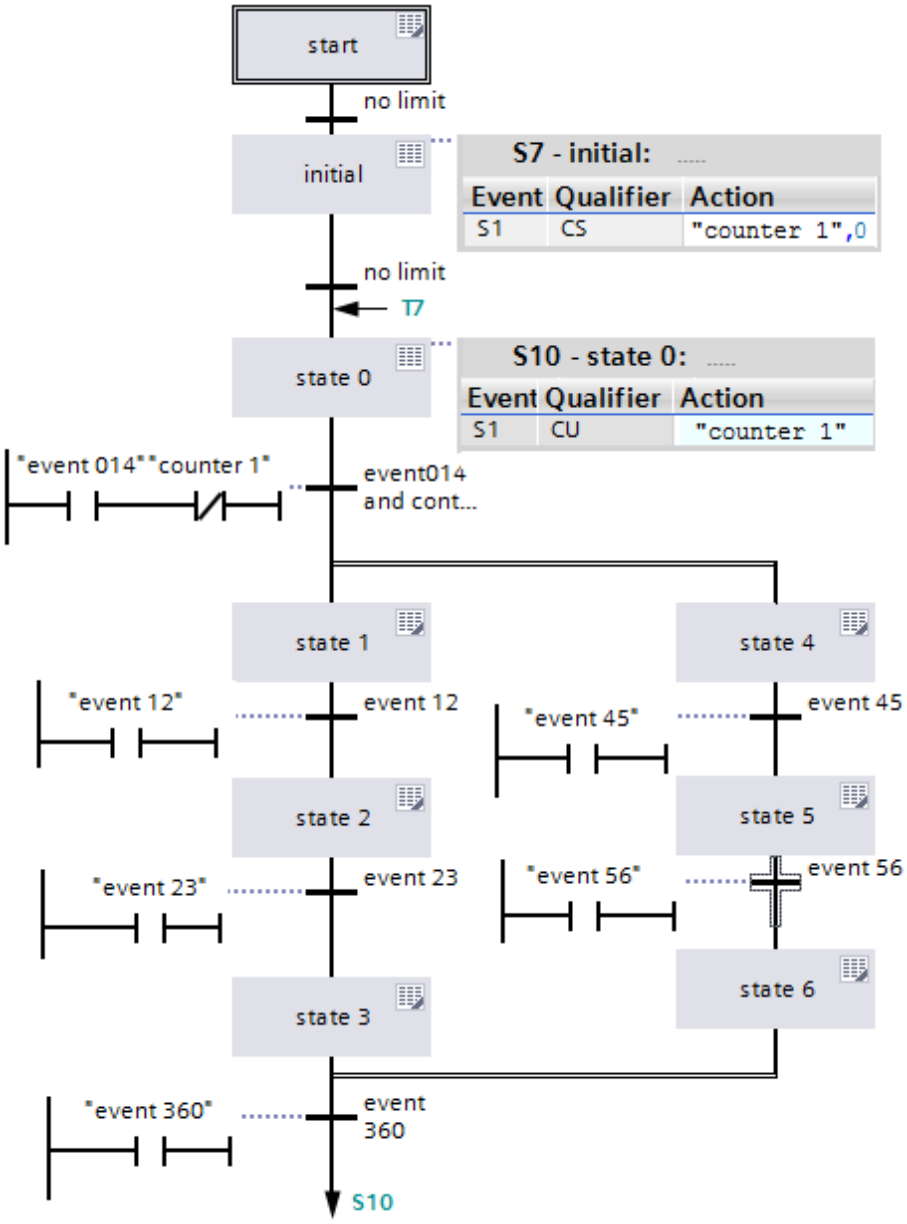


Figure 7. Control program of the parallel process plant

The control program is downloaded to a Siemens S7-300 PLC. The program runs correctly. From the experiment, the DES with parallel branches could be supervised by a structured language. And the theory could be used easily in industrial controllers.

VIII. CONCLUSIONS

The model and supervisory theory presented in this paper are extended from the traditional DES theory. In the traditional DES model, the system could be only at one state at the same time and also only one event could happen at the same time. When more events happen, it should be considered as one event. It is reasonable in theory, but it is not easy to analyze some DESs with a universal model.

When the system has more threads, which means that language has parallel structure, all the possible sequences will be enumerated out in the traditional DES model. The number of the states will be exploded. On another hand, structured languages are same as structured sequences in structure for every system. So the structured languages are natural languages of a system.

Events in different parallel branches are irrelevant and it's not convenient to mix them together and enumerate all possible single sequences. A system including the parallel sequences has multi-threads and could be dealt as a whole. In this paper, the process of a discrete-event system was analyzed. The model of a DES was established as a 5-tuple set, which is called a generator. The new generator will generate structured languages which are in accordance with the sequences of the process.

The controllability of structured languages was defined. And the theorem about the existence of the supervisor for a generator has been proved. For specifications, the method to solve the languages after supervisor was presented. The new model and supervisor theory for MSMS DES are more common than the traditional DES theory.

ACKNOWLEDGEMENTS

This work was financially supported by Natural Science Foundation of Higher Education of Jiangsu Province, China (15KJD510010).

REFERENCES

[1] Y. He, and B. Xiao. "Research of the forklift power-assisted steering system based on safety steering speed control." International Journal on Smart Sensing & Intelligent Systems, vol.8, no.1, pp.749-765, September 2015.

[2] Y. Meng, et al. "Design and implementation of intelligent integrated measuring and controlling system for sugar cane crystallization process." International Journal on Smart Sensing & Intelligent Systems, vol.8, no.3, pp.1687-1705, March 2015.

[3] Sazilah Salleh, M. F. Rahmat, and S. M. Othman. "Review on modeling and controller design of hydraulic actuator systems." International Journal on Smart Sensing & Intelligent Systems, vol.8, no.1, pp.338-367, September 2015.

[4] W. M. Wonham, Supervisory control of discrete-event systems, http://www. control . toronto. edu/ DES, Updated 2015.07.01.

[5] C. G. Cassandras and S. Lafortune, "Introduction to Discrete Event Systems." Kluwer International vol.11, no.94, pp.369–375, December 2006.

[6] R. Cieslak, C. Desclaux and A.S. Fawaz, et al, "Supervisory control of discrete-event processes with partial observations," IEEE Transactions on Automatic Control, vol.33, no.3,pp.249-260, March 2010.

[7] Abdelhakim Khatab, "The supervisory control of Timed Discrete Event Systems In the Operational Safety context." The Workshop on Discrete Event Systems, 1998.

[8] R. Y. Zhang, K. Cai and Y. Gan, et al, “Supervision localization of timed discrete-event systems”, Automatica, vol.49, no.9, pp.2786-2794, September 2013.

[9] K. Schmidt, H. Marchand, and B. Gaudin, “Modular and decentralized supervisory control of concurrent discrete event systems using reduced system models,” Discrete Event Systems, 8th International Workshop on. IEEE, pp. 149-154, 2006.

[10] R. J. Leduc, M. Lawford and P. Dai, “Hierarchical interface-based supervisory control of a flexible manufacturing system”, IEEE Trans. Control Systems Technology, vol.14, no.4, pp. 654-668, April 2006.

[11] M. Z. Fekri and S. Hashtrudi-Zad. "Hierarchical robust supervisory control of discrete-event systems," American Control Conference Westin Seattle Hotel, Seattle, Washington, USA, pp.1178-1183, June 11-13, 2008.

[12] S. Takai and T. Ushio, “Supervisory control of a class of concurrent discrete event systems”, IEICE Trans. Fundam., vol.87, no.3, pp. 850-855, March 2004.

[13] R. Su, “Supervisory control of concurrent discrete-event systems”, 50th IEEE Conference on Decision and Control and European Control Conference, pp.1811-1816, December 2011.

[14] X. Wang, Z. Li, and W. M. Wonham. "Dynamic Multiple-Period Reconfiguration of Real-Time Scheduling Based on Timed DES Supervisory Control." IEEE Transactions on Industrial Informatics, vol.12, no.1, pp.101-111, January 2016.

[15] Hrúz, M. C. Zhou. Modeling and control of discrete-event dynamic systems: With petri nets and other tools, Springer, 2007.

[16] M. dos Santos Soares, "Modeling and analysis of discrete event systems using a Petri net component. Systems", 2011 IEEE International Conf. on Man, and Cybernetics (SMC), pp.814-819, October 2011.

[17] Z. W. Li, M. C. Zhou, "Deadlock resolution in automated manufacturing systems: a novel Petri net approach", Springer, 2009.

[18] A. Jayasiri, G. K. Mann, and R. G. Gosine, "Behavior coordination of mobile robotics using supervisory control of fuzzy discrete event systems," IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 41, no. 5, pp. 1224-1238, May 2011.

[19] C. Yao and C. G. Cassandras, "Using infinitesimal perturbation analysis of stochastic flow models to recover performance sensitivity estimates of discrete event systems," Discrete Event Dynamic Systems, vol. 22, no. 2, pp. 197-219, August 2012.

[20] R. Su, J. H. van Schuppen, and J. E. Rooda, "Model abstraction of nondeterministic finite-state automata in supervisor synthesis," IEEE Trans. on Autom. Control, vol.55, no.11, pp. 2527-2541, November 2010.

[21] C. H. Chen, C. M. Kuo, and C. Y. Chen, "The design and synthesis using hierarchical robotic discrete-event modeling," Journal of Vibration and Control, vol. 19, no. 11, pp. 1603-1613, October 2013.

[22] http://en.wikipedia.org/wiki/Thread_(computer_science).