NON-BLOCKING DECENTRALIZED CONTROL OF DISCRETE EVENT SYSTEMS BASED ON PETRI NETS

Feihua Lu, Weimin Wu, Hongye Su and Jian Chu

ABSTRACT

The non-blocking property of discrete event systems can formulate many practical and important properties of manufacturing systems, such as deadlock freeness, liveness and reversibility. But it is difficult to guarantee non-blocking control. This paper presents a hybrid approach to decentralized control of discrete event systems. More generalized constraints are considered in this approach, which gives a graphical way of designing coordinators to keep the non-blocking property of the closed-loop system with decentralized supervisors. This approach also guarantees that the closed-loop system is maximally permissive.

Key Words: Decentralized control, discrete event systems (DES), hybrid approach, coordinator, non-blockingness.

I. INTRODUCTION

Due to the rapid development of computer science and its extensive application in modern society, some researchers found a new type of dynamic system called discrete event systems (DES), which is different from continuous variable dynamic systems.

The dominating characteristic of DES is that its evolution is not driven by time but by events. Essentially, DES is a manmade system. Typical DESs include manufacturing systems (especially FMS and CIMS), system schedulers, communication networks, traffic control systems, random service systems and computer operating systems.

Supervisory control theory (SCT), based on finite state machines (FSM) and formal language concepts, was introduced to extend control theory concepts for continuous systems to the discrete event environment [1–7]. In SCT, a forbidden state problem specifies conditions that must be avoided, typically the simultaneous utilization of some resource by two or more users; in addition, SCT generally requires the controlled system to be non-blocking (namely that specified target states, often just the initial state, are maintained reachable), and to be maximally permissive, i.e., to permit the occurrence of all events not leading to violation of the foregoing requirements. Although FSM provide a general framework for establishing fundamental properties of DES control problems, there are some disadvantages in using FSM [8]. Firstly, for some practical systems, the number of states used to model the system increases exponentially as the system acquires more components. This means that
FSM may be computationally inefficient. Secondly, a meaningful graphical representation is impossible for all but modest problems. Finally, one has to expend a very high initial effort to become familiar with the necessary mathematical tools.

Petri-net-based approaches, as an alternative to the original automaton framework of SCT to supervisory control design, have been considered as well [9–17]. The main reason is that the state space representation of PN as a vector addition system (whose states are integer vectors, the PN ‘markings’) can result in a compact system description, thus keeping the net structure small even though the number of possible markings may become large. PN models lend themselves not only to systematic construction of supervisory controllers, but also to the analysis of various qualitative properties and quantitative performance evaluation. But the PN models also have some disadvantages. First, in general, optimal supervisors need not exist within the class of PN [18]; and secondly, the non-blocking property (e.g., reversibility) is difficult to achieve by standard PN methods (which tend to focus on weaker requirements such as net liveness and deadlock-freeness).

As a result, some researchers have sought to explore the hybrid approaches which combine the FSM and PN [19]. Since the supervisory control of DES based on PN is a kind of decentralized control, the conflicting problems among controllers will arise. But there is no efficient approach to deal with this problem yet.

In SCT, there is an efficient way to solve this problem, which is to design a high level controller called a coordinator. The coordinator can prevent the conflict between decentralized supervisors. So, the coordinator design can be introduced into the hybrid approach to achieve non-blocking decentralized control of DES based on PN.

There are many approaches to design the coordinator. The way used in the paper is the control flow net (CFN) approach which is introduced by Feng Lei and others [20, 21]. This approach is a structural method which can provide maximally permissive and non-blocking supervisory control equivalent to the monolithic controller.

This paper extends the hybrid approach in three aspects. First, in the specification aspect, [19] just considers the forbidden state problem with the generalized mutual constraints; this paper considers a class of more general linear constraints. Then, on supervisory control aspect, this paper focuses on the decentralized supervisory control which has the blocking problem. The coordinator is introduced into PN to solve the conflicting between the decentralized supervisors. Finally, a structural approach based on control-flow net (CFN) [20, 21] is used to design the coordinator to achieve non-blocking decentralized supervisory control.

The paper is organized as follows. In Section II, the hybrid approach to supervisory control of DES based on modeling transform is introduced. In Section III, this paper shows how to transform the PN constraints into automata form. We describe our hybrid approach to design a non-blocking decentralized supervisory control in Section IV. In Section V, two illustrative examples are provided. Our conclusion is stated in the last section.

II. THE HYBRID APPROACH TO SUPERVISORY CONTROL OF DES

For the integrality and easily readability of the paper, in this section we will review the most basic concepts and notation of SCT [22, 23].

Supervisory control theory is directly based on regular languages and finite state automata. Let $\Sigma$ be a finite set of events. The empty string of length 0 is denoted as $\varepsilon$ and the set of all finite strings over $\Sigma$ including $\varepsilon$ denoted as $\Sigma^*$. For two strings $s, t \in \Sigma^*$, we write $s \leq t$ if $s$ is a prefix of $t$, namely, $t = su$ for some $u \in \Sigma^*$. Given a regular language, the language $L := \{s | s \leq t \text{ for some } t \in L\}$ is its prefix-closure.

A unique advantage of SCT is the separation of the concept of plant (open-loop dynamics) from the feedback control, so the traditional control theoretic notions, such as controllability, observability, modularity, decentralized and hierarchical control, can be exploited. In applications, the plant is modeled as an automaton (G). And the desirable behavior of the controlled system is determined by a control specification which is also modeled as an automaton (E). Those two automata may be the synchronous product of many smaller modular components.

To formalize partial observation, we partition $\Sigma$ as observable event set $\Sigma_o$ and unobservable event set $\Sigma_{uo}$, namely, $\Sigma = \Sigma_o \cup \Sigma_{uo}$. We define the natural projection $P : \Sigma^* \rightarrow \Sigma_o$ according to

$$P(\sigma) = \begin{cases} \sigma, & \sigma \in \Sigma_o \\ \varepsilon, & \sigma \notin \Sigma_o \end{cases}$$

$P(\sigma) = \varepsilon$, and $P(s\sigma) = P(s)P(\sigma)$ for all $s \in \Sigma^*$ and $\sigma \in \Sigma$.

The effect of $P$ on a string $s \in \Sigma^*$ is just to erase the events in $s$ that are unobservable and retain the observable ones in the previous order.

A group of supervisory controllers (supervisors) closes the loop of a controlled DES and forces the plant
to respect the control specifications. The controllers act only to disable certain events that are originally able to occur in the plant, thus preventing them from occurring. The control logic of a supervisory controller is derived from the event disablement list at each state. In practice, we may assume that some events in the alphabet can never, or need not be disabled. Such events are called uncontrollable, while those preventable by a supervisory controller are called controllable. Hence, the alphabet $\Sigma$ is partitioned into two disjoint subsets of controllable events ($\Sigma_c$) and uncontrollable events ($\Sigma_{uc}$), such that $\Sigma = \Sigma_c \cup \Sigma_{uc}$.

To synthesize a satisfactory supervisor, SCT provides a formal method for theoretically tackling the typical supervisory control problem:

Given a plant $G$ over alphabet $\Sigma$ with its partition $\Sigma = \Sigma_c \cup \Sigma_{uc}$ and control specifications modeled as $E$, finds a maximally permissive supervisor $S$ such that the controlled system $S/G$ is non-blocking and always meets the control specifications.

The hybrid approach to supervisory control of DES coupling Ramadge–Wonham (RW) supervisors to Petri nets has been presented in [19]. This approach applies the SCT into Petri nets, and use the software TCT [24].

This approach first assumes that there is an uncontrollable PN model (UPNM) of a DES, which is bounded and has initial marking, and a set of forbidden state specifications. Then UPNM is reduced, if possible, by PN reduction rules. On the next step, convert the reduced UPNM and the specifications into equivalent buffer models. When all the preparatory work is done, supervisory control theory (SCT) can be applied to obtain an RW supervisor (SUPER) and its control data (SUPDAT), and also obtain the simplified supervisor (SIMSUP) and the simplified control data (SIMDAT) from the RW-supervisor together with its control data by using a supervisor reduction method. Next, we convert SIMSUP into an auto-net representation, say AUTONET. Finally, we obtain the closed-loop (or controlled) hybrid model, say CHM, by coupling AUTONET to UPNM using inhibitor arcs according to SIMDAT.

III. TRANSFORMATION OF CONSTRAINTS

In the hybrid approach [19], the generalized mutual exclusion constraints are discussed.

**Definition 1.** The generalized mutual exclusion constraint (GMEC) is a kind of linear PN constraint which has the form:

$$L\mu \leq b$$  \hspace{1cm} (1)

where $\mu$ is the system’s marking, $|L| \in \mathbb{Z}^{n_c \times m}$, $b \in \mathbb{Z}^{n_c}$, $m$ is the number of places of the PN, and $n_c$ is the number of constraints.

In this paper, a class of more general linear constraints which is called extended generalized mutual exclusion constraint is considered.

**Definition 2.** The extended generalized mutual exclusion constraint (EGMEC) is a kind of linear PN constraint which has the form:

$$L\mu + Hq + Cv \leq b$$  \hspace{1cm} (2)

where $q$ is the firing vector and $v$ is the Parikh vector, $H \in \mathbb{N}^{n_c \times n}$, $C \in \mathbb{Z}^{n_c \times n}$.

In such constraints, an element $q_i$ of the firing vector $q$ is set to be 1 if the transition $t_i$ is to be fired next from $\mu$; else $q_i = 0$. Alternatively, if multiple firings are allowed at the same time, the element $q_i$ of the firing vector $q$ represents how many times the transition $t_i$ is fired at the next firing instance. The element $v_i$ of the Parikh vector $v$ counts how many times the transition $t_i$ has fired. The firing vector is the determining constraint of system, which does not change the state of system itself. The firing vector has two states: $q = 0$, which means the event $q$ does not happen at this moment; $q = 1$, which means the event $q$ will happen at this moment. So the event $q$ could be prevented only if and only if the constraints $Hq + Cv \leq b$ is held whenever $q = 0$ or $q = 1$.

A supervisor enforcing (2) ensures that (i) all states of $(\mu, v)$ satisfy $L\mu + Cv \leq b$; (ii) if $\mu \xrightarrow{t_i} \mu'$ and $v' = v + q$, then $L\mu + Hq + Cv \leq b$ and $L\mu' + C(v') \leq b$.

In [25], it is shown that the class of constraints

$$Hq + Cv \leq b$$  \hspace{1cm} (3)

is as general as the class (2). That means, given any set of constraints $L\mu + Hq + Cv \leq b$, there is $C'$ such that $L\mu + Hq + Cv \leq b$ and $Hq + C'v \leq b$ are equivalent.

It is simple to find the equivalent form, since each marking of places can be represented as the difference of its input and output transitions’ Parikh vector.

The constraints $Hq + Cv \leq b$ are satisfiable when no transition is firing ($q = 0$) as well as when a transition is fired ($q \neq 0$). In this way, these constraints can be easily transformed into buffer specifications in automata form.

The equivalent specifications have $(b + 1)$ states which means that the buffer has capacity $b$. Every $v_i$
will load items into the buffer if $c_i$ is positive, else it will unload the items from the buffer. The event $q$ can happen when the buffer is not overflowed.

Thus, supervisor enforcing those specifications ensures that: (i) the buffer is never overflowed or underflowed; (ii) the event could happen only when both the states happened before and after it satisfy (i).

We will show the detail by the example “Shared Area” in Section V.

IV. NON-BLOCKING CONTROL

In [26], the authors address the decentralized control problem, but without regard to blocking. However, one of the main problems in decentralized supervisory control of DES with PN is how to design the non-blocking decentralized control. Many methods have been developed to find suitable coordinators to solve the blocking. In this paper, the control flow net (CFN) method [20, 21] is used. A brief introduction to CFN can be found in the Appendix of this paper.

It is a graphical approach to the decomposition and simplification of DES using CFN. A CFN allows a partition of the system into parts that are respectively “harmless” and “problematic” from the viewpoint of blocking (of which deadlock, for instance, is a special case). The harmless part is irrelevant to blocking and can be removed from the net, leaving only the problematic part to be verified and coordinated. Then this approach will detect whether a class of systems is blocking, and obtain control policies for optimal non-blocking control. The approach is computationally efficient, as it is based only on system interconnection relationships and is insensitive to various numerical parameters like buffer capacities. Moreover, the resulting control logic is transparent in that it takes the form of interpretable “if...then...” rules rather than being left implicit in the structure of supervisory control automata. So it will be easy to transform to PN form.

Our design approach can be summarized as follows:

1. Assume that there is an uncontrolled PN model (UPNM) of a DES, with initial marking, and a set of specifications.
2. Reduce UPNM, if possible, by the PN reduction rule, and then make sure that UPNM is bounded and determine explicit bounds on the place markings (By software PNetLab).
3. Convert the reduced UPNM and the specifications into equivalent buffer models, in the framework of FSM.
4. Apply supervisory control theory (SCT) to obtain the Ramadge–Wonham (RW) supervisor (SUPER) and its control data (SUPDAT) for each single specification.
5. Use CFN to find out whether the specification may cause blocking or not. If it would, go to step 6, else to step 7.
6. Achieve the specifications to prevent the blocking problem by using CFN, and then use the coordinators to realize the specifications.
7. Use the hybrid approach to get the hybrid supervisor.

Sometimes we can simplify steps 1-3 if we have the original system, because we can get the PN and FSM models directly from the original system.

To check if the decentralized supervisors will cause blocking or not, we can use two methods. One is to check if the combination of those supervisors is as same as the monolithic supervisor. The other way is the graphical method, which detects if the decentralized supervisors will create the structure of a cycle which is called the coupling module (CM). The decentralized supervisors will cause blocking when they cannot prevent the CM from being full, and then, the coordinators are designed to prevent the CM from being full.

If the system is small, the first way is easy to apply. However, when the system is quite large, the latter methods will be better.

We will explain the steps in detail by the following example in the next section.

V. EXAMPLES

5.1 Shared area

The original problem is presented in [27] Section 4.2. This example corresponds to a region of a factory cell in which autonomous vehicles (AVs) access a shared area (SA). The numbers of AVs that can enter the SA at the same time is limited. (In this case, it is supposed to be 2). The AVs enter the SA from two directions: left and right; AVs coming from the left side enter SA via t4 and t13, and the right side via t5 and t14. The AVs exit the restricted area via t9 and t10. (Fig. 1).

In the original problem [27], there are four constraints, which are called arbitration constraints. The Petri net forms of the constraints are as follows:

\[ 2\mu_5 + \mu_2 + \mu_7 \leq 2-(v_{13}+v_{14}+v_{4}+v_{5}-v_{9}-v_{10})+1 \quad (4) \]
In those constraints, there are three kinds of vectors, $q$ is the firing vector, $v$ is the Parikh vector, and $\mu$ is the place marking. We convert those constrains $L\mu + Hq + Cv \leq b$ into the form $Hq + Cv \leq b$.

Take the constraint (4), for example, the state marking $q_2$ is equal to the differences of its input and output transitions’ Parikh vector $q_2 = q_7 - q_3 - q_4$. Similarly, we have $q_3 = q_7 - q_3 - q_13$, $q_6 = q_2 - q_14 - q_5 - q_9 - q_10$. By replacing all the state markings with their equivalent Parikh vectors, all the constraints are as follows:

$$2q_4 + \mu_3 + \mu_8 \leq 2 - (v_{13} + v_{14} + v_5 - v_9 - v_{10}) + 1$$

$$2q_3 \leq \mu_3 + \mu_8 + (v_{13} + v_{14} + v_4 + v_5 - v_9 - v_{10})$$

$$2q_6 \leq \mu_2 + \mu_7 + (v_{13} + v_{14} + v_4 + v_5 - v_9 - v_{10})$$

Now, we begin to transform the PN constraints into automata specifications.

First we code the transitions as follows:

$$t1:1 \quad t2:2 \quad t3:3 \quad t4:41 \quad t5:5 \quad t6:61 \quad t7:7 \quad t8:81 \quad t9:8 \quad t10:10 \quad t13:13 \quad t14:11$$

Take the constraint (8) for example. First we ignore the $q_5$, and it is easy to transform the constraint $v_2 + v_{14} + v_5 - v_9 - v_{10} \leq 3$. We construct a buffer, say $SPEC_1$, with capacity 3, which is incremented by 1 under events $t2$, $t5$, $t14$ and decremented by 1 under events $t9$, $t10$. The equivalent specification is as follows.

![SPEC1](image)

Then, we take the $q_5$ into our consideration. We find that event $t5$ is to be enabled only when $SPEC_1$ is in state 0 or 1, we delete the transition $[2 \ t5 \ 3]$ to obtain the intended specification.

![SPEC2](image)

Similarly, we can get the FSM model of constraint (6).

![SPEC3](image)

But for constraints (7) (8), it is a little different. Since events $t3$ and $t6$ don’t change the buffers’ capacity, they will add into the specifications as self-loops in certain states.
The make-it files can be found in the Appendix.

In the last line of the make-it file, we can find the Isomorph(SUPER, TEST) is true, which means that the decentralized control has the same effect as the centralized control.

Using CFN to analyze this system, we find that there is no cycle (CM) in this system. Consequently, the decentralized control will keep the non-blockingness.

5.2 A manufacturing system

Now we consider a more complicated example which has the blocking problem. This example has been first discussed in [26]. Iordache et al. applied his approach in this example [28, 29]. In [28] we can find that the close-loop system did not keep the non-blockingness. In [29], with more strict constraints, the decentralized control was non-blocking, but no longer maximally permissive. Using our approach, we can achieve a non-blocking and maximally permissive decentralized control. We check the above approaches by software PNetLab [30].

The example is a manufacturing system consisting of two machines (M1, M2), four material handling devices (H1,H2, H3, H4), and six buffers (B1, B2, B3, B4, B5, B6). The connection is shown in Fig. 2.

Since the transport between buffers and devices is uncontrollable, we can simplify the system as Fig. 3, and it is also the CFN model of the system.

We can get the system’s PN model and FMS model, since we already have the system’s description. For the convenience of analysis, we give the FSM model and supervisor design first.

The system produces two types of parts. Type1 parts require three operations: operation T1P1 to be performed on machine M1, T1P2 on M2, and T1P3 on M1. Type2 parts also require three operations: T2P1 performed on M2, T2P2 on M1, and T2P3 on M2. For Type1 parts, the complete event sequence is $\gamma_1, t_1, \alpha_3, t_2, \alpha_4, \eta_1$; and for Type2 parts, the complete event sequence is $\gamma_2, t_4, \alpha_1, t_2, \alpha_4, \eta_2$.

The FSM model of M1 is shown in Fig. 4, the one of M2 is similar.

For easy analysis, we assume that the capacity of B1-B4 is one, and the capacity of B5 and B6 is infinite.

The FSM model of B1 is shown in Fig. 5, the other buffers are similar.

We consider the M1 and M2 as plant, and the B1, B2, B3, and B4 as the specifications. All the events are controllable. Using TCT as our analysis tool, we can get the results as shown in Section III of the appendix of this paper.

At the end of Section 3.4 of the Appendix, we can find that the TCT command “Nonconflict (PLANT, SIMPLANT)” is false, which means that the decentralized controllers are conflicting with the plant. Appropriate coordinators are needed to solve this problem. Notice that this system has four loops (Fig. 3), the CMs in CFN: M1 B1 M2 B3; M1 B1 M2 M3; M2 B2 M1 B4; M2 B2 M1 B5.

We consider the M1 and M2 as plant, and the B1, B2, B3, and B4 as the specifications. All the events are controllable. Using TCT as our analysis tool, we can get the results as shown in Section III of the appendix of this paper.

At the end of Section 3.4 of the Appendix, we can find that the TCT command “Nonconflict (PLANT, SIMPLANT)” is false, which means that the decentralized controllers are conflicting with the plant. Appropriate coordinators are needed to solve this problem. Notice that this system has four loops (Fig. 3), the CMs in CFN: M1 B1 M2 B3; M1 B1 M2 M3; M2 B2 M1 B4; M2 B2 M1 B5.
Fig. 6. The PN model.

Fig. 7. The PN model with supervisors and coordinators.
B4; M1 B2 M2 B3; M1 B2 M2 B4. Then we have to prevent the CMs from being full, which means that each loop’s work pieces’ number should be limited; otherwise, the system would be blocking. In this way, four coordinators are created.

In fact, each coordinator has its own physical meaning. Take the coordinator2, which represents the CM M1 B1 M2 B3, for example. This CM represents the working sequence of Part1. The input of this loop is γ1, and output is γ2. The capacity of this CM is 4. So we need to design a coordinator to make sure that the difference of input and output should be less than 4. Otherwise, the CM is full, and the system will be blocked.

The coordinator3 (for the CM: M1 B2 M2 B4) is similar to coordinator2, and represents the working sequence of Part2.

The coordinator1 represents the interaction of two working sequences. This CM contains M1 B1 M2 B4. When M1 has a work piece from B5, B1 has one from M1, M2 has one from B6, and B4 has one from M2, the CM will be full. And each unit is waiting for the next one (M1-B1-M2-B4-M1). And the system will be blocked. The inputs of this CM are γ1 and γ2, and the outputs are t2 and t3. The capacity of this CM is 4. So we need to design the coordinator1 to make sure that the difference of input and output should be less than 4.

The coordinator4 is similar to the coordinator1. The progress of achieving the four coordinators is as shown in the Appendix.

In the end of Section 3.5 of the Appendix, we can find that the TCT command “Isomorph(SUPER, TEST; identity)” is true, which means it is successful to deal with the conflicting problem, and achieve a non-blocking close-loop system finally.

VI. CONCLUSION

A hybrid approach to deal with the decentralized control of DES is shown in the paper. This paper considers more general constraints, and focuses on the non-blockingness of decentralized control. We introduce the coordinator into a Petri net, which is an efficient tool to solve the blocking problem between the decentralized supervisors. The coordinator is a high level controller to prevent the conflicting between low level controllers (the decentralized supervisors). As a result, designing the proper coordinator is the key work for achieving non-blocking decentralized supervisory control of DES.

The method we used is a structural method which can provide maximally permissive and non-blocking supervisory control equivalent to the monolithic controller.

The observability is not considered in this paper, and will be the subject of further research.

APPENDIX

AI. THE INTRODUCTION OF CONTROL FLOW NET (CFN)

CFN captures the interaction among machines of a shuffle plant and buffer and/or server specifications. This introduction is based on [20, 21].

A shuffle plant is a regular language:

\[ G := \prod_{i=1}^{m} G_i \]

where m is the number of plant components and \( G_i \) is the regular language of the ith component. A string \( s \in G_i \) is called a task if \( s \neq \epsilon \) and \( (\forall r: \epsilon < r < s) r \neq G_i \). Let \( G[^k] \) be the set of tasks of the component.

To describe controllability, we define the function

\[ \text{skidext}_i: \tilde{G} \rightarrow 2^\Sigma^*: s \mapsto \{ u \in \Sigma^* | su \in \tilde{G}_i \} \]

where \( \tilde{G}_i \) is the prefix closure of the language \( G_i \). For a string \( s \in G_i \), \( \text{skidext}_i(s) \) returns all the uncontrollable event strings that extend \( s \).

To formalize the buffer specifications, we bring in the following notation. For a string \( s \in \Sigma^* \) and \( \sigma \in \Sigma \), \( \#\sigma(s) \) denotes the number of instances of \( \sigma \) appearing in \( s \). The notation is extended to subsets \( T \subseteq \Sigma \) according to \( \#T(s) := \sum_{\sigma \in T} \#\sigma(s) \). For a language \( L \subseteq \Sigma^* \), \( \#T(L) := \{ \#T(s) | s \in L \} \).

Consider a buffer resource that stores \( d \) (\( d \geq 1 \)) types of items. We call \( d \) the diversity of the buffer and let set \( d := \{1, \ldots, d\} \). Let \( \text{Load}_i \), \( \text{Unld}_i \) (\( i \in d \)) denote the event sets that load and unload the buffer, respectively. To record the number of type \( i \) (\( i \in d \)) items in buffer, we define the function

\[ \text{quant}_i: \Sigma^* \rightarrow \mathbb{Z}: s \mapsto \#\text{Load}_i(s) - \#\text{Unld}_i(s) \]

Let \( \text{Load} := \bigcup_{i=1}^{d} \text{Load}_i \) and \( \text{Unld} := \bigcup_{i=1}^{d} \text{Unld}_i \). The total number of items in the buffer is then returned by the function

\[ \text{quant}: \Sigma^* \rightarrow \mathbb{Z}: s \mapsto \#\text{Load}(s) - \#\text{Unld}(s) \]

Thus \( \text{quant}(s) = \sum_{i=1}^{d} \text{quant}_i(s) \). We identify the types of the items in the buffer by the function

\[ \text{type}: \Sigma^* \rightarrow 2^d: s \mapsto \{ i \in d | \text{quant}_i(s) > 0 \} \]
Now, we can give our basic definitions as follows.

**Definition A1.** A control-flow net (CFN) is a bipartite directed graph (digraph) \( N = (M, B, \Psi_1, \Psi_o) \) where

- \( M \) is a set of nodes representing machines that compose a shuffle plant;
- \( B \) is a set of nodes representing buffers;
- \( \Psi_1 \subseteq M \times \Sigma \times B \) is a set of edges directed from machines \( G_i \in M \) to buffers \( B \in B \). Edge \((G_i, \sigma, B) \in \Psi_1\), if and only if \( \sigma \in \text{Load} \cap \Sigma_i \), i.e., machine \( G_i \) loads an item into the buffer via event \( \sigma \);
- \( \Psi_o \subseteq B \times \Sigma \times M \) is a set of edges directed from buffers \( B \in B \) to machines \( G_i \in M \). Edge \((B, \sigma, G_i) \in \Psi_o\), if and only if \( \sigma \in \text{Unload} \cap \Sigma_i \), i.e., machine \( G_i \) unloads an item from the buffer via event \( \sigma \).

Graphically, we represent a machine in \( M \) by a square and a buffer in \( B \) by an oval.

Define the following three projection functions on the edges of the CFN \( N \).

- **exit:** \( \Psi_1 \cup \Psi_o \rightarrow M \cup B : (v_1, \sigma, v_2) \mapsto v_1 \)
- **event:** \( \Psi_1 \cup \Psi_o \rightarrow \Sigma : (v_1, \sigma, v_2) \mapsto \sigma \)
- **exit:** \( \Psi_1 \cup \Psi_o \rightarrow M \cup B : (v_1, \sigma, v_2) \mapsto v_2 \)

At any vertex \( v \in M \cup B \), the incoming and outgoing edges of \( N \) are given by the functions

\[
in_N : M \cup B \rightarrow 2^{\Psi_1 \cup \Psi_o} : v \mapsto \{ \psi \in \Psi_1 \cup \Psi_o | \text{entr}(\psi) = v \}
\]

\[
out_N : M \cup B \rightarrow 2^{\Psi_1 \cup \Psi_o} : v \mapsto \{ \psi \in \Psi_1 \cup \Psi_o | \text{exit}(\psi) = v \}
\]

Let \( B \in B \) be a buffer with capacity \( C_B \) and diversity \( d_B \). Which type of item an edge loads or unloads at \( B \) is decided by the function

\[
edgetp : \Psi_1 \cup \Psi_o \rightarrow \mathbb{N}
\]

**Definition A2.** Consider two event subsets \( X, Y \subseteq \Sigma \) of a machine \( G_i \subseteq \Sigma^*_i \). We say \( X \) does not inhibit \( Y \) if for all \( s \in G_i^{T^k} \).

\[
\# X \circ \text{skidext}_i(s) = 0 \Rightarrow (\exists t \in \Sigma^*_i) s t \in G_i, \# Y(st) > 0,
\]

\[
\# X(t) = 0, \# X \circ \text{skidext}_i(s t) = 0
\]

where \( s t := \{ w | s \leq w \leq s t \} \); otherwise, \( X \) inhibits \( Y \) at some string \( s \) that violates the above relation.

To identify the buffers that influence the occurrences of a set of events, we define the function

\[
inht : \bigcup_{G_i \in M} 2^{\text{in}(G_i)} \rightarrow 2^{\Psi_1}
\]

**Definition A3.** An inhibition flow subnet of \( N = (M, B, \Psi_1, \Psi_o) \) is a subnet \( C = (N, T, \Phi_1, \Phi_o) \) that satisfies

1. \( T \neq \emptyset \)
2. For all machines \( G_i \in N \), \( \text{in}_C(G_i) \neq \emptyset \), \( \text{out}_C(G_i) \neq \emptyset \) and \( \text{inht} \circ \text{in}_C(G_i) \neq \emptyset \)
3. Every buffer \( B \in T - \{ \text{Sink}, \text{Source} \} \) capacity \( C_B \) and diversity \( d_B \) contains \( k \) distinct types of items where \( 1 \leq k \leq \min(C_B, d_B) \). These items in the buffer determine the outgoing edges.

**Definition A4** (Cache). A machine \( G_i \in N \) is a cache of subsystem \( C \) at string \( s \in G_i \) if there exists a string \( t \in \Sigma^*_i \) such that \( s t \in G_i^{T^k} \), \# \( Y_i(t) = 1 \), \# \( X_i(t) = 0 \) and \( \# X_i \circ \text{skidext}_i(s t) = \emptyset \).

**Definition A5** (Coupling Module). A coupling module (CM) of a CFN \( N = (M, B, \Psi_1, \Psi_o) \) is a strong inhibition-flow subnet \( C = (N, T, \Phi_1, \Phi_o) \)

**Definition A6.** Let \( G \) describe a shuffle plant. We say a CM \( C = (N, T, \Phi_1, \Phi_o) \) is full at string \( s \in G \), if \((\forall B \in T) \text{quant}_B(s) = C_B \), \( \text{type}_B(s) = \text{edgetp} \circ \text{out}_C(B) \), and for all \( G_i \in N \), \( G_i \) is not a cache of \( C \) at string \( P_i(s) \).

If the CFN \( N = (M, B, \Psi_1, \Psi_o) \) does not contain any CM, then the system \( N \) is deadlock-free.

If any CM in CFN \( N = (M, B, \Psi_1, \Psi_o) \) is full, then the system is blocking.

**Example.** Here is one of the most simple examples for the CFN (Fig. A1). This manufacturing system consists of two parts: the AGV and the machine. The AGV delivers the work piece from the outside buffer to the machine. When the machine finishes the operation, and then the AGV delivers the finished piece from the machine to the outside buffer.

This system has one CM, which is composed by the AGV and the machine. Suppose that both the AGV and the machine’s capacity are 1, and then we know that the capacity of the CM is 2. So when the piece in the CM should be less than 2, otherwise the system would
be blocking. Suppose that the CM now has 2 pieces, which means that the AGV has a working piece and the machine has a piece in operation. Then the system is blocking, since the AGV wants to put the working into the machine but the machine is full, and the machine want to put the finished piece into AGV but the AGV is full. The system is in a waiting-cycle now, which means blocking.

**AII. THE MAKE-IT FILE (THE TCT CODE) OF “THE SHARED AREA”**

**A2.1 Create each place, and combine those together to get the plant**

\[
P_2 = \text{Create}(P2, [\text{mark} 0], [\text{tran } [0,21,1],[1,3,0],[1,41,0]]) \quad (2,3)
\]

\[
P_3 = \text{Convert}(P2, [[3,61],[21,7],[41,5]]) \quad (2,3)
\]

\[
P_7 = \text{Create}(P7, [\text{mark} 0], [\text{tran } [0,3,1],[1,13,0]]) \quad (2,2)
\]

\[
P_8 = \text{Convert}(P7, [[3,61],[13,11]]) \quad (2,2)
\]

\[
S_A = \text{Create}(S_A, [\text{mark} 0], [\text{tran } [0,5,1],[0,11,1],[0,13,1],[0,41,1],[1,5,2],[1,8,0],[1,10,0],[1,11,2],[1,13,2],[1,41,2],[2,5,3],[2,8,1],[2,10,1],[2,11,3],[2,13,3],[2,41,3],[3,8,2],[3,10,2]]) \quad (4,18)
\]

\[
\text{PLANT} = \text{Sync}(P2, P7) \quad (4,7) \quad \text{Blocked events} = \text{None}
\]

\[
\text{PLANT} = \text{Sync}(\text{PLANT}, P3) \quad (8,26) \quad \text{Blocked events} = \text{None}
\]

\[
\text{PLANT} = \text{Sync}(\text{PLANT}, P8) \quad (16,56) \quad \text{Blocked events} = \text{None}
\]

\[
\text{PLANT} = \text{Sync}(S_A, \text{PLANT}) \quad (64,288) \quad \text{Blocked events} = \text{None} \land \text{ALL} = \text{Allevents(PLANT)} \quad (1,10)
\]

**A2.2 Create each specification, and combine those together to get the final specification**

\[
\text{SPEC1} = \text{Create}(\text{SPEC1}, [\text{mark} 0], [\text{tran } [0,21,1],[1,3,0],[1,41,0]]) \quad (2,3)
\]

\[
\text{SPEC3} = \text{Create}(\text{SPEC3}, [\text{mark} 0], [\text{tran } [0,5,1],[0,11,1],[0,13,1],[0,41,1],[1,5,2],[1,8,0],[1,10,0],[1,11,2],[1,13,2],[1,41,2],[2,5,3],[2,8,1],[2,10,1],[2,11,3],[2,13,3],[2,41,3],[3,8,2],[3,10,2]]) \quad (4,18)
\]

\[
\text{SPEC5} = \text{Create}(\text{SPEC5}, [\text{mark} 0], [\text{tran } [0,5,1],[0,11,1],[0,13,1],[0,41,1],[1,5,2],[1,8,0],[1,10,0],[1,11,2],[1,13,2],[1,41,2],[2,5,3],[2,8,1],[2,10,1],[2,11,3],[2,13,3],[2,41,3],[3,8,2],[3,10,2]]) \quad (4,18)
\]

\[
\text{SPEC6} = \text{Convert}(\text{SPEC6}, [[61,31],[11,13],[21,71]]) \quad (4,16)
\]

\[
\text{SP1} = \text{Sync}(\text{SPEC1}, \text{ALL}) \quad (2,17) \quad \text{Blocked events} = \text{None}
\]

\[
\text{SP2} = \text{Sync}(\text{SPEC6}, \text{ALL}) \quad (2,17) \quad \text{Blocked events} = \text{None}
\]

\[
\text{SP3} = \text{Sync}(\text{SPEC5}, \text{ALL}) \quad (3,24) \quad \text{Blocked events} = \text{None}
\]

\[
\text{SP4} = \text{Sync}(\text{SPEC4}, \text{ALL}) \quad (5,48) \quad \text{Blocked events} = \text{None}
\]

\[
\text{SP5} = \text{Sync}(\text{SPEC5}, \text{ALL}) \quad (4,32) \quad \text{Blocked events} = \text{None}
\]

\[
\text{SP6} = \text{Sync}(\text{SPEC6}, \text{ALL}) \quad (4,32) \quad \text{Blocked events} = \text{None}
\]

\[
\text{SP} = \text{Meet}(\text{SP1}, \text{SP2}) \quad (4,28)
\]

\[
\text{SP} = \text{Meet}(\text{SP3}, \text{SP}) \quad (12,68)
\]

\[
\text{SP} = \text{Meet}(\text{SP4}, \text{SP}) \quad (60,316)
\]

\[
\text{SP} = \text{Meet}(\text{SP5}, \text{SP}) \quad (60,287)
\]

\[
\text{SP} = \text{Meet}(\text{SP6}, \text{SP}) \quad (60,258)
\]

**A2.3 Get the monolithic control of PLANT by taking SP as the specification**

\[
\text{SUPER} = \text{Supcon}(\text{PLANT}, \text{SP}) \quad (135,410)
\]

\[
\text{SUPER} = \text{Condat}(\text{PLANT}, \text{SUPER}) \land \text{Controllable.}
\]

\[
\text{SIMSUP} = \text{Supreduce}(\text{PLANT}, \text{SUPER}, \text{SUPER}) \quad (90,352; \text{slb}=60)
\]

Then, we try the decentralized control:

**A2.4 Get each single specification’s supervisory control**

\[
\text{UP1} = \text{Supcon}(\text{PLANT}, \text{SP1}) \quad (48,200)
\]

\[
\text{UP1} = \text{Condat}(\text{PLANT}, \text{SUP1}) \land \text{Controllable.}
\]

\[
\text{SIMSUP1} = \text{Supreduce}(\text{PLANT}, \text{SUP1}, \text{SUP1}) \quad (2,16; \text{slb}=2)
\]

\[
\text{SUP2} = \text{Supcon}(\text{PLANT}, \text{SP2}) \quad (48,200)
\]

\[
\text{SUP2} = \text{Condat}(\text{PLANT}, \text{SUP2}) \land \text{Controllable.}
\]

\[
\text{SIMSUP2} = \text{Supreduce}(\text{PLANT}, \text{SUP2}, \text{SUP2}) \quad (2,16; \text{slb}=2)
\]

\[
\text{SUP3} = \text{Supcon}(\text{PLANT}, \text{SP3}) \quad (48,200)
\]

\[
\text{SUP3} = \text{Condat}(\text{PLANT}, \text{SUP3}) \land \text{Controllable.}
\]

\[
\text{SIMSUP3} = \text{Supreduce}(\text{PLANT}, \text{SUP3}, \text{SUP3}) \quad (3,24; \text{slb}=3)
\]

\[
\text{SUP4} = \text{Supcon}(\text{PLANT}, \text{SP4}) \quad (320,1408)
\]

\[
\text{SUP4} = \text{Condat}(\text{PLANT}, \text{SUP4}) \land \text{Controllable.}
\]

\[
\text{SIMSUP4} = \text{Supreduce}(\text{PLANT}, \text{SUP4}, \text{SUP4}) \quad (5,48; \text{slb}=5)
\]

\[
\text{SUP5} = \text{Supcon}(\text{PLANT}, \text{SP5}) \quad (48,192)
\]

\[
\text{SUP5} = \text{Condat}(\text{PLANT}, \text{SUP5}) \land \text{Controllable.}
\]

\[
\text{SIMSUP5} = \text{Supreduce}(\text{PLANT}, \text{SUP5}, \text{SUP5}) \quad (4,29; \text{slb}=4)
\]

\[
\text{SUP6} = \text{Supcon}(\text{PLANT}, \text{SP5}) \quad (48,192)
\]

\[
\text{SUP6} = \text{Condat}(\text{PLANT}, \text{SUP5}) \land \text{Controllable.}
\]

\[
\text{SIMSUP6} = \text{Supreduce}(\text{PLANT}, \text{SUP5}, \text{SUP5}) \quad (4,29; \text{slb}=4)
\]

**A2.5 Test the decentralized control is the same effect as the monolithic control**

\[
\text{MODSUP} = \text{Meet}(\text{SIMSUP1}, \text{SIMSUP2}) \quad (4,24)
\]

\[
\text{MODSUP} = \text{Meet}(\text{MODSUP}, \text{SIMSUP3}) \quad (12,56)
\]
A3.2 Create the specification

B1 = Sync(B1, ALL) (2, 22) Blocked_events = None
B2 = Sync(B2, ALL) (2, 22) Blocked_events = None
B3 = Sync(B3, ALL) (2, 22) Blocked_events = None
B4 = Sync(B4, ALL) (2, 22) Blocked_events = None
B = Meet(B1, B2) (4, 40)
B = Meet(B3, B) (8, 72)
B = Meet(B, B4) (16, 128)

A3.3 Get the monolithic supervisor

SUPER = Supcon(PLANT, B) (225, 472)
SUPER = Condat(PLANT, SUPER) Controllable.
SIMSUP = Supreduce(PLANT, SUPER, SUPER) (38, 258; slb = 32)
SIMSUP = Condat(PLANT, SIMSUP) Controllable.

A3.4 Get the decentralized supervisors

SUP1 = Supcon(PLANT, B1) (32, 88)
SIMSUP1 = Condat(PLANT, SUP1) Controllable.
SUP1 = Condat(PLANT, SUP1) Controllable.
SIMSUP1 = Supreduce(PLANT, SUP1, SUP1) (2, 22; slb = 2)
SIMSUP1 = Condat(PLANT, SIMSUP1) Controllable.
SUP2 = Supcon(PLANT, B2) (32, 88)
SUP2 = Condat(PLANT, SUP2) Controllable.
SIMSUP2 = Supreduce(PLANT, SUP2, SUP2) (2, 22; slb = 2)
SUP3 = Supcon(PLANT, B3) (32, 88)
SUP3 = Condat(PLANT, SUP3) Controllable.
SIMSUP3 = Supreduce(PLANT, SUP3, SUP3) (2, 22; slb = 2)
SUP4 = Supcon(PLANT, B4) (32, 88)
SUP4 = Condat(PLANT, SUP4) Controllable.
SIMSUP4 = Supreduce(PLANT, SUP4, SUP4) (2, 22; slb = 2)
TEST = Meet(SIMSUP, PLANT) (225, 472)
true = Isomorph(SUPER, TEST; identity)

A3.5 Design coordinators to solve the blockiness

C1 = Create(C1, [mark 0, [tran[0,11,1],[0,21,1],[1,21,1],[1,11,2], [1,21,2],[1,23,0],[1,33,0],[2,11,3],[2,21,3],[2,23,1],[2,33,1],[3,23,2],[3,33,2]]) (4, 12)
C2 = Create(C2, [mark 0, [tran[0,11,1],[1,11,2], [1,13,0],[2,11,3],[2,13,1],[3,13,2]]) (4, 46)
C3 = Create(C3, [11,21],[13,43]) (4, 6)
C4 = Create(C4, [mark 0, [tran[0,23,1],[0,33,1],[1,13,0],[1,23,2],[1,33,2],[1,43,0],[2,13,1],[2,23,3],[2,33,3],[2,43,1],[3,13,2],[3,43,2]]) (4, 12)
C1 = Sync(C1, ALL) (4, 44) Blocked_events = None
C2 = Sync(C2, ALL) (4, 46) Blocked_events = None
C3 = Sync(C3, ALL) (4, 46) Blocked_events = None
C4 = Sync(C4, ALL) (4, 44) Blocked_events = None
CSUP1 = Supcon(PLANT, C1) (62, 172)
CSUP1 = Supreduce(PLANT, CSUP1, CSUP1) Controllable.
SIMSUP1 = Supreduce(PLANT, SIMSUP1, CSUP1) (8, 86; slb = 4)
CSUP2 = Supcon(PLANT, C2) (56, 164)
CSUP2 = Supreduce(PLANT, CSUP2, CSUP2) Controllable.
CSUP3 = Supreduce(PLANT, CSUP3, CSUP3) (4, 44; slb = 4)
CSUP3 = Supcon(PLANT, C3) (56, 164)
CSUP3 = Supreduce(PLANT, CSUP3, CSUP3) Controllable.
CSUP3 = Supreduce(PLANT, CSUP3, CSUP3) (4, 44; slb = 4)
CMODSUP = Meet(SIM, SIMSUP1) (236, 484)
CMODSUP = Meet(CMODSUP, SIMSUP2) (232, 480)
CMODSUP = Meet(CMODSUP, SIMSUP3) (228, 476)
CMODSUP = Meet(CMODSUP, SIMSUP4) (225, 472)
TEST = Meet(CMODSUP, PLANT) (225, 472)
true = Isomorph(SUPER, TEST; identity)
REFERENCES


Feihua Lu received the B.Eng., and Ph.D. degrees in Control Science and Engineering from Zhejiang University, Hangzhou, China, in 2004 and 2009, respectively. His Ph.D. degree is of the joint education program between Zhejiang University and University of Toronto, Toronto, Canada. He currently works in the College of Economics, Zhejiang University, Hangzhou, China. His research interests include supervisory control of discrete-event systems, Petri nets theory and application.

Weimin Wu received the B.S. and M.S. degrees from the Taiyuan Heavy Machinery Institute, Taiyuan, China, in 1996 and 1999, respectively, and the Ph.D. degree in Control Science and Engineering from Zhejiang University, Hangzhou, China, in 2002. Since 2003, he has been a Faculty Member with the Institute of Cyber-Systems and Control, Zhejiang University, where he is currently an Associate Professor with the State Key Laboratory of Industrial Control Technology. His main research interests include Petri net theories and applications and supervisory-control theory of discrete-event systems.

Hongye Su received the B.S. degree in Industrial Automation from the Nanjing University of Chemical Technology, Nanjing, China, in 1990 and the M.S. and Ph.D. degrees in Industrial Automation from Zhejiang University, Hangzhou, China, in 1993 and 1995, respectively. Since 1995, he has been a Faculty Member with the Institute of Cyber-Systems and Control, Zhejiang University, where he is currently a Professor with the State Key Laboratory of Industrial Control Technology. His recent research interests include robust control, time-delay systems, nonlinear systems, discrete-event systems, and advanced process-control theory and applications.

Jian Chu received the B.Eng., M.Sc., and Ph.D. degrees in Industrial Automation from Zhejiang University, Hangzhou, China, in 1982, 1984, and 1989, respectively. His Ph.D. degree is of the joint education program between Zhejiang University and Kyoto University, Kyoto, Japan. Since 1989, he has been a Faculty Member with the Institute of Cyber-Systems and Control, Zhejiang University, where he is currently a Professor, the Director of the Institute, the Director of the National Laboratory of Industrial Control Technology, and the Deputy Director of the National Engineering Research Center for Industrial Automation. His research areas are advanced process control, robust control, and fieldbus control system. He has published over 300 papers in journals and conference proceedings. Dr. Chu was the recipient of the 1997/1998 IEE Heaviside Premium.