Discrete-Event Coordination Design for Distributed Agents

Manh Tung Pham and Kiam Tian Seow

***Abstract*— This paper presents new results on the formal design of distributed coordinating agents in a discrete-event framework. In this framework, agents are modeled to be individually equipped with a coordination module, through which they interact and communicate. In terms of existing control-theoretic concepts, we first define the concept of a coordinable language and show that it is the necessary and sufficient existence condition of coordination modules for distributed agents to achieve conformance to a pre-specified inter-agent constraint language. Following, we present a synthesis algorithm to compute near-optimal coordination modules. An example is provided to illustrate the design synthesis using the proposed algorithm. Finally, a discussion with related work distinguishes our coordination design problem from related problems in the literature.**

***Note to Practitioners*— Multiagent coordination presents a key approach to developing complex systems. In this approach, the basic idea is to model a system as a network of interacting agents, and design for each agent a coordination module by which the agents can interact and communicate to manage the inter-dependencies arising due to system needs or limitations [1]. In this paper, we propose a novel approach for coordination design of distributed agents. By modeling coordinating agents as discrete-event processes, we formulate and address the problem of synthesizing for each agent a coordination module to achieve conformance to a given inter-agent constraint. Importantly, by showing that our multiagent coordination problem shares the same algorithmic foundation with existing problems in the literature, we are able to adapt existing techniques to develop a new coordination synthesis algorithm, without reinventing the wheel. The coordination modules synthesized by our algorithm are proven to be minimally interventive with the agents' local plans and ensure that communication among the agents is made only when necessary. Potential applications of our work can be found in domains where discrete-event modeling has proven to be suitable and effective, and these include manufacturing [2], transportation [3] and logistics systems [4].**

***Index Terms*— Coordinability, Coordination Design, Control Synthesis, Discrete-Event Systems**

I. Introduction

In modern electronic environments, it is becoming increasingly important to deploy multiple agents with the autonomy capability of coordinating among themselves in conformance to inter-agent constraints [1]. In general, these are social constraints that specify mandatory policies for regulating the interaction behaviors of agents. Intelligent schedulers [5], distributed sensor networks [9], [10], unmanned vehicles [8], distributed networked resource allocation [11] are some examples of multiagent applications where the aforementioned agent regulation capability is a mandatory requirement.

In the abovementioned application domains, agents must follow certain "built-in" strategies realized as coordination modules to coordinate their activity in order to ensure the success of their design goals. For example, providers and consumers of networked resources must operate cooperatively by following a negotiation protocol that regulates their interactions [11]; distributed sensors must coordinate the allocation of their sensing resources in order to accurately track multiple targets [9]; and unmanned train agents accessing a shared two-way tunnel must interact and communicate to respect, say, a "first arrival first access" protocol [7]. Generally, these agents are autonomous entities designed by different groups of people and so it is critical for the coordination modules to be minimally interventive, giving the coordinating agents maximal control of their own operation as long as it does not lead to the violation of the constraint. Moreover, when the underlying communication infrastructure has limited capability or the communication cost is high, it is important that the amount of communication among the agents is minimal. Furthermore, it is desirable that resource requirements in terms of computational time or memory required to implement the coordination module for each agent are minimal.

Modern engineering systems such as those described above, as well as in manufacturing [2], transportation [3] and logistics [4], have to deal with discrete state spaces and event-driven dynamics [14]. To design and deploy systems as distributed coordinating agents, the concern, in one way or another, is with the problem of how agents can interact among themselves to ensure some orderly state-to-state occurrences of interleaving events specified as inter-agent constraints. This motivates our development of a generic design framework for coordinating agents in a discrete-event paradigm.

Previous work [6], [7] has established a fundamental mathematical connection between discrete-event multiagent coordination and supervisory control [12]. The important implication is that coordination planning can be done by utilizing well-established results from discrete-event control theory. That leads to a method using control synthesis for multiagent planning. By modeling agents as discrete-event processes, the method proposes to solve a coordination planning problem by first synthesizing a global supervisor for the system to achieve conformance to a pre-specified inter-agent constraint, and then using it to construct a coordination module for every agent, which together specifies how the agents should interact and communicate. Importantly, using that method,

M.T Pham and K.T. Seow are with the Division of Computing Systems, School of Computer Engineering, Nanyang Technological University, Republic of Singapore 639798. pham0028@ntu.edu.sg, asktseow@ntu.edu.sg

minimal intervention is mathematically guaranteed by discrete-event control theory. However, in the proposed method, every coordination module is constructed almost similarly as the supervisor, which results in their nearly identical structure. So, although conceptually illuminating, the method does not consider that an inter-agent constraint may impose different restrictions on different agents. As a result, some agents may have to interact and communicate more than it is necessary.

This paper borrows and re-interprets discrete-event concepts from supervisory control [12] and sensor selection [13] to develop a novel approach that allows coordination modules to be individually synthesized for each agent. As a result, an agent can expect a simpler coordination module if it is not tightly restricted by the inter-agent constraint. Two new developments are presented in this paper. Firstly, the concept of a coordinable language is introduced, based on which the necessary and sufficient condition for the existence of coordination modules is presented. Secondly, a synthesis algorithm for computing coordination modules, if they do exist, is developed. Importantly, the synthesized coordination modules are not only minimally interventive, but also ensure that communication among the agents is made only when necessary. Each of the coordination modules can also be efficiently implemented in terms of memory requirements as their state size may be greatly reduced, though not necessarily minimized, for each agent model. Such coordination modules are said to be near-optimal.

The rest of the paper is organized as follows. In Section II, we review preliminary results in languages and automata, and in supervisory control theory, that are most relevant to this paper. In Section III, we present our main results on discrete-event multiagent coordination, along with an illustrative design synthesis example in Section IV and a discussion with related work in Section V, before concluding the paper in Section VI.

II. Preliminaries

A. Languages and Automata

Let Σ be a finite alphabet of symbols representing individual events. A *string* is a finite sequence of events from Σ. Denote Σ^* as the set of all strings from Σ including the empty string ε. A string s' is a *prefix* of s if $(\exists t \in \Sigma^*)\ s't = s$. A *language* L over Σ is a subset of Σ^*. Say L_1 is a *sublanguage* of L_2 if $L_1 \subseteq L_2$. The *prefix closure* $\bar{L}$ of a language L is the language consisting of all prefixes of its strings. Clearly $L \subseteq \bar{L}$, because any string s in Σ^* is a prefix of itself. A language L is *prefixed-closed* if $L = \bar{L}$.

Given $\Sigma^1 \subseteq \Sigma^2$, the natural projection $P_{\Sigma^2,\Sigma^1} : (\Sigma^2)^* \to (\Sigma^1)^*$, which erases from a string $s \in (\Sigma^2)^*$ every event $\sigma \in (\Sigma^2 - \Sigma^1)$, is defined recursively as follows: $P_{\Sigma^2,\Sigma^1}(\varepsilon) = \varepsilon$, and $(\forall s \in (\Sigma^2)^*)(\forall \sigma \in \Sigma^2)$,

$$P_{\Sigma^2,\Sigma^1}(s\sigma) = \begin{cases} P_{\Sigma^2,\Sigma^1}(s)\sigma, & \text{if } \sigma \in \Sigma^1; \\ P_{\Sigma^2,\Sigma^1}(s), & \text{otherwise.} \end{cases}$$

For $L \subseteq (\Sigma^2)^*$, $P_{\Sigma^2,\Sigma^1}(L) \subseteq (\Sigma^1)^*$ denotes the language $\{P_{\Sigma^2,\Sigma^1}(s) \mid s \in L\}$. The inverse image of P_{Σ^2,Σ^1}, denoted by $P^{-1}_{\Sigma^2,\Sigma^1}$, is a mapping from $(\Sigma^1)^*$ to $(\Sigma^2)^*$, and defined as: for $L_1 \in (\Sigma^1)^*$, $P^{-1}_{\Sigma^2,\Sigma^1}(L_1) = \{L \subseteq (\Sigma^2)^* \mid P_{\Sigma^2,\Sigma^1}(L) = L_1\}$. Clearly, for $L \in (\Sigma^2)^*$, $P^{-1}_{\Sigma^2,\Sigma^1}(P_{\Sigma^2,\Sigma^1}(L)) \supseteq L$.

If a language is *regular* [14], then it can be *generated* by an automaton. An *automaton* A is a 5-tuple $(X^A, \Sigma^A, \delta^A, x_0^A, X_m^A)$, where X^A is the finite set of states, Σ^A is the finite set of events, $\delta^A : \Sigma^A \times X^A \to X^A$ is the (partial) transition function, x_0^A is the *initial state* and $X_m^A \subseteq X^A$ is the subset of *marker states*.

Write $\delta^A(\sigma, x)!$ to denote that $\delta^A(\sigma, x)$ is defined. An event $\sigma \in \Sigma^A$ is a *strictly self-loop* event of A if $(\forall x \in X^A)[\delta^A(\sigma, x)! \Rightarrow (\delta^A(\sigma, x) = x)]$. Such an event would never bring A from one state to a different state. For an automaton A, Σ^A_{loop} denotes the set of all its strictly self-loop events.

The definition of δ^A can be extended to $(\Sigma^A)^* \times X^A$ as follows: $\delta^A(\varepsilon, x) = x$, and $(\forall \sigma \in \Sigma^A)(\forall s \in (\Sigma^A)^*)\delta^A(s\sigma, x) = \delta^A(\sigma, \delta^A(s, x))$. The behaviors of automaton A can then be described by the prefix-closed language $L(A)$ and the marked language $L_m(A)$. Formally, $L(A) = \{s \in (\Sigma^A)^* \mid \delta^A(s, x_0)!\}$, and $L_m(A) = \{s \in L(A) \mid \delta^A(s, x_0) \in X_m^A\}$.

A state $x \in X^A$ is *reachable* if $(\exists s \in (\Sigma^A)^*)\ \delta^A(s, x_0) = x$, and *coreachable* if $(\exists s \in (\Sigma^A)^*)\ \delta^A(s, x) \in X_m^A$. Automaton A is *trim* if all its states are both reachable and coreachable and so $\overline{L_m(A)} = L(A)$. If A is not trim, then a trim automaton, denoted by $Trim(A)$, can be computed to generate the same marked language as A by deleting from A every state that is either not reachable or not coreachable.

Let A_i, $i \in \{1, 2\}$, be two automata. Then their *synchronous product* A, denoted by $A = A_1 \parallel A_2$, models a discrete-event system (DES) of A_1 and A_2 operating concurrently by interleaving events generated by A_1 and A_2, with synchronization on shared events $\sigma \in \Sigma^{A_1} \cap \Sigma^{A_2}$. Formally, synchronous automaton $A = (X^A, \Sigma^A, \delta^A, x_0^A, X_m^A)$ is computed as follows: $X^A = X^{A_1} \times X^{A_2}$, $X_m^A = X_m^{A_1} \times X_m^{A_2}$, $\Sigma^A = \Sigma^{A_1} \cup \Sigma^{A_2}$, $x_0^A = (x_0^{A_1}, x_0^{A_2})$, and $\delta^A(\sigma, (x_1, x_2))$ is defined by:

$$\begin{cases} (\delta^{A_1}(\sigma, x_1), \delta^{A_2}(\sigma, x_2)), & \text{if } \sigma \in \Sigma^{A_1} \cap \Sigma^{A_2} \text{ and} \\ & \delta^{A_1}(\sigma, x_1)! \text{ and } \delta^{A_2}(\sigma, x_2)!; \\ (\delta^{A_1}(\sigma, x_1), x_2), & \text{if } \delta^{A_1}(\sigma, x_1)! \text{ and } \sigma \notin \Sigma^{A_2}; \\ (x_1, \delta^{A_2}(\sigma, x_2)), & \text{if } \delta^{A_2}(\sigma, x_2)! \text{ and } \sigma \notin \Sigma^{A_1}; \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

It has been shown that if $A = A_1 \parallel A_2$ then $L(A) = P^{-1}_{\Sigma^A,\Sigma^{A_1}}(L(A_1)) \cap P^{-1}_{\Sigma^A,\Sigma^{A_2}}(L(A_2))$ and $L_m(A) = P^{-1}_{\Sigma^A,\Sigma^{A_1}}(L_m(A_1)) \cap P^{-1}_{\Sigma^A,\Sigma^{A_2}}(L_m(A_2))$ [14].

If $\Sigma^{A_1} = \Sigma^{A_2}$, then $L(A_1 \parallel A_2) = L(A_1) \cap L(A_2)$ and $L_m(A_1 \parallel A_2) = L_m(A_1) \cap L_m(A_2)$. The synchronous product of $n \geq 2$ automata A_1, A_2, ... A_n, denoted by $\|_{i=1}^{i=n} A_i$, can be defined recursively using the associativity of $\|$ [14].

B. Control of Discrete-Event Systems

Given a DES $A = (X^A, \Sigma^A, \delta^A, x_0^A, X_m^A)$ to be controlled (A can be built by the synchronous product of a set of simple automata modeling the system components), let the event set Σ^A be partitioned into (i) the *controllable* event set Σ_c^A and the *uncontrollable* event set Σ_{uc}^A, and also into (ii) the *observable* event set Σ_o^A and the *unobservable* event set Σ_{uo}^A. In the control context, a *supervisor* which observes only the observable events in Σ_o^A can modify the behavior of A by disabling only the controllable events in Σ_c^A. Formally, a supervisor S is an automaton over Σ^A ($\Sigma^S = \Sigma^A$) that satisfies the following conditions:

1) S is Σ^A_{uc}-enabling, namely, $(\forall s \in (\Sigma^A)^*)(\forall \sigma \in \Sigma^A_{uc})$ $[(s \in L(S \parallel A)$ and $s\sigma \in L(A)) \Rightarrow s\sigma \in L(S \parallel A)]$.
2) S is Σ^A_{uo}-feasible, namely, $(\forall \sigma \in \Sigma^A_{uo})[((\exists x \in X^S)\delta^S(\sigma, x)!) \Rightarrow \sigma \in \Sigma^S_{loop}]$.

Supervisor S tracks and controls the behavior of DES A. It changes state according to the events generated by A and enables (or allows to occur) at each state only those events defined at that state. That S is Σ^A_{uc}-enabling ensures that uncontrollable events in Σ^A_{uc} will never be prevented from happening. That S is Σ^A_{uo}-feasible means that its state change can only be triggered by the occurrence of events in Σ^A_o which it can observe.

Write S/A for A under the supervision of S. The prefix-closed behavior of the controlled DES, $L(S/A)$, consists of those strings in $L(A)$ containing events enabled by S. The marked behavior $L_m(S/A)$ consists of those strings in $L(S/A)$ that are marked by both S and A. Formally, $L(S/A) = L(S) \cap L(A)$ and $L_m(S/A) = L_m(S) \cap L_m(A)$. Thus, the behavior of system A under the supervision of supervisor S can be modeled by the synchronous automaton $S \parallel A$. Supervisor S is said to be nonblocking (for DES A) if every string generated by S/A can be completed to a marked string, i.e., $\overline{L_m(S/A)} = L(S/A)$. Formally, the general problem of supervisory control may be stated as follows.

Problem 1 (Supervisory Control Problem (SCP)): Given DES A and a control specification automaton C with $\Sigma^C = \Sigma^A$, construct a nonblocking supervisor S for A such that $L_m(S/A) = L_m(A) \cap L_m(C)$.

In the context of SCP, $L_m(C)$ specifies the desired behavior for controlled DES A, i.e., it embodies all the desirable event sequences that one wishes to impose on the system A. In addressing SCP, the following concepts of language controllability and observability are developed by Ramadge and Wonham [12] and Lin and Wonham [15], respectively.

Definition 1: Controllable language [12]: A language $K \subseteq L(A)$ is said to be *controllable* with respect to (w.r.t) A and Σ^A_c (or just controllable if Σ^A_c is understood) if $(\forall s \in \overline{K})(\forall \sigma \in \Sigma^A_{uc})$ $[s\sigma \in L(A) \Rightarrow s\sigma \in \overline{K}]$.

In other words, K is controllable provided no $L(A)$-string which is already a prefix of some string in K, that when followed by an uncontrollable event in Σ^A_{uc}, would exit from $\overline{K}$.

Definition 2: Observable language [15]: A language $K \subseteq L_m(A)$ is said to be *observable* w.r.t A and P_{Σ^A,Σ^A_o} (or just observable if P_{Σ^A,Σ^A_o} is understood) if $(\forall s, s' \in (\Sigma^A)^*)$ for which $P_{\Sigma^A,\Sigma^A_o}(s) = P_{\Sigma^A,\Sigma^A_o}(s')$, the following two conditions are satisfied: (1) $(\forall \sigma \in \Sigma^A)[(s\sigma \in \overline{K}$ and $s' \in \overline{K}$ and $s'\sigma \in L(A)) \Rightarrow s'\sigma \in \overline{K}]$, and (2) $[s \in K$ and $s' \in \overline{K} \cap L_m(A)] \Rightarrow s' \in K$.

The above conditions ensure that Σ^A_o provides a sufficient view for an observer to determine all necessary control and marking actions. Given a DES A and an automaton C with $L_m(C) = K$, the observability property of K can be checked in polynomial time [14]. Note that if $\Sigma^A_o = \Sigma^A$, an arbitrary sublanguage K of $L_m(A)$ is observable.

A fundamental theorem in supervisory control theory [15] may now be stated.

Theorem 1: There is a nonblocking supervisor S, which observes only Σ^A_o and controls only Σ^A_c, such that $L_m(S/A) = K$, if and only if (1) K is controllable w.r.t A and Σ^A_c, and (2) K is observable w.r.t A and P_{Σ^A,Σ^A_o}.

It has been shown that the *supremal controllable sublanguage* [12] of K w.r.t A and Σ^A_c exists, and is equal to K if it is controllable. For an automaton C, the $Supcon(C, A, \Sigma^A_c)$ procedure [16], which computes a nonblocking automaton S such that $L_m(S)$ is the supremal controllable sublanguage of $L_m(A) \cap L_m(C)$, can be implemented with polynomial time complexity [14]. Such automaton S can be used as (the internal model for) a nonblocking supervisor which observes Σ^A and controls Σ^A_c such that $L_m(S/A) = L_m(S)$ [12].

III. Multiagent Coordination

A. Discrete-Event Agents and Coordination

Consider a system of $n \geq 2$ agents modeled by the respective automata $A_i = (X^{A_i}, \Sigma^{A_i}, \delta^{A_i}, x^{A_i}_0, X^{A_i}_m)$ $(1 \leq i \leq n)$, where $\Sigma^{A_i} \cap \Sigma^{A_j} = \emptyset$ for $i \neq j$. From an agent planning viewpoint, automaton A_i is viewed as the local plan of an agent, referred to as agent A_i (or just A_i), encompassing all possible local ways to achieve the agent's local goal. Note that, since each agent is assumed to formulate its local plan independently, the total absence of shared events among coordinating agents, i.e., $\Sigma^{A_i} \cap \Sigma^{A_j} = \emptyset$ for $i \neq j$, is not uncommon.

If the agents operate independently, then the synchronous product $A = \parallel^n_{i=1} A_i$ which encompasses every possible way of interleaving event sequences of the n automata, can be used to describe the system evolution. The agents A_i $(1 \leq i \leq n)$ would need to coordinate among themselves if, due to system needs or limitations, the execution of some event sequences in $L(A)$ is undesirable and should be prevented. If a language $K \subseteq L(A)$ encompasses all the desirable event sequences, then at an appropriate level of abstraction, the coordination problem can be defined as modifying the system in a certain way so that none of those sequences in $L(A) - K$ will ever be generated.

The event set Σ^{A_i} of agent A_i is partitioned into the controllable set $\Sigma^{A_i}_c$ and the uncontrollable set $\Sigma^{A_i}_{uc}$. Interpreted from the coordination viewpoint, an uncontrollable event is inherently autonomous and can be executed solely at the free will of the owner agent. As a rule, an event is pre-specified as uncontrollable if it is critical to the owner agent such that disabling the event and limiting its autonomy just to conform to an inter-agent constraint is undesirable, expensive or impossible. However, unlike in discrete-event control, the observable and unobservable event sets are not pre-specified. It is quite natural to assume that an agent is capable of observing every event of its own. Whether the agent needs to observe the other agents' events will depend on the inter-agent constraint and the coordination means among the agents. Those that need to be observed are coordination-relevant events that must be communicated to the agent.

In enabling distributed agents to coordinate, each agent A_i is equipped with a coordination module (CM) modeled by an automaton S_i defined as follows.

Definition 3: Coordination Module

A coordination module for agent A_i $(1 \le i \le n)$ is an automaton S_i with the following properties:

1) $\Sigma^{S_i} = \Sigma^{A_i} \cup \bigcup_{1\le j\le n, j\ne i} ComSet(S_i, A_j)$, where $ComSet(S_i, A_j) \subseteq \Sigma^{A_j}$, $(1 \le j \le n, j \ne i)$. Σ^{S_i} is called the coordination event set for agent A_i, and $ComSet(S_i, A_j)$ is the subset of events that agent A_j needs to communicate to A_i to synchronize S_i.
2) S_i is $\Sigma_{uc}^{A_i}$-enabling, namely, $(\forall s \in (\Sigma^{S_i})^*)(\forall \sigma \in \Sigma_{uc}^{A_i})$ $[s \in L(S_i \parallel A_i)$ and $P_{\Sigma^{S_i},\Sigma^{A_i}}(s)\sigma \in L(A_i)] \Rightarrow [s\sigma \in L(S_i \parallel A_i)]$.
3) $\forall 1 \le i, j \le n, i \ne j$, S_i and S_j are cooperative, namely, $(\forall s \in (\Sigma^A)^*)(\forall \sigma \in ComSet(S_i, A_j))$ $[P_{\Sigma^A,\Sigma^{A_j}}(s)\sigma \in L(A_j)$ and $P_{\Sigma^A,\Sigma^{S_j}}(s)\sigma \in L(S_j)] \Rightarrow [P_{\Sigma^A,\Sigma^{S_i}}(s)\sigma \in L(S_i)]$.

CM S_i is used by agent A_i to implement its local coordination process. Σ^{S_i} represents the *observation capability* of A_i: a string $s \in L(A)$ is perceived by A_i as $P_{\Sigma^A,\Sigma^{S_i}}(s)$. An agent can observe all its events, hence $\Sigma^{A_i} \subseteq \Sigma^{S_i}$.

By Definition 3, the coordination among discrete-event agents through their respective CM's, as depicted in Fig. 1, can be explained as follows. Following the execution of a string $s \in L(A)$, agent A_i, $1 \le i \le n$, due to partial observation, perceives only $s_i = P_{\Sigma^A,\Sigma^{S_i}}(s)$ and updates the state of CM S_i to $x_i = \delta^{S_i}(s_i, x_0^{S_i})$. Only every event $\sigma_i \in \Sigma^{A_i}$ that is defined at x_i (i.e., $\delta^{S_i}(\sigma_i, x_i)!$) can be enabled (allowed to be executed next) by A_i. That S_i is $\Sigma_{uc}^{A_i}$-enabling guarantees that uncontrollable events are always enabled (hence are never prevented from being executed). With $i \ne j$, that S_i and S_j are *cooperative* ensures that whenever an event $\sigma \in ComSet(S_i, A_j)$ is executed and communicated by agent A_j, A_i can update its CM accordingly. Importantly, this means that if every S_i, $1 \le i \le n$, is an agent A_i's CM, every agent A_j, $1 \le j \le n$, coordinating with agent A_i can independently enable and execute its own events through S_j, and continually do so following its event execution and communication, and in response to a communicated event occurrence. The result is a restriction of the system behavior to a sublanguage of $L(A)$.

Note that the events in the set $\bigcup_{1\le j\le n, j\ne i} ComSet(S_i, A_j)$ or equivalently $(\Sigma^{S_i} - \Sigma^{A_i})$ are those to be communicated to agent A_i by the other agents in the system during coordination. It then follows that the events in the set $\bigcup_{i=1}^{n}(\Sigma^{S_i} - \Sigma^{A_i})$, called the system communication set, are those to be communicated among the agents during coordination.

Let $A_i^{S_i}$ denote agent A_i coordinating with the other agents using its (local) CM S_i, and $\|_{i=1}^{i=n} A_i^{S_i}$ denote the system of n agents A_i coordinating among themselves through their respective CM's. From the foregoing discussions, the behaviors of the coordinated system can be defined as follows.

Definition 4: (Coordinated Behaviors)[1]*:*

1) *Prefix-closed coordinated behavior* $L(\|_{i=1}^{i=n} A_i^{S_i})$
   a) $\varepsilon \in L(\|_{i=1}^{i=n} A_i^{S_i})$,
   b) if $s \in L(\|_{i=1}^{i=n} A_i^{S_i})$, $\sigma \in \Sigma^{A_i}$, $s\sigma \in L(A)$ and $P_{\Sigma^A,\Sigma^{S_i}}(s)\sigma \in L(S_i)$ then $s\sigma \in L(\|_{i=1}^{i=n} A_i^{S_i})$,
   c) no other strings belong to $L(\|_{i=1}^{i=n} A_i^{S_i})$.
2) *Marked coordinated behavior* $L_m(\|_{i=1}^{i=n} A_i^{S_i})$
   $(\forall s \in L(A))$ $s \in L_m(\|_{i=1}^{i=n} A_i^{S_i})$ if and only if $[s \in L(\|_{i=1}^{i=n} A_i^{S_i}))$ and $s \in L_m(A)$ and $(\forall 1 \le i \le n) P_{\Sigma^A,\Sigma^{S_i}}(s) \in L_m(S_i)]$.

[1]The definition of marked coordinated behavior specified herein revises that in the preliminary conference version [17].

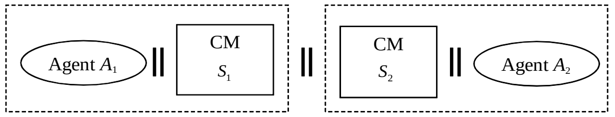


Fig. 1. Two agents A_1 and A_2 coordinating through their respective CM's S_1 and S_2: Following the execution of a string $s \in L(A_1 \parallel A_2)$, agent A_i updates the state of its CM S_i to $x_i = \delta^{S_i}(s_i, x_0^{S_i})$. Only every event $\sigma_i \in \Sigma^{A_i}$ that is defined at x_i (i.e., $\delta^{S_i}(\sigma_i, x_i)!$) is then enabled (allowed to be executed next) by A_i. The result is that the system behavior is restricted to a sublanguage of $L(A_1 \parallel A_2)$.

In Definition 4, $L_m(\|_{i=1}^{i=n} A_i^{S_i})$ consists of every string $s \in L(\|_{i=1}^{i=n} A_i^{S_i}) \cap L_m(A)$ whose projection $P_{\Sigma^A,\Sigma^{S_i}}(s)$ is marked by the respective CM S_i. CM set $\{S_i \mid 1 \le i \le n\}$ is then said to be nonblocking if every string generated during coordination can be completed to a marked string, i.e., $\overline{L_m(\|_{i=1}^{i=n} A_i^{S_i})} = L(\|_{i=1}^{i=n} A_i^{S_i})$.

Since CM's S_i and S_j are cooperative (Definition 3), it can be easily shown that the prefix-closed and marked behaviors of the coordinated system can be equivalently represented by the respective languages generated by the synchronous product $\|_{i=1}^{i=n} (A_i \parallel S_i)$. It follows that $A_i^{S_i}$ can be modeled as $A_i \parallel S_i$, and the coordinated system as a whole can be represented by $\|_{i=1}^{i=n} (A_i \parallel S_i)$.

Now, based on the mathematical equivalence between supervision and multiagent coordination [6], [7], we could borrow and re-interpret discrete-event concepts from supervisory control [12], [15], and define a new concept called language coordinability for coordinating agents, as follows.

Definition 5: Coordinable Language: Given $n \ge 2$ agent automata A_i, $1 \le i \le n$, with $\Sigma^{A_i} \cap \Sigma^{A_j} = \emptyset$ for $i \ne j$. Let $A = \|_{i=1}^{n} A_i$ and $\Sigma_{com} \subseteq \Sigma^A$. A language $K \subseteq L_m(A)$ is said to be *coordinable* w.r.t A and Σ_{com} if

1) K is controllable w.r.t A and $\Sigma_c^A = \bigcup_{i=1}^{n} \Sigma_c^{A_i}$, and
2) K is observable w.r.t A and $P_{\Sigma^A,\Sigma^{A_i}\cup\Sigma_{com}}$, $1 \le i \le n$.

If the two conditions of coordinability in Definition 5 are satisfied, then a set of CM's, with one CM for each agent, can be synthesized such that the overall system behavior conforms to K and the system communication event set is Σ_{com}. This is formally stated in Theorem 2. The proof of this fundamental theorem requires a procedure called CM which computes a CM S_i for an agent A_i, given automaton S and event set $\Sigma^{CM_i} \subseteq \Sigma^S$ for $\Sigma^{S_i} = \Sigma^{CM_i}$, with $L(S_i) = P_{\Sigma^S,\Sigma^{CM_i}}(L(S))$ and $L_m(S_i) = P_{\Sigma^S,\Sigma^{CM_i}}(L_m(S))$.

Computation-wise, procedure CM is the same as the well-known procedure [14] used in supervisory control to compute an observer for a partially observable DES.

Theorem 2: Given $n \ge 2$ agent automata A_i, $1 \le i \le n$, with $\Sigma^{A_i} \cap \Sigma^{A_j} = \emptyset$ for $i \ne j$. Let $A = \|_{i=1}^{n} A_i$, $\emptyset \ne K \subseteq$

Procedure $CM(S, \Sigma^{CM_i})$

```
begin
1   Let $\pi : X_p \to 2^{X^S} - \{\emptyset\}$ be a bijective mapping;
2   Compute automaton $S'_i = (\Sigma^{CM_i}, X_p, \delta^{S'_i}, x_0^{S'_i}, X_m^{S'_i})$
    where:
      • $x_0^{S'_i} \in X_p$ with $\pi(x_0^{S'_i}) = \{\delta^S(s, x_0^S) \mid P_{\Sigma^S,\Sigma^{CM_i}}(s) = \varepsilon\}$;
      • $X_m^{S'_i} = \{x_p \in X_p \mid (\exists s \in L_m(S))\delta^S(s, x_0^S) \in \pi(x_p)\}$;
      • $(\forall\sigma \in \Sigma^{CM_i})(\forall x_p \in X_p)$ [$\delta^{S'_i}(\sigma, x_p)!$ if and only if
        $(\exists s\sigma \in L(S))\delta^S(s, x_0^S) \in \pi(x_p)$];
        When defined, $\delta^{S'_i}(\sigma, x_p) = x'_p$ with
        $\pi(x'_p) = \{\delta^S(s', x) \mid x \in \pi(x_p), P_{\Sigma^S,\Sigma^{CM_i}}(s') = \sigma\}$;
3   Return $S_i = Trim(S'_i)$;
```

$L_m(A)$ and $\Sigma_{com} \subseteq \Sigma^A$. Then, there exists a CM set $\{S_i \mid 1 \le i \le n\}$, where S_i is for A_i, such that $L_m(\|_{i=1}^n A_i^{S_i}) = K$, $L(\|_{i=1}^n A_i^{S_i}) = \overline{K}$ and $\bigcup_{i=1}^n(\Sigma^{S_i} - \Sigma^{A_i}) = \Sigma_{com}$, if and only if K is coordinable w.r.t A and Σ_{com}.

Proof: For economy of notation, let P_i denote $P_{\Sigma^A,\Sigma^{S_i}}$ for $1 \le i \le n$.

(If) Suppose that K is coordinable w.r.t A and Σ_{com}, namely K is controllable w.r.t A and $\Sigma_c^A = \bigcup_{i=1}^n \Sigma_c^{A_i}$, and for all $1 \le i \le n$, K is observable w.r.t A and $P_{\Sigma^A,\Sigma^{A_i}\cup\Sigma_{com}}$. We present a constructive proof that computes a CM set $\{S_i \mid 1 \le i \le n\}$, where S_i is for A_i, such that $L_m(\|_{i=1}^n A_i^{S_i}) = K$, $L(\|_{i=1}^n A_i^{S_i}) = \overline{K}$ and $\bigcup_{i=1}^n(\Sigma^{S_i} - \Sigma^{A_i}) = \Sigma_{com}$.

Let S be a trim automaton with $L_m(S) = K$; and for each $1 \le i \le n$, let $S_i = CM(S, \Sigma^{CM_i})$, where $\Sigma^{CM_i} = \Sigma^{A_i} \cup \Sigma_{com}$. Since the event sets of the agents are pair-wise disjoint and $\Sigma_{com} \subseteq \Sigma^A$, it is obvious that $\bigcup_{i=1}^n(\Sigma^{S_i} - \Sigma^{A_i}) = \bigcup_{i=1}^n((\Sigma^{A_i} \cup \Sigma_{com}) - \Sigma^{A_i}) = \Sigma_{com}$. Following, since K is controllable w.r.t A and Σ_c^A and observable w.r.t A and $P_{\Sigma^A,\Sigma^{A_i}\cup\Sigma_{com}}$ for all $1 \le i \le n$, it is easy to verify that the automata S_i's meet the CM requirements as formalized in Definition 3, namely, $\Sigma^{S_i} \supseteq \Sigma^{A_i}$, S_i is $\Sigma_{uc}^{A_i}$-enabling and S_i and S_j are cooperative for $i \ne j$.

To show that our construction works, it remains to show that $L_m(\|_{i=1}^n A_i^{S_i}) = K$ and $L(\|_{i=1}^n A_i^{S_i}) = \overline{K}$.

To begin with, note that since $S_i = CM(S, \Sigma^{CM_i})$, we have $\Sigma^{S_i} = \Sigma^{A_i} \cup \Sigma_{com}$, $L_m(S_i) = P_i(L_m(S))$ and $L(S_i) = P_i(L(S))$. It follows that $\forall 1 \le i \le n$, K is observable w.r.t A and P_i, and $(\forall s \in (\Sigma^A)^*)$,

$$P_i(s) \in L(S_i) \Leftrightarrow (\exists s' \in \overline{K})P_i(s') = P_i(s) \quad (1)$$

$$P_i(s) \in L_m(S_i) \Leftrightarrow (\exists s' \in K)P_i(s') = P_i(s) \quad (2)$$

- *Proof of* $L(\|_{i=1}^n A_i^{S_i}) = \overline{K}$
  - Since for $1 \le i \le n$, $L(S_i) = P_i(L(S))$, we have $P_i^{-1}(L(S_i)) = P_i^{-1}(P_i(L(S))) \supseteq L(S)$. Therefore,

$$\begin{aligned} L(\|_{i=1}^n A_i^{S_i}) &= L[\|_{i=1}^n (A_i \parallel S_i)] \\ &= L[(\|_{i=1}^n S_i) \parallel (\|_{i=1}^n A_i)] \\ &= L[(\|_{i=1}^n S_i) \parallel A] \\ &= [\bigcap_{i=1}^n P_i^{-1}(L(S_i))] \cap L(A) \\ &\supseteq L(S) \cap L(A) = \overline{K}. \end{aligned}$$

  - We show the other inclusion $L(\|_{i=1}^n A_i^{S_i}) \subseteq \overline{K}$ by induction on the length of strings, as follows.
    - $*$ *Base:* It is obvious that $\varepsilon \in L(\|_{i=1}^n A_i^{S_i}) \cap \overline{K}$.
    - $*$ *Inductive Hypothesis:* Assume that $(\forall s \in (\Sigma^A)^*)$, $|s| = m$ for some $m \ge 0$, $s \in L(\|_{i=1}^n A_i^{S_i}) \Rightarrow s \in \overline{K}$.
      Now we must show that $(\forall\sigma \in \Sigma^A)$ and $(\forall s \in (\Sigma^A)^*)$, $|s| = m$, $s\sigma \in L(\|_{i=1}^n A_i^{S_i}) \Rightarrow s\sigma \in \overline{K}$. This can be done as follows.
      - · Without loss of generality, suppose $\sigma \in \Sigma^{A_i}$ for some $1 \le i \le n$.
      - · By Definition 4, $s\sigma \in L(\|_{i=1}^n A_i^{S_i}) \Rightarrow P_i(s)\sigma \in L(S_i)$.
      - · Since $\sigma \in \Sigma^{A_i}$, by (1), $P_i(s)\sigma \in L(S_i) \Rightarrow ((\exists s'\sigma \in \overline{K})P_i(s') = P_i(s))$.
      - · By inductive hypothesis, $s \in L(\|_{i=1}^n A_i^{S_i}) \Rightarrow s \in \overline{K}$.
      - · Hence, since K is observable w.r.t A and P_i, by Condition 1) of Definition 2, the conditions $P_i(s) = P_i(s'), s \in \overline{K}, s\sigma \in L(A)$ and $s'\sigma \in \overline{K}$ together imply $s\sigma \in \overline{K}$.
    - $*$ Hence the inclusion $L(\|_{i=1}^n A_i^{S_i}) \subseteq \overline{K}$.
- *Proof of* $L_m(\|_{i=1}^n A_i^{S_i}) = K$
  - Since for $1 \le i \le n$, $L_m(S_i) = P_i(L_m(S))$, we have $P_i^{-1}(L_m(S_i)) = P_i^{-1}(P_i(L_m(S))) \supseteq L_m(S)$. Therefore,

$$\begin{aligned} L_m(\|_{i=1}^n A_i^{S_i}) &= L_m[\|_{i=1}^n (A_i \parallel S_i)] \\ &= L_m[(\|_{i=1}^n S_i) \parallel (\|_{i=1}^n A_i)] \\ &= L_m[(\|_{i=1}^n S_i) \parallel A] \\ &= [\bigcap_{i=1}^n P_i^{-1}(L_m(S_i))] \cap L_m(A) \\ &\supseteq L_m(S) \cap L_m(A) = K. \end{aligned}$$

  - We now show the other inclusion $L_m(\|_{i=1}^n A_i^{S_i}) \subseteq K$. Let $s \in L_m(\|_{i=1}^n A_i^{S_i})$, we show $s \in K$, as follows.
    - $*$ By Definition 4, $s \in L_m(\|_{i=1}^n A_i^{S_i}) \Rightarrow [s \in L(\|_{i=1}^n A_i^{S_i})$ and $s \in L_m(A)$ and $(\forall 1 \le i \le n)P_i(s) \in L_m(S_i)]$.
    - $*$ Since $L(\|_{i=1}^n A_i^{S_i}) = \overline{K}$, $s \in L_m(\|_{i=1}^n A_i^{S_i}) \Rightarrow [s \in \overline{K} \cap L_m(A)$ and $(\forall 1 \le i \le n)P_i(s) \in L_m(S_i)]$.
    - $*$ By (2), $(\forall 1 \le i \le n)$ $[P_i(s) \in L_m(S_i) \Rightarrow [(\exists s_i \in K)P_i(s_i) = P_i(s)]]$.
    - $*$ Hence, since $\forall 1 \le i \le n$, K is observable w.r.t A and P_i, by Condition 2) of Definition 2, the conditions $s \in \overline{K} \cap L_m(A), P_i(s) = P_i(s_i)$ and $s_i \in K$ together imply $s \in K$.
    - $*$ Hence the inclusion $L_m(\|_{i=1}^n A_i^{S_i}) \subseteq K$.

(Only If) Suppose that there exists a CM set $\{S_i \mid 1 \le i \le n\}$, where S_i is for A_i, such that $L_m(\|_{i=1}^n A_i^{S_i}) = K$, $L(\|_{i=1}^n A_i^{S_i}) = \overline{K}$ and $\bigcup_{i=1}^n(\Sigma^{S_i} - \Sigma^{A_i}) = \Sigma_{com}$. We show that K is coordinable w.r.t A and Σ_{com}.

By Definition 5, to show that K is coordinable w.r.t A and Σ_{com}, we have to show that

1) K is controllable w.r.t A and $\Sigma_c^A = \bigcup_{i=1}^n \Sigma_c^{A_i}$, and
2) K is observable w.r.t A and $P_{\Sigma^A, \Sigma^{A_i} \cup \Sigma_{com}}$, $1 \leq i \leq n$.

- *Proof of Condition 1 of Coordinality*
  To prove the controllability of K, let $s \in \overline{K}$ and $\sigma \in \Sigma_{uc}^{A_i}$ for some $1 \leq i \leq n$ such that $s\sigma \in L(A)$. By Definition 1, we have to show that $s\sigma \in \overline{K}$. This can done as follows.
  - To begin with, since $\overline{K} = L(\|_{i=1}^n A_i^{S_i})$, by Definition 4, $s \in \overline{K} \Rightarrow s \in L(\|_{i=1}^n A_i^{S_i}) \Rightarrow P_i(s) \in L(S_i)$.
  - Next, since $s\sigma \in L(A)$ and $\sigma \in \Sigma_{uc}^{A_i}$, we have $P_{\Sigma^A, \Sigma^{A_i}}(s)\sigma \in L(A_i)$. Therefore, since CM S_i is $\Sigma_{uc}^{A_i}$-enabling, $[P_{\Sigma^A, \Sigma^{A_i}}(s)\sigma \in L(A_i)$ and $\sigma \in \Sigma_{uc}^{A_i}] \Rightarrow P_i(s)\sigma \in L(S_i)$.
  - Therefore, by Definition 4, the conditions $s \in L(\|_{i=1}^n A_i^{S_i}), s\sigma \in L(A)$ and $P_i(s)\sigma \in L(S_i)$ together imply $s\sigma \in L(\|_{i=1}^n A_i^{S_i})$ or $s\sigma \in \overline{K}$. Hence the controllability condition.
- *Proof of Condition 2 of Coordinality*
  To prove the observability of K, let $s, s' \in (\Sigma^A)^*$ and $\sigma \in \Sigma^A$ such that $P_{\Sigma^A, \Sigma^{A_i} \cup \Sigma_{com}}(s) = P_{\Sigma^A, \Sigma^{A_i} \cup \Sigma_{com}}(s')$ for some $1 \leq i \leq n$. By Definition 2, we have to show that:
  (i) $[s\sigma \in \overline{K}$ and $s' \in \overline{K}$ and $s'\sigma \in L(A)] \Rightarrow s'\sigma \in \overline{K}$, and
  (ii) $[s \in K$ and $s' \in \overline{K} \cap L_m(A)] \Rightarrow s' \in K$.
  To begin with, we first note that since $\bigcup_{i=1}^n (\Sigma^{S_i} - \Sigma^{A_i}) = \Sigma_{com}$, $(\forall 1 \leq i \leq n)$ $\Sigma^{A_i} \cup \Sigma_{com} \supseteq \Sigma^{S_i}$. It follows that $P_{\Sigma^{A_i} \cup \Sigma_{com}}(s) = P_{\Sigma^{A_i} \cup \Sigma_{com}}(s')$ implies $(\forall 1 \leq i \leq n) P_i(s) = P_i(s')$.
  - *Proof of Condition i) of Observability*
    Suppose that $P_{\Sigma^A, \Sigma^{A_i} \cup \Sigma_{com}}(s) = P_{\Sigma^A, \Sigma^{A_i} \cup \Sigma_{com}}(s')$, $s\sigma \in \overline{K}$, $s' \in \overline{K}$ and $s'\sigma \in L(A)$. We show that $s'\sigma \in \overline{K}$, as follows.
    Without loss of generality, suppose $\sigma \in \Sigma^{A_i}$ for some $1 \leq i \leq n$.
    * Since $\overline{K} = L(\|_{i=1}^n A_i^{S_i})$, $[s\sigma \in \overline{K}$ and $s' \in \overline{K}$ and $s'\sigma \in L(A)] \Rightarrow [s\sigma \in L(\|_{i=1}^n A_i^{S_i})$ and $s' \in L(\|_{i=1}^n A_i^{S_i})$ and $s'\sigma \in L(A)]$.
    * By Definition 4, $[s\sigma \in L(\|_{i=1}^n A_i^{S_i})$ and $\sigma \in \Sigma^{A_i}] \Rightarrow P_i(s)\sigma \in L(S_i)$.
    * Since $P_{\Sigma^A, \Sigma^{A_i} \cup \Sigma_{com}}(s) = P_{\Sigma^A, \Sigma^{A_i} \cup \Sigma_{com}}(s')$, $P_i(s) = P_i(s')$ and therefore, $P_i(s)\sigma \in L(S_i) \Rightarrow P_i(s')\sigma \in L(S_i)$.
    * Thus, by Definition 4, the conditions $\sigma \in \Sigma^{A_i}, s' \in L(\|_{i=1}^n A_i^{S_i})$ and $P_i(s')\sigma \in L(S_i)$ together imply $s'\sigma \in L(\|_{i=1}^n A_i^{S_i})$, or $s'\sigma \in \overline{K}$. Hence Condition i) of observability.
  - *Proof of Condition ii) of Observability*
    Suppose $P_{\Sigma^A, \Sigma^{A_i} \cup \Sigma_{com}}(s) = P_{\Sigma^A, \Sigma^{A_i} \cup \Sigma_{com}}(s')$, $s \in K$ and $s' \in \overline{K} \cap L_m(A)$. We show that $s' \in K$, as follows.
    * Since $L(\|_{i=1}^n A_i^{S_i}) = \overline{K}$ and $L_m(\|_{i=1}^n A_i^{S_i}) = K$, $[s \in K$ and $s' \in \overline{K} \cap L_m(A)] \Rightarrow [s \in L_m(\|_{i=1}^n A_i^{S_i})$ and $s' \in L(\|_{i=1}^n A_i^{S_i}) \cap L_m(A)]$.
    * By Definition 4, $s \in L_m(\|_{i=1}^n A_i^{S_i}) \Rightarrow (\forall 1 \leq i \leq n) P_i(s) \in L_m(S_i)$.
    * Since $P_{\Sigma^A, \Sigma^{A_i} \cup \Sigma_{com}}(s) = P_{\Sigma^A, \Sigma^{A_i} \cup \Sigma_{com}}(s')$, $(\forall 1 \leq i \leq n) P_i(s) = P_i(s')$ and therefore, $(\forall 1 \leq i \leq n) P_i(s) \in L_m(S_i) \Rightarrow (\forall 1 \leq i \leq n) P_i(s') \in L_m(S_i)$.
    * Hence, by Definition 4, the conditions $s' \in L(\|_{i=1}^n A_i^{S_i}), s' \in L_m(A)$, and $(\forall 1 \leq i \leq n) P_i(s') \in L_m(S_i)$ together imply $s' \in L_m(\|_{i=1}^n A_i^{S_i})$, or $s' \in K$. Hence Condition ii) of observability.

■

B. Problem Statement and Solution Properties

A general multiagent coordination planning problem may now be stated as follows.

Problem 2 (Multiagent Coordination Problem (MCP)): Given multiagent system $A = \|_{i=1}^{i=n} A_i$ of $n \geq 2$ interacting agents A_i and an inter-agent constraint automaton C with $\Sigma^C = \Sigma^A$, construct a set of nonblocking CM's $\{S_i \mid 1 \leq i \leq n\}$, with S_i for agent A_i, such that $L_m(\|_{i=1}^{i=n} A_i^{S_i}) = L_m(Supcon(C, A, \Sigma_c^A))$ and $L(\|_{i=1}^{i=n} A_i^{S_i}) = L(Supcon(C, A, \Sigma_c^A))$, where $\Sigma_c^A = \bigcup_{i=1}^n \Sigma_c^{A_i}$.

In the context of MCP, $L_m(C)$ specifies the desired behavior, i.e., it embodies all the desirable event sequences that one wishes to impose on the system A. A CM set $\{S_i \mid 1 \leq i \leq n\}$ that satisfies the conditions stated in Problem 2 is said to be *minimally interventive* since, using these CM's, each agent A_i does not unnecessarily disable its controllable events, unless not doing so could lead eventually to the violation of the inter-agent constraint C.

When solving MCP, it is desirable to synthesize optimal CM's, i.e., minimally interventive CM's with the following additional properties:

1) *Minimal Communication*: The cardinality of the event set to be communicated among the agents is minimal. Such a property is desirable when the underlying infrastructure has limited capability or the communication cost is high.

2) *Efficient Implementation*: Each CM S_i is of minimal state size (among all minimally interventive CM's for agent A_i satisfying the minimal communication property), and so can be efficiently implemented in terms of memory requirements.

Remark 1: That the cardinality of the event set to be communicated among the agents is minimal does not necessary imply that the actual number of events that the agents communicate among themselves during coordination will be minimal. It is possible that some events in a minimal cardinality communication set are executed and communicated more frequently than the events in some larger communication set, resulting in a larger number of events communicated during coordination. However, in order for our coordination framework to be widely applicable, the underlying mechanism with which a coordinating agent selects an enabled event to execute is assumed not modeled. It follows that quantitative information such as the frequencies of event occurrences are not known *a priori*, and we therefore postulate that the problem of minimizing inter-agent communication can be logically addressed as that of minimizing the cardinality of the event set to be communicated among the agents.

As will be explained, by Definition 5 and Theorem 2, MCP can be solved by re-interpreting and utilizing control synthesis methods developed for SCP (Problem 1).

Definition 6: Minimal Inter-Agent Communication Set: Let $L \subseteq L_m(A)$. A subset Σ' of Σ^A is a minimal (cardinality) inter-agent communication set (of A for L) if

1) L is observable w.r.t A, $P_{\Sigma^A,\Sigma^{A_i}\cup\Sigma'}$ for all $1 \leq i \leq n$.
2) $(\forall \Sigma'' \subseteq \Sigma^A)$ if L is observable w.r.t A and $P_{\Sigma^A,\Sigma^{A_i}\cup\Sigma''}$ for all $1 \leq i \leq n$ then $|\Sigma'| \leq |\Sigma''|$.

A minimal (cardinality) inter-agent communication set is denoted by $MinSysComSet(L, A)$.

By Definition 5 and Theorem 2, when the agents are coordinating to achieve conformance to an inter-agent constraint prescribed by the language L, $MinSysComSet(L, A)$ in Definition 6 is a set of events that must be communicated among the agents during coordination. Computing such an event set is mathematically equivalent to a variant of the minimal (cardinality) sensor-selection problem [13], [18]. The original sensor-selection problem [13] is addressed in the control context, on the premise that event observation incurs sensor installation cost. Given the language L for system A, the problem is finding a minimal cardinality event set $\Sigma' \subseteq \Sigma^A$ whose observation ensures the observability of L w.r.t A and $P_{\Sigma^A,\Sigma'}$. Formally, the statement of the sensor-selection problem [13] can be given as follows.

Problem 3 (Minimal sensor-selection problem [13]): Given a DES A and an automaton H with $L_m(H) = L \subseteq L_m(A)$, find an event set $\Sigma' \subseteq \Sigma^A$ (of minimal cardinality) that satisfies the following conditions: (1) L is observable w.r.t A and $P_{\Sigma^A,\Sigma'}$, and (2) $(\forall \Sigma'' \subseteq \Sigma^A)(L$ is observable w.r.t A and $P_{\Sigma^A,\Sigma''} \Rightarrow |\Sigma'| \leq |\Sigma''|)$.

The minimal sensor-selection problem has been shown to be NP-hard [19]. Its current state-of-the-art solutions include an exponential time exact algorithm [13] and polynomial time approximate algorithms [18], [25]. Besides not guaranteeing a minimal cardinality solution set, the latter are also only applicable under some assumptions. $MinSysComSet(L, A)$ could be computed by adapting the former [13].

Remark 2: Given L and A, the original algorithm [13] considers all subsets of Σ^A and selects from them a minimal cardinality event set Σ' for which the observability of L w.r.t A and $P_{\Sigma^A,\Sigma'}$ holds. In modifying the algorithm for $MinSysComSet(L, A)$, we consider all subsets of Σ^A and select from them a minimal cardinality event set Σ' for which the observability of L w.r.t A and $P_{\Sigma^A,\Sigma^{A_i}\cup\Sigma'}$ holds for all $1 \leq i \leq n$.

By Definition 6, a language $K \subseteq L_m(A)$ is always observable w.r.t A and $P_{\Sigma^A,\Sigma^{A_i}\cup MinSysComSet(K,A)}$ for all $1 \leq i \leq n$. If K is also controllable, then by Definition 5, it is coordinable w.r.t A and $MinSysComSet(K, A)$. It follows that, to guarantee minimal communication among $n \geq 2$ agents coordinating to achieve K, by Theorem 2 and its constructive proof, CM's S_i can be computed such that $\bigcup_{i=1}^{n}(\Sigma^{S_i} - \Sigma^{A_i}) = MinSysComSet(K, A)$, and done with $\Sigma^{S_i} = \Sigma^{A_i} \cup MinSysComSet(K, A)$.

Remark 3: For a coordinable $K \subseteq L_m(A)$ w.r.t A and $MinSysComSet(K, A)$, there might exist some other CM's S_j, $1 \leq j \leq n$, with $\bigcup_{i=1}^{n}(\Sigma^{S_i} - \Sigma^{A_i}) = MinSysComSet(K, A)$, for which Σ^{S_j} is a strict subset of $\Sigma^{A_j} \cup MinSysComSet(K, A)$. Such CM's might further reduce the communication needs for some individual sending agents. However, as explained above, CM's S_i with $\Sigma^{S_i} = \Sigma^{A_i} \cup MinSysComSet(K, A)$ for all $1 \leq i \leq n$ do guarantee minimal communication at the system level.

Finally, from the control viewpoint, consider A_i, when self-looped at each state with events in $\Sigma^{S_i} - \Sigma^{A_i}$, as a plant to be supervised. Then each CM S_i can be thought of as a supervisor controlling the plant to achieve the controlled behavior $S_i \parallel A_i$. The problem of minimizing the state size of CM S_i for agent A_i is therefore mathematically related to the minimal supervisor problem [20]. The original minimal supervisor problem [20], which is formally stated below, is proposed in the control context to address the economy of implementation in terms of memory requirements for the supervisor.

Problem 4 (Minimal supervisor problem [20]): Given a DES A and a supervisor S for A with $\Sigma^S = \Sigma^A$, synthesize a minimal state size supervisor automaton S_{min} that is control equivalent to S, i.e., a supervisor automaton that satisfies the following conditions: (1) $L(S_{min}) \cap L(A) = L(S) \cap L(A)$ and $L_m(S_{min}) \cap L_m(A) = L_m(S) \cap L_m(A)$, and (2) $(\forall S')(L(S') \cap L(A) = L(S) \cap L(A)$ and $L_m(S') \cap L_m(A) = L_m(S) \cap L_m(A)) \Rightarrow |X^{S_{min}}| \leq |X^{S'}|)$.

The minimal supervisor problem has been shown to be NP-hard [20]. Thus, the computational complexity of an algorithm addressing Problem 4 is expected to be exponential. To mitigate the computational hardness of synthesizing a minimal state size supervisor S_{min} for Problem 4, a heuristic (polynomial time) reduction procedure called $Supreduce$ [20] is proposed. The procedure $Supreduce(S, A)$ synthesizes and returns a reduced state size supervisor $S_{reduced}$ which is control equivalent to the given supervisor S for DES A. Formally, the synthesized supervisor $S_{reduced}$ is an automaton with $L(S_{reduced}) \cap L(A) = L(S) \cap L(A)$, $L_m(S_{reduced}) \cap L_m(A) = L_m(S) \cap L_m(A)$ and ideally should have $X^{S_{reduced}} << X^S$. Numerical experimentation has shown that $Supreduce$ can often return a supervisor $Supreduce$ that has significant state size reduction for a moderate size supervisor S [20].

Procedure $CMreduce(S_i, A_i)$

```
begin
1    Add every event in $(\Sigma^{S_i} - \Sigma^{A_i})$ as a self-loop
     transition at every state of $A_i$. Let the resulting
     automaton be $A'_i$;
2    Return $Supreduce(S_i, A'_i)$;
```

Thus, procedure $Supreduce$ [20] could be modified as a new procedure called $CMreduce$, in attempting to address the *efficient implementation* of CM's S_i, $1 \leq i \leq n$. The result is that, given S_i and A_i, $CMreduce(S_i, A_i)$ can often return a greatly state-size reduced CM automaton for agent A_i achieving the same behavior of $S_i \parallel A_i$. Since for all $1 \leq i \leq n$, $L(A_i \parallel S_i) = L(A_i || CMreduce(S_i, A_i))$ and $L_m(A_i \parallel S_i) = L_m(A_i || CMreduce(S_i, A_i))$, it can be easily shown that $CMreduce(S_i, A_i)$ returns a CM as formalized in Definition 3.

To highlight, procedure $CMreduce$ is different from the original procedure $Supreduce$ [20] in that it is for $S_i \parallel A_i$ with $\Sigma^{S_i} \supseteq \Sigma^{A_i}$, instead of that with $\Sigma^{S_i} = \Sigma^{A_i}$.

C. Coordination Module Synthesis

An important implication of the preceding discussions is that the discrete-event techniques of control and sensor selection can be adapted and applied to synthesize CM's for distributed agents.

Algorithm 1: Coordination Module Synthesis

```
Input: $n \geq 2$ agents $A_1$, $A_2$, ..., $A_n$ with $\Sigma^{A_i} \cap \Sigma^{A_j} = \emptyset$
       for $1 \leq i \neq j \leq n$, and constraint $C$ where
       $\Sigma^C = \bigcup_{i=1}^n \Sigma^{A_i}$
Output: A near-optimal nonblocking CM set
        $\{S_i \mid 1 \leq i \leq n\}$, where $S_i$ is for $A_i$, such that
        $L_m(\|_{i=1}^n A_i^{S_i}) \subseteq L_m(\|_{i=1}^n A_i) \cap L_m(C)$
begin
    Compute automaton $A$ and controllable set $\Sigma_c^A$
1       $A \leftarrow \|_{i=1}^n A_i$; $\Sigma_c^A \leftarrow \bigcup_{i=1}^n \Sigma_c^{A_i}$;
    Compute a nonblocking supervisor $S$
2       $S \leftarrow Supcon(C, A, \Sigma_c^A)$;
    Compute coordination event sets $\Sigma^{CM_i}$, $1 \leq i \leq n$
3       $\Sigma^{CM_i} \leftarrow \Sigma^{A_i} \cup MinSysComSet(L_m(S), A)$;
    Compute CM's $S_i$, $1 \leq i \leq n$
4       $S_i \leftarrow CM(S, \Sigma^{CM_i})$;
    Reduce state size of the CM's $S_i$, $1 \leq i \leq n$
5       $S_i \leftarrow CMreduce(S_i, A_i)$;
6   Return CM set $\{S_i \mid 1 \leq i \leq n\}$;
```

Algorithm 1 details the planning steps for coordination design of $n \geq 2$ agents. Given n agents A_i, $1 \leq i \leq n$, and inter-agent constraint C, it constructs a CM set $\{S_i \mid 1 \leq i \leq n\}$, where S_i is for A_i, which is nonblocking and minimally interventive, i.e., $L_m(\|_{i=1}^n A_i^{S_i})$ is equal to the supremal controllable sublanguage of $L_m(A) \cap L_m(C)$. Moreover, the event set to be communicated among the agents is minimal among all the CM sets that achieve the same coordinated behavior. Furthermore, the state size of each CM S_i is often significantly reduced by the $CMreduce$ procedure. Whereas the two properties of minimal intervention and communication are guaranteed, the CM's returned by the algorithm have a relatively small state size that is not necessarily minimal; hence these CM's can only be said to be *near-optimal*. The correctness of the proposed algorithm is immediate based on the previous discussions. The planning steps presented may utilize TCT [16], a freely available software for DES synthesis.

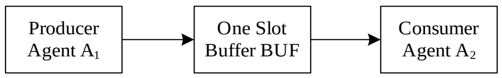
Note that every event in $MinSysComSet(L_m(S), A)$ will need to be communicated by one agent and received by at least one other agent A_j. However, in cases where an event in $MinSysComSet(L_m(S), A)$ appears as a self-loop transition at every state of agent A_j's CM, it need not be communicated to agent A_j. As a matter of practical interest, every such event should be removed from agent A_j's CM to obtain a new CM S_j as in Remark 3 for actual implementation.

In the worst case, Algorithm 1 has exponential time complexity because all the following procedures suffer from exponential complexity: the synchronous product of A, the projection of S onto the event subset $\Sigma^{CM_i} \subseteq \Sigma^S$ in Procedure CM, and the construction of the minimal inter-agent communication set in Procedure $MinSysComSet$. This exponential complexity is not surprising since in our MCP, the multiagent system $A = \|_{i=1}^n A_i$, the inter-agent constraint C and each CM S_i, when self-looped at each state with events in $\Sigma^A - \Sigma^{S_i}$, share the same mathematical models as an n-component modular DES plant, a control specification and a supervisor controlling the plant, respectively; and the problem of synthesizing multiple supervisors for such a modular DES subject to a specification has been shown to be NP-hard [21].

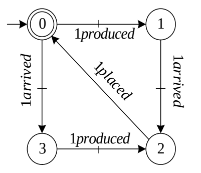
When necessary, heuristic algorithms or data structures should be designed for the synthesis of large systems. It is envisioned that existing work on complexity reduction in discrete-event systems could be utilized to address this design issue. For example, binary decision diagrams and state tree structures [22] have proven to be efficient encoding techniques for reducing the complexity of synthesizing the product A and the automaton S. To mitigate the complexity of Procedure CM, each event subset Σ^{CM_i} could be enlarged such that $P_{\Sigma^S, \Sigma^{CM_i}}$ becomes a natural observer of $L_m(S)$ [23], before applying Procedure CM to compute CM S_i from S and Σ^{CM_i}. It has been shown that if $P_{\Sigma^S, \Sigma^{CM_i}}$ is a natural observer of $L_m(S)$, then computing a projection of S on Σ^{CM_i} only requires polynomial time w.r.t the state size of S [23]. However, one should note that enlarging Σ^{CM_i} would increase the number of events to be communicated among coordinating agents. To mitigate the complexity of Procedure $MinSysComSet$, the polynomial time approximate algorithms for sensor selection [18], [25] may be adapted and applied to compute a small but not necessarily minimal inter-agent coordination event set. However, this approach also compromises the minimal inter-agent communication property guaranteed by Algorithm 1. Thus, one might have to trade-off between the computational complexity of synthesizing CM's and the number of events to be communicated among coordinating agents. A deeper investigation of this issue is an interesting topic that is beyond the scope of this paper.

IV. Illustrative Example

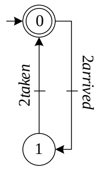
We now present an example to illustrate the use of Algorithm 1 and the design improvement it offers over the original approach [6], [7]. The system under study is a manufacturing system consisting of two agents A_1 and A_2 connected in tandem and separated by a one-slot buffer BUF [Fig. 2(a)]. The working environment considered requires autonomous and "active" agents capable of coordinating between themselves to achieve the goal of satisfying a given inter-agent constraint. These agents would need to be individually equipped with CM's implementing built-in strategies for achieving the goal. In a conventional manufacturing environment, we might be able to treat each agent as a "passive" process to be controlled,



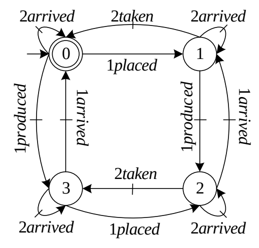
(a) Overall structure of the manufacturing system



(b) Producer agent A_1



(c) Consumer agent A_2



(d) Inter-agent constraint C

Fig. 2. Coordination planning for a manufacturing system

and develop external supervisors - with each over the system of passive agents – using a decentralized control framework [24], [26]. However, as will be discussed in the next section, treating agents as passive processes to be controlled is not always possible or the most appropriate, and may have some limitations in distributed system modeling.

In all subsequent figures for the example, an automaton G is represented by an edge-labelled directed graph with a state represented by a node, and a transition $\delta^G(\sigma, x) = x'$ by a directed edge from state x to x' labelled with the symbol $\sigma \in \Sigma^G$ of an event whose occurrence it represents. The σ-labelled edge is drawn as a directed line with an optional tick ($\circ\!\!-\!\!\!+\!\!\!\rightarrow\!\!\circ$) if the event $\sigma \in \Sigma^G$ is controllable. The initial state is represented by a node with an entering arrow, and a marker state by a node drawn as a double concentric circle.

Initially, the buffer is empty. A_1 is a producer that produces workpieces continually, one piece at a time, and places them into the buffer BUF. In order to do so, A_1 has to produce a workpiece, go to the buffer place, and place the workpiece into the buffer. According to its local plan, A_1 can either produce a workpiece first or go to the buffer place first [Fig. 2(b)]. A_2 is a consumer that consumes workpieces continually, one piece at a time, from the buffer BUF. To do so, A_2 first needs to go to the buffer place. Upon arriving, it then takes a workpiece from BUF and returns to its initial state for a new consumption cycle [Fig. 2(c)]. In this example, we arbitrarily fix $\Sigma_c^A = \{1produced, 1arrived, 2arrived, 2taken\}$ and $\Sigma_{uc}^A = \{1placed\}$.

The inter-agent constraint is stated as follows: *Producer agent must produce a workpiece first before going to the buffer place. Moreover, the buffer must never overflow or underflow.*

Intuitively, the constraint requires $1produced$ to be executed first, and $1arrived$ and $1produced$ must be executed alternately thereafter; and similarly for $1placed$ and $2taken$. Clearly, the constraint imposes more restrictions on the plan of A_1 than that of A_2. Hence, one would expect CM S_1 to be more complex than CM S_2. The textual description of the constraint can be formulated by automaton C [Fig. 2(d)]. How this can be done is simply taken for granted here. Such an automaton may be more easily prescribed with the aid of a high-level specification translator [27].

Nonblocking CM pair (S_1, S_2) with $L_m(A_1^{S_1} \parallel A_2^{S_2}) \subseteq L_m(A) \cap L_m(C)$ may now be synthesized using Algorithm 1. After Line 2, automaton $S = SupCon(C, A, \Sigma_c^A)$, whose marked language $L_m(S)$ is the supremal controllable sublanguage of $L_m(A) \cap L_m(C)$, is constructed [see Fig. 3]. Following Line 3, we obtain $MinSysComSet(L_m(S), A) = \{1placed, 2taken\}$, $\Sigma^{CM_1} = \{1produced, 1arrived, 1placed, 2taken\}$ and $\Sigma^{CM_2} = \{2arrived, 2taken, 1placed\}$; using which, CM's $S_i = CM(S, \Sigma^{CM_i})$, $i \in \{1, 2\}$, are computed at Line 4 [see Fig. 4]. Finally, the state size of these CM's is reduced by procedure $CMreduce$ at Line 5, and the synthesis CM solution is returned at Line 6 [see Fig. 5]. To elaborate, using these CM's means: A_1 must inform A_2 whenever it places a workpiece into the buffer, and A_2 reciprocates in turn whenever it takes a workpiece from the buffer. For this example, the CM's S_1 and S_2 returned by Algorithm 1 are verified to be optimal, i.e., each is of minimal state size among all the minimally interventive CM's that entail minimal communication between the agents.

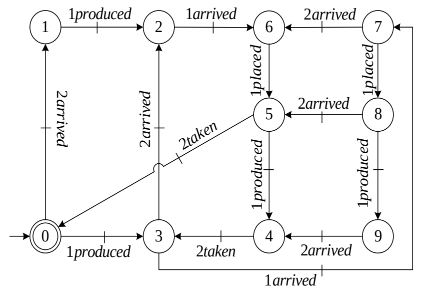
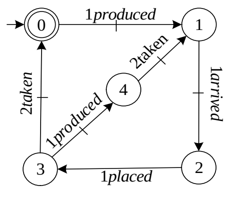
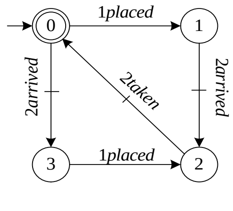


Fig. 3. Automaton S with $L_m(S)$ - the supremal controllable sublanguage of $L_m(A) \cap L_m(C)$



(a) CM S_1 for producer agent A_1



(b) CM S_2 for consumer agent A_2

Fig. 4. Coordination modules for the manufacturing agents before applying the state-size reduction procedure $CMreduce$

In [6], [7], a different approach to the synthesis of CM's is presented. There, a supervisor S' is first computed by applying procedure $Supreduce$ to automaton S [see Fig. 6]. Following, CM's S_1' and S_2', with S_i' for A_i, are computed by removing from S' all its strictly self-loop events that are not defined in agent models A_1 and A_2, respectively [see Fig. 7]. The supervisor S' returned might be minimally reactive as is the case in this example, i.e., S' has the least number of states and

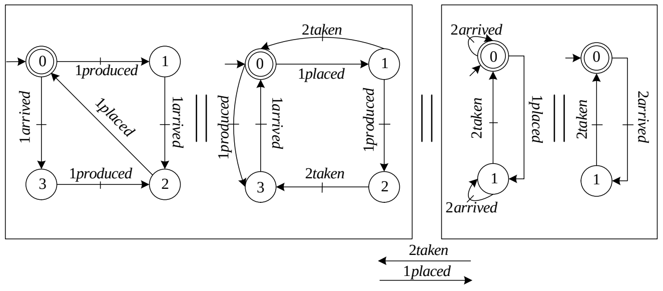


Fig. 5. Coordination modules for the manufacturing agents

the largest number of strictly self-loop events among those that can control A to satisfy C [7]. With this method, a minimally reactive S' is the best outcome though this is not guaranteed in general. Synthesizing from such an S' produces rather state-efficient CM's S_i' with a reduced number of events (in $\Sigma^A - \Sigma_{loop}^{S'}$) to be communicated between the agents.

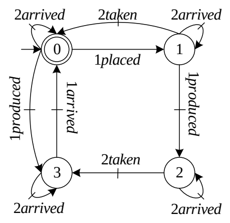
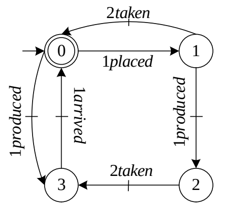
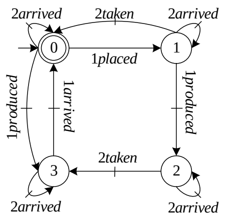


Fig. 6. Minimally reactive supervisor S'



(a) CM S_1' for producer agent A_1



(b) CM S_2' for consumer agent A_2

Fig. 7. Non-optimal coordination modules for the manufacturing agents

However, as this example clearly illustrates, Algorithm 1 offers an improvement in coordination design synthesis over the original method [6], [7], even with the latter producing a minimally reactive S' for CM design. Firstly, the system communication set $\{1placed, 2taken\}$ between the agents (using CM's S_1 and S_2) is a subset of that when using CM's S_1' and S_2', which is $\{1produced, 1arrived, 1placed, 2taken\}$, thanks to the $MinSysComSet$ procedure. Secondly, CM S_2 has a smaller state size than CM S_2', and thus has a lower memory requirement. Achieving this higher level of efficiency in implementation is the result of individually reducing the CM's state size using procedure $CMreduce$ in Line 5 of Algorithm 1, instead of reducing them 'together' using procedure $Supreduce$, as in the original method. In essence, the individual state-size reduction of CM's exploits the fact that an inter-agent constraint may impose different restrictions on different agent models, and thus the less restricted agent should expect a simpler CM.

V. Related Work

A. In Agent Research

Among related work under the same discrete-event paradigm, we have earlier discussed the design improvement in synthesizing CM's that Algorithm 1 offers over an existing approach [6], [7]. We shall now discuss our discrete-event coordination framework in relation to some other important frameworks for coordinating agents.

Yokoo et al. [9], [10] study a Distributed Constraint Satisfaction Problem (DCSP). In its most basic form, a DCSP consists of a finite set of agents and their inter-agent constraints. An agent is represented by a variable and its set of possible actions is represented by the variable's domain of discrete values. Each inter-agent constraint is a restriction on the actions among the agents, specified as a predicate on the domain values of their variables. The problem of interest is then for the agents to cooperatively find a combination of their actions that satisfies these inter-agent constraints. It has been shown that various important multiagent application problems can be formalized and addressed as DCSPs [9]. Examples include distributed resource allocation, distributed sensor network and distributed scheduling. However, to the best of our knowledge, the DCSP framework could not be readily exploited to address the coordination problem among interacting agents modeled as discrete-event processes. Our work runs in parallel with the DCSP framework, presenting a new constraint satisfaction foundation for discrete-event agents. An advantage of our work is that it is expressed in the rudimentary framework of languages and automata, and hence can furnish a theoretical basis for a wide range of applications.

In another research direction, a formal agent model called Markov Decission Process (MDP) [28] has been generalized in different ways to model a network of coordinating agents [29], [30]. In these MDP-extension frameworks, a coordination strategy for an individual agent maps a set of historical system state sequences observed by the agent onto a set of the agent's local actions, each represented by a probability distribution over the agent's (next) local states. The problem of interest is to synthesize optimal coordination strategies for MDP agents to maximize some given reward function defined on the set of possible system state sequences. A key feature that differentiates our work from these research efforts is that we treat events rather than states as the fundamental concept, and model them as explicit transitions in an agent structure. That enables interesting characteristics of agents to be modeled using the properties of events. For instance, the autonomy of coordinating agents can be modeled using controllable and uncontrollable events, as explained in Section III-A. Moreover, it is argued that in some technical situations, when compared to the state-based modeling approach, the event-based modeling approach is more flexible and may be computationally more advantageous [31]. This is because the structural information of a system, which is not reflected in state-based models, is readily captured by the concept of events in event-based models.

Finally, one feature of our work is that we assume an inter-agent constraint has already been given and focus on synthesizing CM's for the agents to satisfy the constraint. In contrast, there have been extensive research efforts focusing on how to specify inter-agent constraints. For example, Rosenschein and Zlotkin [32] investigate on how to design inter-agent constraints, referred to as "rules of encounters", to restrict individual self-interested agents to act in a certain way that benefits the whole system. Shoham and Tennenholtz [33], [34] study the problem of designing social constraints to ensure that interacting agents will not arrive at situations that can lead to conflict. An interesting direction for future work is to integrate these research efforts with ours in a unified framework so that the whole process of specifying inter-agent constraints and designing CM's for interacting agents can be fully automated.

B. In Control Research

The proof of Theorem 2 has utilized established mathematical results for the existence of a supervisor in discrete-event control theory [15]. However, multiagent coordination and supervisory control are conceptually different problems. As first explained in [6], [7], the latter entails enablement or disablement of events in a DES by external supervisors, while the former entails interaction and communication among agents in the DES through their local CM's. Below, we attempt to further distinguish our coordination design problem from related problems in the control literature.

Lin and Wonham [24] and Rudie and Wonham [26] study the decentralized discrete-event control problem which shares the same mathematical foundation with but is different from our problem of multiagent coordination, as will be explained later in this section. The decentralized control problem is to synthesize multiple supervisors, each with different observation and control capabilities, that jointly control a DES to achieve conformance to a given global control specification. The observation and control capabilities of individual (decentralized) supervisors are predetermined respectively as subsets of observable and controllable events of the DES, and can be different for each supervisor.

In [26], an important condition is established, specifying that a set of decentralized supervisors exists if and only if the global specification language satisfies the two properties of controllability and co-observability. It has been shown that the latter property reduces to the observability [15] of the specification language with respect to each of the supervisors' observable event subset, if the supervisors' controllable event subsets are pair-wise disjoint [14]. Thus, for the case of $n \geq 2$ supervisors controlling a DES $A = \|_{i=1}^{n} A_i$ with $\Sigma^{A_i} \cap \Sigma^{A_j} = \emptyset$ for $1 \leq i \neq j \leq n$, and with each supervisor i observing what is equivalent to the event set $\Sigma^{A_i} \cup \Sigma_{com}$ and controlling event set $\Sigma_c^{A_i}$, a controllable and co-observable language is mathematically equivalent to a coordinable language given in Definition 5.

When the co-observability of the global specification language is not satisfied, decentralized supervisors may communicate observable events among themselves to establish co-observability [35], [36], [25]. Among the first research efforts, Wong and Schuppen [35] present a necessary and sufficient condition for solving the decentralized control problem with communication by formulating a refinement relation between observation and control. Barrett and Lafortune [36] develop a decentralized control with communication framework using state estimation. Rohloff and Schuppen [25] discuss the relationship between the minimal communication and minimal sensor selection problems, and propose several heuristic algorithms for approximating the minimal communication event sets among decentralized supervisors.

Like [25], we also study the problem of minimal communication, but in a new discrete-event multiagent coordination setting. In our multiagent coordination framework, the sets of observable events constitute the system communication set Σ_{com}, which is a union of local event subsets of the sending agents that need to be determined for each receiving agent. Unlike decentralized control and supervisory control in general, the observable events for a receiving agent (or events to be communicated to the agent when they occur) are not pre-determined but computed with the aim of minimizing communication, and therefore can be different for a different inter-agent constraint. Applying our synthesis algorithm for two agents, the local event subset $\Sigma_{com} \cap \Sigma^{A_i}$ of the sending agent A_i can be determined and minimized for the receiving agent A_j, thereby minimizing communication among the coordinating agents. Minimizing communication is a problem of significant importance for application domains in which communication bandwidth is a scare resource.

Significantly, what distinguishes our multiagent coordination problem from the decentralized control problem (with or without communication) reviewed above is that, the former arises in application domains such as robotic agents where distributed agent autonomy is a key consideration, whereas the latter arises in domains where it is not and the discrete-event processes are not active agent models but (can be treated as) passive system components to be controlled. From this perspective, the supervisory control framework presents a "supervisor-subordinate" architecture where a plant (which can consist of multiple components) is monitored and controlled by a single supervisor or a set of decentralized supervisors. In contrast, our multiagent framework presents a "peer-to-peer" architecture where distributed agents operate independently but cooperatively to achieve conformance to some system level constraint. Unlike a component to be controlled in the supervisory control framework, which acts passively under the direction of external supervisors, an agent in our framework acts actively following its own defined plan and its built-in coordination strategy (represented by its CM).

Recently, the problem of minimizing communication between interacting discrete-event systems has been gaining increasing attention. Rudie, Lin and Lafortune [37], [38] consider a problem where one control agent (or supervisor) communicates with another agent for information so as to distinguish the states of its automaton, or recognize the set of transitions pre-specified as essential, for control decision-making or diagnosis. Since communication may be costly, a strategy to reduce communication between agents is developed. Like theirs, we also seek to reduce communication between agents, but consider a different problem where the agents coordinate by interacting and communicating for information so as to cooperatively satisfy an inter-agent constraint.

Finally, to emphasize, a coordination module in our framework is conceptually different from a local supervisor in modular supervisory control [39]. The former is an interface through which a coordinating agent interacts and communicates with other agents in a multiagent system. The latter is an external supervisor controlling a group of discrete-event processes in a modular DES plant.

VI. Conclusion and Future Work

This paper has presented new results on multiagent coordination in a discrete-event framework. A fundamental insight unearthed in this paper is that *minimal sensor selection* and *minimal agent communication* actually share the same algorithmic solution although they are conceptually different domains. Importantly, this insight, together with earlier work [6], [7] establishing the mathematical connection between supervisory control and multiagent coordination, leads us to adapting and applying the DES algorithmic foundation to develop an algorithm (Algorithm 1) to synthesize coordination modules for coordinating agents. As guaranteed by an established theoretical result (Theorem 2), the synthesis algorithm (Algorithm 1) computes and returns near-optimal coordination modules which are minimally interventive and entail minimal communication among the agents. Moreover, they are state-reduced individually for each agent, leading to design improvement over the original method [6], [7] as demonstrated by an example. To reduce computational complexity, future research includes investigating a localized version of synthesis that can avoid the explicit computation of global supervisor altogether. Another line of research is a nontrivial extension to multiple inter-agent constraints distributed among agents in a network, where every agent would need to interact with those agents sharing mutually relevant constraints, but not necessarily with all the others in the network. The framework will also need to be further extended to model real-time events, and applied to modeling and designing some real-world systems to demonstrate the practical utility and benefits of the approach.

References

[1] T. W. Malone and K. Crowston, “The interdisciplinary study of coordination,” *ACM Computing Surveys*, vol. 26, no. 1, pp. 87–119, 1994.

[2] S. F. Chew, S. Wang, and M. A. Lawley, “Robust supervisory control for product routings with multiple unreliable resources,” *IEEE Transactions on Automation Science And Engineering*, vol. 6, no. 1, pp. 195–200, 2009.

[3] A. Giua and C. Seatzu, “Modeling and supervisory control of railway networks using petri nets,” *IEEE Transactions on Automation Science And Engineering*, vol. 5, no. 3, pp. 431–445, 2008.

[4] H. Chen, L. Amodeo, F. Chu, and K. Labadi, “Modeling and performance evaluation of supply chains using batch deterministic and stochastic petri nets,” *IEEE Transactions on Automation Science And Engineering*, vol. 2, no. 2, pp. 132–144, 2005.

[5] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordóñez, and M. Tambe, “Iris - a tool for strategic security allocation in transportation networks,” in *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multi-Agent Systems – Industrial Track (AAMAS’09)*, Budapest, Hungary, May 2009, pp. 37–44.

[6] K. T. Seow, C. Ma, and M. Yokoo, “Multiagent planning as control synthesis,” in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS’04)*, Columbia University, New York City, USA, July 2004, pp. 972–979.

[7] K. T. Seow, M. T. Pham, C. Ma, and M. Yokoo, “Coordination planning: Applying control synthesis methods for a class of distributed agents,” *IEEE Transactions on Control Systems Technology*, vol. 17, no. 2, pp. 405–415, 2009.

[8] M. Montemerlo, S. Thrun, H. Dahlkamp, D. Stavens, and S. Strohband, “Winning the darpa grand challenge with an ai robot,” in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI’06)*, Boston, Massachusetts, USA, July 2006, pp. 982–987.

[9] M. Yokoo, *Distributed Constraint Satisfaction : Foundations of Cooperation in Multi-Agent Systems*. Springer-Verlag, Heidelberg, Germany, 2000, Springer Series on Agent Technology.

[10] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara, “The distributed constraint satisfaction problem : Formalization and algorithms,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 5, pp. 673–685, 1998.

[11] B. An, V. Lesser, D. Irwin, and M. Zink, “Automated negotiation with decommitment for dynamic resource allocation in cloud computing,” in *Proceedings of the 9th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS’2010)*, Toronto, Canada, May 2010, pp. 981–988.

[12] P. J. Ramadge and W. M. Wonham, “Supervisory control of a class of discrete event processes,” *SIAM Journal of Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[13] A. Haji-Valizadeh and K. A. Loparo, “Minimizing the cardinality of an events set for supervisors of discrete-event dynamical systems,” *IEEE Transactions on Automatic Control*, vol. 41, no. 11, pp. 1579–1593, 1996.

[14] W. M. Wonham, *Notes on Control of Discrete-Event Systems*. Systems Control Group, University of Toronto, Canada, 2008, http://www.control.toronto.edu/cgi-bin/dldes.cgi.

[15] F. Lin and W. M. Wonham, “On observability of discrete event systems,” *Information Sciences*, vol. 44, no. 3, pp. 173–198, 1988.

[16] W. M. Wonham, *Control Design Software: TCT*. Developed by Systems Control Group, University of Toronto, Canada, Updated 1st July 2008, http://www.control.toronto.edu/cgi-bin/dlxptct.cgi.

[17] M. T. Pham and K. T. Seow, “Towards synthesizing optimal coordination modules for distributed agents,” in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI’08)*, Chicago, Illinois, USA, July 2008, pp. 1479–1480.

[18] K. R. Rohloff, S. Khuller, and G. Kortsarz, “Approximating the minimal sensor selection for supervisory control,” *Discrete Event Dynamic Systems: Theory and Applications*, vol. 16, no. 1, pp. 143–170, 2006.

[19] T.-S. Yoo and S. Lafortune, “NP-completeness of sensor selection problems arising in partially observed discrete-event systems,” *IEEE Transactions on Automatic Control*, vol. 47, no. 9, pp. 1495–1499, 2002.

[20] R. Su and W. M. Wonham, “Supervisor reduction for discrete-event systems,” *Discrete Event Dynamic Systems : Theory and Applications*, vol. 14, no. 1, pp. 31–53, 2004.

[21] P. Gohari and W. M. Wonham, “On the complexity of supervisory control design in the rw framework,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 30, no. 5, pp. 643–652, 2000.

[22] C. Ma and W. M. Wonham, *Nonblocking Supervisory Control of State Tree Structures. Lecture Notes in Control and Information Sciences, Vol 317*. Springer-Verlag, New York, 2005.

[23] L. Feng and W. M. Wonham, “Computationally efficient supervisor design: Abstraction and modularity,” in *Proceedings of the Eighth International Workshop on Discrete Event Systems (WODES’06)*, New York, USA, July 2006, pp. 3–8.

[24] F. Lin and W. M. Wonham, “Decentralized control and coordination of discrete event systems with partial observation,” *IEEE Transactions on Automatic Control*, vol. 35, no. 12, pp. 1330–1337, 1990.

[25] K. R. Rohloff and J. H. van Schuppen, “Approximating minimal communicated event sets for decentralized supervisory control,” in *IFAC World Congress 2005, http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac2005/Fullpapers/02992.pdf.*

[26] K. Rudie and W. M. Wonham, “Think globally, act locally : Decentralized supervisory control,” *IEEE Transactions on Automatic Control*, vol. 37, no. 11, pp. 1692–1708, 1992.

[27] K. T. Seow, “Integrating temporal logic as a state-based specification language for discrete-event control design in finite automata,” *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 3, pp. 451–464, 2007.

[28] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 2005.

[29] J. Marecki, T. Gupta, P. Varakantham, M. Tambe, and M. Yokoo, “Not all agents are equal: Scaling up distributed pomdps for agent networks,” in *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS’08)*, Estoril, Portugal, May 2008, pp. 485–492.

[30] C. V. Goldman and S. Zilberstein, “Decentralized control of cooperative systems: Categorization and complexity analysis,” *Journal of Artificial Intelligence Research*, vol. 22, pp. 143–174, 2004.

[31] X.-R. Cao, *Stochastic Learning and Optimization: A Sensitivity-based Approach*. Springer, 2007.

[32] J. S. Rosenschein and G. Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press Cambridge, Mass, 1994.

[33] Y. Shoham and M. Tennenholtz, “On social laws for artificial agent societies: Off-line design,” *Artificial Intelligence*, vol. 73, no. 1-2, pp. 231–252, 1995.

[34] ——, “On the emergence of social conventions: modeling, analysis, and simulations,” *Artificial Intelligence*, vol. 94, no. 1, pp. 139–166, 1997.

[35] K. C. Wong and J. H. van Schuppen, “Decentralized supervisory control of discrete-event systems with communication,” in *Proceedings of the Third International Workshop on Discrete Event Systems (WODES’96)*, Edinburgh, U.K, August 1996, pp. 284–289.

[36] G. Barrett and S. Lafortune, “Decentralized supervisory control with communicating controllers,” *IEEE Transactions on Automatic Control*, vol. 45, no. 9, pp. 1620-1638, 2000.

[37] K. Rudie, S. Lafortune, and F. Lin, “Minimal communication in a distributed discrete-event system,” *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 957–975, 2003.

[38] ——, “Minimal communication for essential transitions in a distributed discrete-event system,” *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1495–1502, 2007.

[39] M.H. de Queiroz and J.E.R. Cury, “Modular control of composed systems,” *Proceedings of the American Control Conference*, Chicago, IL, USA, June 2000, pp. 4051–4055.