Synchronized Task Decomposition for Two Cooperative Agents

Mohammad Karimadini and Hai Lin
Department of Electrical and Computer Engineering
National University of Singapore
Singapore, Singapore
{karimadini, elelh}@nus.edu.sg

***Abstract*—One of the most important issues in top-down cooperative control of multi-agent systems is to decompose the global specification in order to design the local supervisors such that the fulfilment of these sub-specifications by each individual agent, results in the satisfaction of the global specification as a team. Given the global desired behavior, represented as an automaton, and the distribution of its events into local plants, the question is whether it is always possible to decompose the task automaton into a finite number of sub-automata such that the parallel composition of sub-automata is bisimilar to the original task automaton, and if not, what are the necessary and sufficient conditions for such decomposability. It is shown that it is not always possible to do so. We then present the necessary and sufficient conditions for decomposability of a given task automaton such that the parallel composition of these local task automata bisimulates the original task automaton. It is found that the task automaton is decomposable if and only if it satisfies some symmetry properties, representing independence of the order and the choice of private events from different local event sets, and some properties on the interleaving of strings that share the same first appearing common event. This result will help to design the local controllers from the global logical specification, to be used in the top-down cooperative control of distributed systems.**

***Index Terms*—Task automaton decomposition, cooperative control, multi-agent systems.**

I. INTRODUCTION

Multi-agent system has rapidly emerged as a rapidly growing area and attracted the support of researches and practitioners in many fields such as manufacturing systems, distributed robotic systems, coordinated surveillance, reconnaissance, distributed defence systems, underwater or space exploration, assembling and transportation and rapid emergency response and rescue systems [1]. The cooperative control of a large number of dynamical agents is still in its infancy and possesses significant theoretical and technical challenges that may fall beyond the conventional methodologies [2], [3]. Each agent is usually equipped with certain, but very limited, degrees of sensing, processing, communication, and maneuvering capabilities. However, a large collection of these elementary systems, as a whole, could display remarkable capabilities and exhibit highly complex collective behaviors [4]. Much work has been done on the formation stabilization and consensus seeking [5]-[8]. Approaches like graph Laplacians for the associated neighborhood graphs [7], artificial potential functions [9], [10], navigation functions for distributed formation stabilization with collision avoidance constraints [11]-[13], and optimization based path planning [14], [15], [16], parallel processing [17], [18], bottom-up task sharing and planning [19], [20] have been developed in the literature. Another popular approach for multi-agent system design is through biomimicry and draws inspirations from the swarming behaviors of biological systems, such as colonies of ants, hives of bees, flocks of birds, and schools of fishes [21], [22], [23]. Simulation and empirical studies have shown that complicated collective behaviors can be emerged from a large collection of simple mobile robots via simple intuitive local interaction rules [24]-[27]. These studies generate increasing excitements, and the past decade has seen significant research activities in this area.

Most of efforts so far have been focused on understanding how and what global behavior can be generated from simple local interactions [23], [28]. A more important issue, however, is how to explicitly design these local interaction rules such that certain desirable global behaviors can be achieved by the team of cooperative agents [29]. The desired global specification could be very complicated, and the design goes beyond the traditional path planning, output regulation, or formation control [2], [3], [30], [31]. On the other hand, the bottom-up philosophy in swarming robotics through biomimetic may fail to guarantee the correctness by design, since there still exist lacks in clear understanding on how to change these rules to fulfill or avoid a specific global behavior. This strongly motivates us to develop a new and efficient approach that can design the local interaction rules and control laws for agents directly, and the desired global behaviors can be guaranteed by design. In particular, we aim to develop a formal design method for distributed coordination and control of multi-agent systems so as to make sure that they, as a group, can achieve the specified requirements, collectively.

To achieve such an ambitious research goal, we propose a "divide-and-conquer" approach. The core idea is to decompose a global specification into sub-specifications for individual agents, and then design local interaction rules and control laws for each agent to satisfy these local specifications, respectively. The decomposition should be done in such a way that the global behavior is achieved provided that all these sub-specifications are held true by individual agents. Hence, the global specification should be satisfied by design. The first difficulty for this approach is whether it is always possible to decompose a given global specification. If not,

978-1-4244-6506-4/10/$26.00

what is the necessary and sufficient condition for the proposed decomposability? This paper aims to answer these questions.

These global specifications often refer to the logical and temporal combinations of events describing the behavior and coordination among the agents, and can be represented in linear temporal logic (LTL) [32]. An advantage of using LTL is its structural resemblance to natural languages, and an LTL formula can be converted to a $B\ddot{u}chi$ automaton [33]. Therefore, we assume here that the global specification is given as an automaton. Accordingly, we will focus on the logical behavior [34] of a multi-agent system and model it as a parallel distributed systems (the parallel composition of local plant automata) [35]. Here, each agent is represented by an automaton and has a namely local event set, which may contain both private events and common events.

The main issue investigated in this paper is that given a global task automaton and given the distribution of the global event set into local event sets, how to decompose the global task automaton into sub-automata with respect to agents' event sets such that the proposed divide-and-conquer approach can be performed. A reasonable guess is to use natural projections with respect to an agent's event set. Namely, the agent will ignore the transitions marked by the events that are not in its event set, i.e., blind to these moves. The obtained automaton will be a sub-automaton by deleting all these moves triggered by blind events of the agent. By a simple counter example, we have shown that it is not always possible to decompose an automaton A into sub-automata A_i by natural projection, where the parallel composition of these sub-automata A_i is bisimilar to the automaton A. We have then proposed the necessary and sufficient conditions for automaton decomposition.

Bisimulation synthesis modulo for a global automaton has been also addressed in [36] by introducing a necessary and sufficient for automaton decomposition based on language product of the automaton and determinism of its bisimulation quotient. Obtaining the bisimulation quotient, however, is generally a difficult task. We have proposed a necessary and sufficient condition that characterizes the decomposability based on common and private events. Another work in computer science literature, on automaton decomposition is [37], that has characterized the asynchronous automata that are isomorphic to parallel composition of some automata, and then it has built the decomposition for such systems, based on the maximal cliques of the dependence graph [37]. Their characterization of independency relies on a so-called forward diamond (FD) and independent diamond (ID) rules, representing the intuitive notion of independent order and independent choice of independent events (private events from different local event sets [36]). Our characterization of independent events are more relax than the charactrization of asynchronous automata in [37], allowing to obtain the decomposition in the sense of bisimulation rather than isomorphism (see Remark 1).

The rest of the paper is organized as follows. Preliminary lemmas, notations, definitions and problem formulation are represented in Section II. Section III introduces the necessary and sufficient condition for decomposition of an automaton with respect to parallel composition and two local event sets. The decomposability conditions have been elaborated in this section through illustrative examples. Finally, the paper concludes with remarks and discussions in Section IV.

II. PROBLEM FORMULATION

We first recall the definition of an automaton [38].

Definition 1: (Automaton) An automaton is a tuple $A = (Q, q_0, E, \delta)$ consisting of

- a set of states Q;
- the initial state $q_0 \in Q$;
- a set of events, E, that cause transitions between the states, and
- a transition relation $\delta \subseteq Q \times E \times Q$ such that $(q, e, q') \in \delta$ if and only if $\delta(q, e) = q'$ (or $q \xrightarrow{e} q'$).

The transition relation can be extended to a finite string of events, $S \in E^*$, where E^* stands for $Kleene - Closure$ of E (the set of all finite strings over elements of E), as follows $\delta(q, \varepsilon) = q$, and $\delta(q, Se) = \delta(\delta(q, S), e)$ for $S \in E^*$ and $e \in E$. We focus on deterministic task automata that are simpler to be characterized, and cover a wide class of specifications. The qualitative behavior of a deterministic system is described by the set of all possible sequences of events starting from initial states. Each such a sequence is called a string, and a collection of strings represents the language generated by the automaton, denoted by $L(A)$. The existence of a transition over string $S \in E^*$ from a state $q \in Q$, is denoted by $\delta(q, S)!$, and considering a language L, by $\delta(q, L)!$ we mean $\forall \omega \in L : \delta(q, \omega)!$.

To describe the decomposability condition in the main result and during the proofs, we define successive event pair and adjacent event pair as the following two definitions.

Definition 2: (Successive event pair) Two events e_1 and e_2 are called successive events if $\exists q \in Q : \delta(q, e_1)! \wedge \delta(\delta(q, e_1), e_2)!$ or $\delta(q, e_2)! \wedge \delta(\delta(q, e_2), e_1)!$.

Definition 3: (Adjacent event pair) Two events e_1 and e_2 are called adjacent events if $\exists q \in Q : \delta(q, e_1)! \wedge \delta(q, e_2)!$.

To compare the task automaton and its decomposed automata, we use the simulation relation [39], defined as follows.

Definition 4: (Simulation) Let two automata $A_i = \big(Q_i, q_i^0, E, \delta_i\big)$, $i = 1, 2$. A relation $R \subseteq Q_1 \times Q_2$ is said to be a simulation relation from A_1 to A_2 (denoted by $A_1 \prec A_2$) if

1) $\big(q_1^0, q_2^0\big) \in R$
2) $\forall\, (q_1, q_2) \in R, \delta_1(q_1, e) = q_1'$, then $\exists q_2' \in Q_2$ such that $\delta_2(q_2, e) = q_2', (q_1', q_2') \in R$.

The mutual symmetric similarity between A_1 and A_2 is called bisimilarity relation and is denoted by $A_1 \cong A_2$. Two automata are (bi)similar when the (bi)simulation relation is defined over all $(Q_1 \times Q_2)$ Q_1, for all $e \in E$.

In this paper, we assume that the task automaton A_S and the sets of local events E_i are all given. It is further assumed that A is deterministic automaton while its event set E is obtained by the union of local event sets, i.e., $E = \cup_i E_i$. The problem is to check whether the task automaton A_S can be decomposed into sub-automata A_{S_i} on the local event sets

E_i, respectively, such that the collection of these sub-automata A_{S_i} is equivalent somehow to A_S when put them together. The equivalence is in the sense of bisimilarity as defined above, while the clustering process for these sub-automata A_{S_i} could be in the usual sense of parallel composition as defined below. Parallel composition is used to model the interactions between automata and represent the logical behavior of multi-agent systems. Parallel composition is formally defined as

Definition 5: (Parallel Composition) [38] Let $A_i = \left(Q_i, q_i^0, E_i, \delta_i\right), i = 1, 2$ be automata. The parallel composition (synchronous composition) of A_1 and A_2 is the automaton $A_1||A_2 = (Q, q_0, E, \delta)$, defined as

- $Q = Q_1 \times Q_2$;
- $q_0 = (q_1^0 \times q_2^0)$;
- $E = E_1 \cup E_2$;
- $\forall (q_1, q_2) \in Q, e \in E : \delta((q_1, q_2), e) =$
$$\begin{cases} \left(\delta_1(q_1, e), \delta_2(q_2, e)\right), & \text{if } \begin{cases} \delta_1(q_1, e)!, \delta_2(q_2, e)! \\ e \in E_1 \cap E_2 \end{cases}; \\ \left(\delta_1(q_1, e), q_2\right), & \text{if } \delta_1(q_1, e)!, e \in E_1 \backslash E_2; \\ \left(q_1, \delta_2(q_2, e)\right), & \text{if } \delta_2(q_2, e)!, e \in E_2 \backslash E_1; \\ undefined, & \text{otherwise} \end{cases}$$

A reasonable guess for task automaton decomposition is to use natural projections with respect to agents' event set. Natural projection over strings is denoted by $p_{E_i} = p_i : E^* \rightarrow E_i^*$, takes a string from the event set E and eliminates events in it that do not belong to the event set $E_i \subseteq E$. The natural projection is formally defined on the strings as

Definition 6: (Natural Projection on String) Consider a global event set E and its local event sets E_i, $i = 1, 2$, with $E = E_1 \cup E_2$. Then, the natural projection $p_i : E^* \rightarrow E_i^*$ is inductively defined as

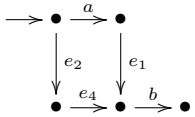
$p_i(\varepsilon) = \varepsilon$;

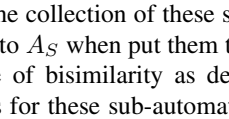
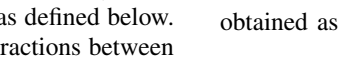
$$\forall S \in E^*, e \in E : p_i(Se) = \begin{cases} p_i(S)e & \text{if } e \in E_i; \\ p_i(S) & \text{otherwise.} \end{cases}$$

The natural projection is also defined on automata as $P_i(A_S) : A_S \rightarrow A_{S_i}$, where, A_{S_i} are obtained from A_S by replacing its events that are belonged to $E\backslash E_i$ by ε-moves, and then, merging the ε-related states. The natural projection is formally defined on an automaton as follows.

Definition 7: (Natural Projection on Automaton) [37] Consider an automaton $A_S = (Q, q_0, E, \delta)$ and local event sets E_i, $i = 1, 2$, with $E = E_1 \cup E_2$. Then, $P_i(A_S) = (Q_i = Q_{/\sim E_i}, q_i^0 = q_{0/\sim E_i}^0, E_i, \delta_i)$, with $\delta_i([q]_{E_i}, e) = [q']_{E_i}$ if there are states q_1 and q_1' such that $q_1 \sim_{E_i} q$, $q_1' \sim_{E_i} q'$, and $\delta(q_1, e) = q_1'$. Here, $[q]_{E_i}$ denotes the equivalence class of the state q defined on $\sim E_i$, where, the relation $\sim E_i$ is the least equivalence relation on the set Q of states such that $\delta(q, e) = q' \wedge e \notin E_i \Rightarrow q \sim_{E_i} q'$.

Following example elaborates the concept of natural projection on a given automaton.

Example 1: Consider an automaton A_S:  with the event set $E = E_1 \cup E_2$ and local event sets $E_1 = \{a, b, e_1\}$, $E_2 = \{a, b, e_2, e_4\}$.

The natural projection of A_S into E_1 is obtained as $P_1(A_S)$:  by replacing $e_2, e_4 \in E\backslash E_1$ with ε and merge the ε-related states. Similarly, the projection $P_2(A_S)$ is obtained as $P_2(A_S)$:  .

To investigate the interactions of transitions between two automata, particularly in $P_1(A_S)$ and $P_2(A_S)$, the interleaving of strings is defined as follows.

Definition 8: Consider two sequences $q_1 \stackrel{e_1}{\rightarrow} q_2 \stackrel{e_2}{\rightarrow} ... \stackrel{e_n}{\rightarrow} q_n$ and $q_1' \stackrel{e_1'}{\rightarrow} q_2' \stackrel{e_2'}{\rightarrow} ... \stackrel{e_m'}{\rightarrow} q_n'$, the interleaving of their corresponding strings, $S = e_1 e_2 ... e_n$ and $S' = e_1' e_2' ... e_m'$, is denoted by $S|S'$, and defined as $S|S' = L\{PA(q_1, S)||PA'(q_1', S')\}$, where, $PA(q_1, S) = (\{q_1, ..., q_n\}, \{q_1\}, \{e_1, ..., e_n\}, \delta_{PA})$ with $\delta_{PA}(q_i, e_i) = q_{i+1}$, $i = 1, ..., n-1$, and $PA'(q_1', S')$ is defined, similarly.

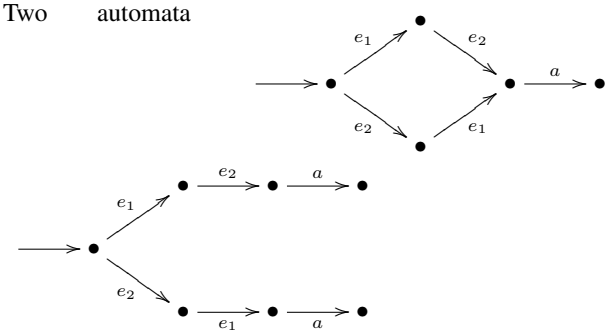
Based on these definitions we may now formally define the decomposability of an automaton with respect to parallel composition and natural projections as follows.

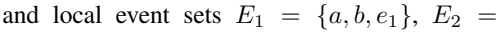
Definition 9: (Automaton decomposability) A task automaton A_S with the event set E and local event sets E_i, $i = 1, 2$, $E = E_1 \cup E_2$, is said to be decomposable with respect to parallel composition and natural projections $P_i : A_S \rightarrow P_i(A_S)$, $i = 1, 2$, when $P_1(A_S)||P_2(A_S) \cong A_S$.

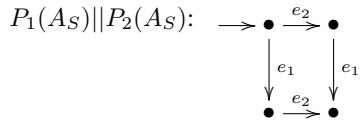
To tackle the task decomposition using projection scheme, it is crucial to know wether any given task automata is decomposable, and if not, what is the decomposability condition. The first question refers to the solvability of decomposition problem as follows.

Problem 1: Given a deterministic task automaton A_S and local event sets E_i, $i = 1, 2$, is it always possible to decompose A_S with respect to parallel composition and natural projections P_1, P_2?

To answer this question, we examine following two examples.

Example 2: Consider the task automaton A_S and its local event sets in Example 1. This automaton is decomposable with respect to parallel composition and natural projections, since $A_S \cong P_1(A_S)||P_2(A_S)$, leading to $L(A_S) = L(P_1(A_S)||P_2(A_S)) = \{\varepsilon, a, ae_1, ae_1b, e_2, e_2e_4, e_2e_4b\}$. Two automata and  with $E = E_1 \cup E_2$, $E_1 = \{a, e_1\}$, $E_2 = \{a, e_2\}$ are other examples of decomposable automata.

Example 3: Consider an automaton A_S:  , with local event sets $E_1 = \{e_1\}$

and $E_2 = \{e_2\}$. The parallel composition of $P_1(A_S)$: $\longrightarrow \bullet \xrightarrow{e_1} \bullet$ and $P_2(A_S)$: $\longrightarrow \bullet \xrightarrow{e_2} \bullet$ is $P_1(A_S)||P_2(A_S)$:  .

One can observe that $A_S \prec P_1(A_S)||P_2(A_S)$ but $P_1(A_S)||P_2(A_S) \not\prec A_S$ leading to $L(A_S) = \{\varepsilon, e_1, e_1e_2\} \subseteq L(P_1(A_S)||P_2(A_S)) = \{\varepsilon, e_1, e_1e_2, e_2, e_2e_1\}$, but $L(P_1(A_S)||P_2(A_S)) \not\subseteq L(A_S)$. Therefore, A_S is not decomposable with respect to parallel composition and natural projections P_i, $i = 1, 2$.

Example 2 shows examples of decomposable automata. Example 3, on the other hand, illustrates a counter example of an undecomposable automaton. Therefore, not all automata are decomposable with respect to parallel composition and natural projections. Then, a natural follow-up question is what makes an automaton decomposable. It can be formally stated as follows.

Problem 2: Given a deterministic task automaton A_S and local event sets E_i, $i = 1, 2$, what is the necessary and sufficient condition that A_S is decomposable with respect to parallel composition and natural projections $P_i : A_S \rightarrow P_i(A_S)$, $i = 1, 2$, such that $P_1(A_S)||P_2(A_S) \cong A_S$?

III. TASK AUTOMATON DECOMPOSITION

The previous section showed that not all automata are decomposable, i.e., not every automaton A_S bisumulates $P_1(A_S)||P_2(A_S)$. On the other hand, from the definition of natural projection it is found that $P_1(A_S)||P_2(A_S)$ is always decomposable into $P_1(A_S)$ and $P_2(A_S)$. Now, focusing on $P_1(A_S)||P_2(A_S)$, obtained from parallel composition of $P_1(A_S)$ and $P_2(A_S)$, we are interested in finding the conditions for bisimilarity of $P_1(A_S)||P_2(A_S)$ and A_S. This bisimilarity is constructed based on mutual simulations as stated in the following two lemmas. Because of limitation in space, the proof of lemmas are omitted from here and are available in [40]. By the first lemma, it always holds the following simulation relationship.

Lemma 1: Consider any deterministic automaton A_S with event set E, local event sets E_i, and natural projections P_i, $i = 1, 2$. Then $A_S \prec P_1(A_S)||P_2(A_S)$.

This lemma shows that, in general, $P_1(A_S)||P_2(A_S)$ simulates A_S. In order to force the entire closed loop system to respect the global specification, A_S is also required to simulate $P_1(A_S)||P_2(A_S)$, that collectively leads to decomposability of A_S as $A_S \cong P_1(A_S)||P_2(A_S)$.

The similarity of $P_1(A_S)||P_2(A_S)$ to A_S, however, is not always true (See Example 3), and needs some conditions as stated in the following lemma.

Lemma 2: Consider a deterministic automaton $A_S = (Q, q_0, E = E_1 \cup E_2, \delta)$ and natural projections $P_i : A_S \rightarrow P_i(A_S)$, $i = 1, 2$. Then $P_1(A_S)||P_2(A_S) \prec A_S$ if and only if it satisfies the following conditions: $\forall e_1 \in E_1 \backslash E_2, e_2 \in E_2 \backslash E_1, q \in Q$, $S \in E^*$,

- $DC1 : [\delta(q, e_1)! \wedge \delta(q, e_2)!] \vee \delta(q, e_1e_2)! \vee \delta(q, e_2e_1)! \Rightarrow \delta(q, e_1e_2)! \wedge \delta(q, e_2e_1)!$;
- $DC2 : \delta(q, e_1e_2S)! \Leftrightarrow \delta(q, e_2e_1S)!$, and
- $DC3 : \forall S, S' \in E^*$, sharing the same first appearing common event $a \in E_1 \cap E_2$, $S \neq S'$, $q \in Q$: $\delta(q, S)! \wedge \delta(q, S')! \Rightarrow \delta(q, p_1(S)|p_2(S'))! \wedge \delta(q, p_1(S')|p_2(S))!$.

From Lemmas 1 and 2, under the same conditions in Lemma 2, mutual symmetric simulation relations will be obtained between $Q_1 \times Q_2$ in $P_1(A_S)||P_2(A_S)$ and Q in A_S over E, as stated in the following main result. Given local event sets E_1 and E_2, the following theorem gives a necessary and sufficient condition for the decomposability of automaton A_S.

Theorem 1: A deterministic automaton $A_S = (Q, q_0, E = E_1 \cup E_2, \delta)$ is decomposable with respect to parallel composition and natural projections $P_i : A_S \rightarrow P_i(A_S)$, $i = 1, 2$, such that $A_S \cong P_1(A_S)||P_2(A_S)$ if and only if it satisfies the following decomposability conditions (DC): $\forall e_1 \in E_1 \backslash E_2, e_2 \in E_2 \backslash E_1, q \in Q$, $S \in E^*$,

- $DC1 : [\delta(q, e_1)! \wedge \delta(q, e_2)!] \vee \delta(q, e_1e_2)! \vee \delta(q, e_2e_1)! \Rightarrow \delta(q, e_1e_2)! \wedge \delta(q, e_2e_1)!$;
- $DC2 : \delta(q, e_1e_2S)! \Leftrightarrow \delta(q, e_2e_1S)!$, and
- $DC3 : \forall S, S' \in E^*$, sharing the same first appearing common event $a \in E_1 \cap E_2$, $S \neq S'$, $q \in Q$: $\delta(q, S)! \wedge \delta(q, S')! \Rightarrow \delta(q, p_1(S)|p_2(S'))! \wedge \delta(q, p_1(S')|p_2(S))!$.

Remark 1: Intuitively, the decomposability condition $DC1$ means that for any successive or adjacent pair of private events $(e_1, e_2) \in \{(E_1 \backslash E_2, E_2 \backslash E_1), (E_2 \backslash E_1, E_1 \backslash E_2)\}$ (from different private event sets), both orders e_1e_2 and e_2e_1 should be legal from the same state, unless they are mediated by a common string. This concept is illustrated in the following example. Furthermore, e_1e_2 and e_2e_1 are not required to meet at the same state; but due to $DC2$, any string $S \in E^*$ after them should be the same, or in other words, if e_1 and e_2 be necessary conditions for occurrence of a string S, then any order of these two events would be legal for such occurrence (see automata in Example 2). Note that, as a special case, S could be ε.

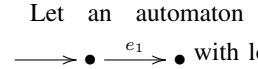
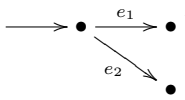
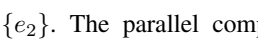
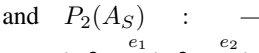
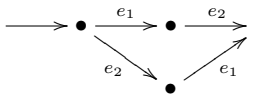
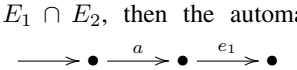
The axioms $DC1$, $DC2$ also like forward diamond (FD) and independent diamond (ID) rules in [37] represent the intuitive notion of independent order and independent choice of independent events (private events from different local event sets), However, $DC1$, $DC2$ differ from FD and ID, by allowing the diamond to not necessarily be closed, i.e., independent events e_1 and e_2 starting from one state are allowed to occur in any order, but according to $DC1$ and $DC2$, the orders may reach to different states. This relaxation allows us to obtain the decomposition in the sense of bisimulation rather than isomorphism. Generally, some automata may satisfy $DC1$ and $DC2$, but not necessarily FD and ID (see for instance the second automaton in Example 2 that does not satisfy FD and ID and is not decomposable in the sense of isomorphism, but it satisfies $DC1$ and $DC2$, and it is decomposable in the sense of bisimulation).

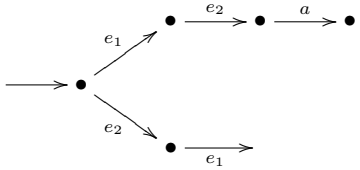
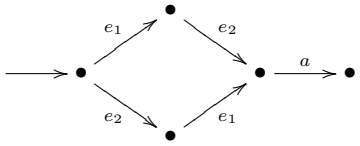
The condition $DC3$ means that if two strings S and S' share the same first appearing common event, then any interleaving of these two strings should be legal in A_S. This requirement is

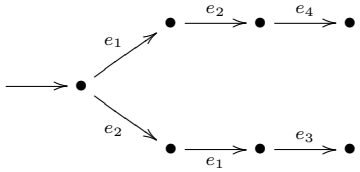
due to synchronization constraint among projections of these strings in $P_1(A_S)$ and $P_2(A_S)$.

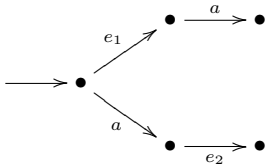
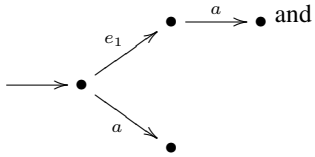
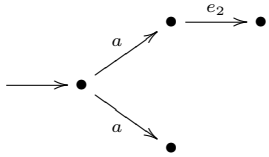
Following three examples illustrate the decomposable and undecomposable automata based on decomposability conditions.

Example 4: This example shows an automaton that satisfies $DC2$ and $DC3$, but not $DC1$, leading to undecomposability.

Let an automaton A_S to be  or  with local event sets $E_1 = \{e_1\}$ and $E_2 = \{e_2\}$. The parallel composition of $P_1(A_S)$:  and $P_2(A_S)$:  is $P_1(A_S)|P_2(A_S)$:  . Therefore, A_S is not decomposable with respect to parallel composition and natural projections P_i, $i = 1, 2$, since two succesive/adjacent events $e_1 \in E_1 \backslash E_2$ and $e_2 \in E_2 \backslash E_1$ do not respect $DC1$. In the first automaton, A_S is not decomposable, due to violation of $DC1$, although it fulfils $DC2$ and $DC3$. One can observe that, if in this example $e_1 \in E_1 \backslash E_2$ and $e_2 \in E_2 \backslash E_1$ were separated by a common event $a \in E_1 \cap E_2$, then the automata  and with local event sets $E_1 = \{e_1, a\}$ and $E_2 = \{e_2, a\}$, were both decomposable.

Example 5: This example shows an automaton which respects $DC1$ and $DC3$, but is undecomposable due to violation of $DC2$. Consider automaton A_S:  with $E_1 = \{a, e_1\}$, $E_2 = \{a, e_2\}$, leading to $P_1(A_S)||P_2(A_s)$:  . The transition $\delta_{||}(z_0, e_2e_1a)!$ in $P_1(A_S)||P_2(A_S)$, but $\neg\delta(q_0, e_2e_1a)!$ in A_S.

Another such example, that satisfies $DC1$ and $DC3$ but not $DC2$ is A_S:  with $E_1 = \{e_1, e_3\}$, $E_2 = \{e_2, e_4\}$.

Example 6: This example illustrates an automaton that satisfies $DC1$ and $DC2$, but is undecomposable as it does not fulfil $DC3$. Consider the automaton  with $E = E_1 \cup E_2$, $E_1 = \{a, e_1\}$, $E_2 = \{a, e_2\}$. From $P_1(A_S)$:  and $P_2(A_S)$:  , it can be seen that $\delta_{||}(z_0, e_1ae_2)!$ in $P_1(A_S)||P_2(A_S)$, but $\neg\delta(q_0, e_1ae_2)!$ in A_S, and hence $A_S \ncong P_1(A_S)||P_2(A_S)$.

IV. CONCLUSION

The paper proposed a formal method for task automaton decomposition, applicable in top-down cooperative control of multi-agent systems. Given a team of two plants whose collective logical behaviors are represented in a parallel distributed system, and a global task automaton, together with the distribution of the global event set, the paper provides a necessary and sufficient condition for decomposability of the task automaton with respect to parallel composition and natural projections into local event sets. Using the associativity property of parallel composition, we are generalizing this result into an arbitrary finite number of agents. This method offers a modular scheme for task decomposition as well as supervisory synthesis so that a new task automaton is decomposed subject to the individual event sets and the new subtask automata are composed with the old ones using parallel composition, without affecting the previous decomposition(s).

This method, however, is restricted to the class of distributed systems that their interactions can be modeled by parallel distributed systems, with a given distribution of the events into local plants. If the task automaton is not decomposable, a future direction could be to make the task automaton decomposable, by modifying the local event sets, i.e., by designing the distribution of the global event set.

V. ACKNOWLEDGMENTS

The financial supports from Singapore Ministry of Educations AcRF Tier 1 funding, TDSI, and TL are gratefully acknowledged.

REFERENCES

[1] P. U. Lima and L. M. Custdio, *Multi-Robot Systems, Book Series Studies in Computational Intelligence*, Book Innovations in Robot, Mobility and Control, Chapter 1, vol. 8, Springer Berlin / Heidelberg, 2005.

[2] P. Tabuada and G. J. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Trans. Automat. Contr.*, vol. 51, no. 12, pp. 1862–1877, 2006.
[3] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins and G. J. Pappas, "Symbolic planning and control of robot motion," *IEEE Robotics and Automation Mag.*, special issue on Grand Challenges of Robotics, vol. 14, no. 1, pp. 61–71, 2007.
[4] V. R. Lesser, "Cooperative multiagent systems: a personal view of the state of the art", *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 133–142, 1999.
[5] N. A. Lynch, *Distributed Algorithm*, Morgan Kaufmann Publishers, 1997.
[6] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transaction on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.
[7] A. Jadbabaie, J. Lin and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transaction on Automatatic Control*, vol. 48, no. 6, pp. 988C-1001, 2003.
[8] H. Tanner, A. Jadbabaie and G. Pappas, "Stable flocking of mobile agents, part II : Dynamic topology," *Proc. of the 42rd IEEE Conference on Decision and Control*, pp. 2016–2021, 2003.
[9] J. H. Reif, H. Wang, "Social potential fields: A distributed behavioral control for autonomous robots," In K. Goldberg, D. Halperin, J. C. Latombe, R. Wilson, (Eds.), *The Algorithmic Foundations of Robotics*, A. K. Peters, Boston, MA, pp. 331–345, 1995.
[10] T. Laue and T. Rfer, "A behavior architecture for autonomous mobile robots based on potential fields," Lecture Notes in Computer Science, RoboCup 2004: Robot Soccer World Cup VIII.
[11] S. G. Loizou and K. J. Kyriakopoulos, "Multirobot navigation functions I," in H. A. P Blom and J. Lygeros (Eds.), *Stochastic Hybrid Systems: Theory and Safety Critical Applications*, Chapter 10, Springer, 2006.
[12] D. V. Dimarogonas, S. G. Loizou and K. J. Kyriakopoulos, "Multirobot navigation functions II: towards decentralization," in H. A. P Blom and J. Lygeros (Eds.) *Stochastic Hybrid Systems: Theory and Safety Critical Applications*, Chapter 11, Springer, 2006.
[13] H. G. Tanner, A. Kumar, "Towards decentralization of multi-robot navigation functions," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 4132– 4137, 18–22 April 2005.
[14] T. Schouwenaars, B. De Moor, E. Feron and J. How, "Mixed integer programming for multi-vehicle path planning," in *European control conference, ECC 2001*, pp. 2603–2608, Sep 2001.
[15] C. Reinl and O. V. Stryk, "Optimal control of multi-vehicle-systems under communication constraints using mixed-integer linear programming," *In Proc. of the 1st international Conf. on Robot Communication and Coordination* (Athens, Greece. ACM International Conference Proceeding Series, vol. 318. IEEE Press, Piscataway, NJ, pp. 1–8, October 15–17, 2007).
[16] Z. Cai and Z. Peng, "Cooperative coevolutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot systems," *Journal of Intelligent and Robotic Systems*, vol. 33, no. 1, Jan. 2002.
[17] N. Carriero, D. Gelernter, "Linda in context," *Communications of the ACM*, vol. 32 , no. 4, pp. 444–458, 1989.
[18] G. S. Almasi, "Overview of parallel processing," *Parallel Computing*, vol. 2, no. 3, pp. 191–203, 1985.
[19] A. Konar and L. C. Jain, *Cognitive Engineering, A Distributed Approach to Machine Intelligence Series: Advanced Information and Knowledge Processing*, Chapter 11, Springer London, 2005.
[20] E. H. Durfee, "Distributed problem solving and planning," In G. Weiss, (Ed.), *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, Cambridge, MA, 1999.
[21] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the travelingsalesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.
[22] E. Bonabeau, M. Dorigo and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford, England: Oxford University Press, 1999.
[23] C. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *Proc. of the 14th annual conference on Computer Graphics*, vol. 21, no. 4, July 1987.
[24] M. G. Hinchey, J. L. Rash, W. F. Truszkowski, C. A. Rouff and R. Sterrit, "Autonomous and autonomic swarms," *In Proc. The 2005 International Conf. on Software Engineering Research and Practice* (SERP'05), CSREA Press, Las Vegas, Nevada, USA, pp. 36–42, 27 June 2005.
[25] C. A. Rouff, W. F. Truszkowski, J. L. Rash and M. G. Hinchey, " A survey of formal methods for intelligent swarms," *Technical Report TM-2005-212779*, NASA Goddard Space Flight Center, Greenbelt, Maryland, 2005.
[26] W. F. Truszkowski, M. G. Hinchey, J. L. Rash and C. A. Rouff, "Autonomous and autonomic systems: A paradigm for future space exploration missions," *IEEE Trans. on Systems*, Man and Cybernetics, Part C, 2006.
[27] M. Kloetzer and C. Belta, "Temporal logic planning and control of robotic swarms by hierarchical abstractions," *IEEE Trans. on Robotics*, vol. 23, no. 2, pp. 30–331, 2007.
[28] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proc. of IEEE International Conference on Neural Networks*, vol. 4, no. 27, pp. 1942–1948, 1995.
[29] V. Crespi, A. Galstyan and K. Lerman, "Top-down vs bottom-up methodologies in multi-agent system design," *Journal: Autonomous Robots*, Publisher: Springer Netherlands, vol. 24 , no. 3, pp. 303–313, April 2008.
[30] P. Tabuada and G. J. Pappas, "From discrete specifications to hybrid control," Decision and Control, *Proc. 42nd IEEE Conference*, vol. 4, no. 9–12, pp. 3366– 3371, CDC, Dec. 2003.
[31] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems From temporal logic specifications," *IEEE Trans. Automat. Contr.*, vol. 53, no.1, pp. 287–297, 2008.
[32] R. Gotzhein, "Temporal logic and applications-a tutorial," presented at *Computer Networks and ISDN Systems*, pp. 203–218, 1992.
[33] P. Wolper, *Constructing Automata from Temporal Logic Formulas: A Tutorial*, Book Series Lecture Notes in Computer Science, Springer Berlin , vol. 2090, 2001, pp. 261–277.
[34] X. D. Koutsoukos, P. J. Antsaklis, J. A. Stiver and M. D. Lemmon, "Supervisory control of hybrid systems," *Proc. of the IEEE*, vol. 88, no. 7, pp. 1026–1049, 2000.
[35] M. Mukund, "From global specifications to distributed implementations," in B. Caillaud, P. Darondeau, L. Lavagno (Eds.), *Synthesis and Control of Discrete Event Systems*, Kluwer, pp. 19–34, 2002.
[36] I. Castellani, M. Mukund, P.S. Thiagarajan, "Synthesizing distributed transition systems from global specification," In: Pandu Rangan, C., Raman, V., Ramanujam, R. (eds.) FSTTCS 1999. LNCS, vol. 1738, pp. 219–231, Springer, Heidelberg, 1999.
[37] R. Morin, "Decompositions of asynchronous systems," In CONCUR 98, LNCS 1466, pp. 549–564, Springer, 1998.
[38] R. Kumar and V. K. Garg, *Modeling and Control of Logical Discrete Event Systems*, Kluwer Academic Publishers, Boston, 1995.
[39] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Springer, U.S.A, 2008.
[40] M. Karimadini and H. Lin, *Synchronized Task Decomposition for Cooperative Multi-agent Systems*, Technical Report: NUS-ACT-10-001-Ver.2, Advanced Control Technology Laboratory, National University of Singapore, [Online]. Available: http://www.ece.nus.edu.sg/stfpage/elelh/publicationlist.html or http://arxiv.org/abs/0911.0231[Accessed April. 28, 2010].