

# Decomposability of Global Tasks for Multi-agent Systems

Mohammad Karimadini and Hai Lin

**Abstract**—Multi-agent system is a rapidly developing research area with strong support from both civilian and military applications. One of the essential problems in multi-agent system research is how to design local interaction rules and coordination principles among agents such that the whole system achieves desired global behaviors. To tackle this problem, a divide-and-conquer approach was proposed in [1], and the basic idea is to decompose the requested global specification into subtasks for individual agents in such a way that the fulfillment of these subtasks by each individual agent should lead to the satisfaction of the global specification. Then, the design reduces to achieving the assigned subtasks for corresponding individual agents. In [1], it was shown that not all global tasks can be decomposed, and a necessary and sufficient condition on the decomposability of a task automaton between two agents was presented. For more than two agents, we then proposed a hierarchical algorithm as a sufficient condition for decomposability. This paper aims to extend the necessary and sufficient decomposability conditions for any arbitrary finite number of cooperative agents. A new necessary and sufficient condition on decomposability of a task automaton is proposed, here. Several examples are provided to illustrate the decomposition scheme and conditions.

## I. INTRODUCTION

Multi-agent system is a rapidly developing cross-disciplinary research area that has been obtaining strong support from both civilian and military applications such as coordinated surveillance, target acquisition, reconnaissance, underwater or space exploration, assembling and transportation and rapid emergency response [2]. It is known that sophisticated collective behaviors can emerge through the cooperation of rudimentary agents with simple local interaction rules. But, most studies in the multi-agent system literature have been focused on bottom-up approaches [3], and mainly through simulation, empirical and heuristic approaches [4]-[7]. The past several years have seen significant research efforts in the theoretical analysis on multi-agent systems using graph theory [8] and symbolic control of swarming systems [9], [10]. However, the cooperative control of multi-agent system is still in its infancy with significant practical and theoretical challenges that are difficult to be formulated and tackled by the traditional methods [9], [11].

One of the essential problems in cooperative control of multi-agent systems is how to design the local interaction rules and coordination principles among agents such that the whole system achieves a desired global specification. To tackle this challenge, in [1], we proposed a divide-and-conquer approach by decomposing the requested global

specification into subtasks for individual agents such that the fulfillment of these subtasks by each individual agent will lead to the satisfaction of the global specification. Then, the design reduces to achieving the assigned subtasks for corresponding individual agents. In order to pursue the idea, several questions need to be answered, such as how to describe the global specification and subtasks in a succinct and formal way, how to decompose the global specification, whether it is always possible to decompose, and if not what is the condition for decomposability.

This paper continues the efforts in our previous work [1] and aims to answer these questions and pace the way towards a formal design method for multi-agent systems. Here, the task for a group of intelligent agents are represented as an automaton due to its expressibility and similarity to our human logical commands [12], [13]. Then, the decomposition of a global task actually becomes the automaton decomposition problem that has been studied in the computer science literature. Roughly speaking, two different classes of problems have been studied so far. The first problem is to design the event distribution so as to make the automaton decomposable, which is typically studied in the context of concurrent systems. For example, [14] characterized the conditions for decomposition of asynchronous automata in the sense of isomorphism based on the maximal cliques of the dependence graph. Their characterization of independence relies on 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). Our decomposability conditions also include two conditions addressing the notion on independence (we will call them *DC1* and *DC2*). However, in *DC1* and *DC2*, although independent events  $e_1$  and  $e_2$  starting from one state are allowed to occur in any order, different occurring orders may lead to different states. This relaxation allows us to obtain the decomposition in the sense of bisimulation rather than isomorphism in [14]. Generally, some automata may satisfy *DC1* and *DC2*, but not necessarily *FD* and *ID*. On the other hand, the second problem assumes that the distribution of the global event set is given and the goal is to find conditions on the automaton such that it is decomposable. This is usually called synthesis modulo problem [13], and bisimulation synthesis modulo for a global automaton was addressed in [15] by introducing a necessary and sufficient condition 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. This motivates us to develop a new

M. Karimadini and H. Lin are both from the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. Corresponding author, H. Lin [elelh@nus.edu.sg](mailto:elelh@nus.edu.sg)

necessary and sufficient condition and consider more general cases, which may occur in multi-agent systems.

Following [1], we assume that the global desired behavior can be represented as a deterministic automaton, whose event set is the union of the collection of local event sets for each agent. Under this assumption, the local task automaton for an individual agent can be obtained through natural projection of the global task automaton into its corresponding local event set. Unfortunately, it was shown in [1] that not all global task automaton can be decomposed in this way, which means that the composition of the obtained sub-task automata is not equivalent to the global task automaton. Formally, the composition is through classical parallel composition, while the equivalence is in the sense of bisimulation. Furthermore, in [1] a necessary and sufficient condition on the decomposability of a task automaton between two agents was proposed. For more than two agents a hierarchical algorithm was presented that was shown to be only sufficient condition. In this paper, we aim to further extend the decomposability result to any arbitrary finite number of cooperative agents, as a necessary and sufficient condition.

The rest of the paper is organized as follows. Preliminary results, 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 arbitrary finite number of event sets. Finally, the paper concludes in Section IV with remarks and discussions.

## II. PROBLEM FORMULATION

We first recall the definition of an automaton [16].

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

- a set of states  $Q$ ;
- the initial state  $q_0 \subseteq Q$ ;
- a set of events  $E$  that causes 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  $\delta(q, \varepsilon) = q$ , 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 the initial state. 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 a string  $S \in E^*$  from a state  $q \in Q$  is denoted by  $\delta(q, S)!$ . Considering a language  $L$ , by  $\delta(q, L)!$  we mean that  $\forall \omega \in L : \delta(q, \omega)!$ .

Next, the successive event pair and adjacent event pair for an automaton are defined as follows.

*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 the composition of its decomposed automata, we use the simulation relation [12], defined as follows.

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

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

A mutual symmetric similarity between  $A_1$  and  $A_2$  is called bisimilarity and denoted as  $A_1 \cong A_2$ . 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 a deterministic automaton while its event set  $E$  is given as 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 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 [16] is a common way to model the interactions between automata, and is employed here to represent the global behavior of a team of cooperative multi-agents.

*Definition 5: (Parallel Composition)* [16] Let  $A_i = (Q_i, q_i^0, E_i, \delta_i)$ ,  $i = 1, 2$  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

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

The parallel composition of  $A_i$ ,  $i = 1, 2, \dots, n$  is called parallel distributed system, and is defined based on the associativity property of parallel composition [12] as  $\parallel_{i=1}^n A_i = A_1 || \dots || A_n = A_1 || (A_2 || (\dots || (A_{n-1} || A_n)))$ .

Next, let us recall the operation of natural projection that will be used later to obtain the decomposed subtask automata.

*Definition 6: (Natural Projection on String)* Consider a global event set  $E$  and its local event sets  $E_i$ ,  $i = 1, 2, \dots, n$ ,

with  $E = \bigcup_{i=1}^n E_i$ . Then, the natural projection  $p_i : E^* \rightarrow E_i^*$  is inductively defined as

$$p_i(\varepsilon) = \varepsilon;$$

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

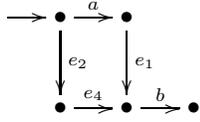
Here  $\text{loc}(e) = \{i | e \in E_i\}$ .

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  $E \setminus E_i$  by  $\varepsilon$ -moves, and then, merging the  $\varepsilon$ -related states. The natural projection is formally defined on an automaton as follows.

**Definition 7:** (Natural Projection on Automaton) Consider an automaton  $A_S = (Q, q_0, E, \delta)$  and local event sets  $E_i$ ,  $i = 1, 2, \dots, n$ , with  $E = \bigcup_{i=1}^n E_i$ . Then,  $P_i(A_S) = (Q_i = Q / \sim_{E_i}, [q_0]_{E_i}, E_i, \delta_i)$ , 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,  $[q]_{E_i} = [q]_i$  denotes the equivalence class of  $q$  defined on  $\sim_{E_i}$ , where, the relation  $\sim_{E_i}$  is the least equivalence relation on the set  $Q$  of states such that  $\delta(q, e) = q' \wedge i \notin \text{loc}(e) \Rightarrow q \sim_{E_i} q'$ .

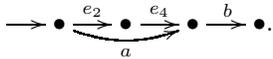
The following example elaborates the concept of natural projection on a given automaton.

*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 projections of  $A_S$  into  $E_1$  is obtained as  $P_1(A_S)$ :  $\bullet \xleftarrow{b} \bullet \xleftarrow[e_1]{e_4} \bullet$  by replacing

$e_2, e_4 \in E \setminus E_1$  with  $\varepsilon$  and merging the  $\varepsilon$ -related states. Similarly, the projection  $P_2(A_S)$  is obtained as  $P_2(A_S)$ :



To investigate the interactions of transitions in 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  $q_1 \xrightarrow{e_1} q_2 \xrightarrow{e_2} \dots \xrightarrow{e_n} q_n$  and  $q'_1 \xrightarrow{e'_1} q'_2 \xrightarrow{e'_2} \dots \xrightarrow{e'_m} q'_m$ , the interleaving of their corresponding strings,  $S = e_1 e_2 \dots e_n$  and  $S' = e'_1 e'_2 \dots e'_m$ , is denoted by  $S|S'$ , and defined as  $S|S' = L\{PA(q_1, S) || PA(q'_1, S')\}$ , where,  $PA(q_1, S) = (\{q_1, \dots, q_n\}, \{q_1\}, \{e_1, \dots, e_n\}, \delta_{PA})$  with  $\delta_{PA}(q_i, e_i) = q_{i+1}$ ,  $i = 1, \dots, n-1$ , and  $\delta_{PA'}$  is defined in a similar way.

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  $E_i$ ,  $i = 1, \dots, n$ ,  $E = \bigcup_{i=1}^n E_i$ , is said to be decomposable with respect to parallel composition and natural projections if  $\prod_{i=1}^n P_i(A_S) \cong A_S$ .

It is easy to show by simple counter examples (see Examples 2 - 5) that 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 1:** Given a deterministic task automaton  $A_S$  and local event sets  $E_i$ ,  $i = 1, \dots, n$ , what is the necessary and sufficient condition that  $A_S$  is decomposable with respect to parallel composition and natural projections  $P_i : A_S \rightarrow P_i(A_S)$ ,  $i = 1, \dots, n$ , such that  $\prod_{i=1}^n P_i(A_S) \cong A_S$ ?

### III. DECOMPOSABILITY OF TASK AUTOMATON

#### A. Decomposability Condition for 2 agents

First, let us recall the main result in [1] for the decomposability of an automaton  $A_S$  for two cooperative agents, to be used as a basis for the more general case of an arbitrary finite number of agents.

**Lemma 1:** (Theorem 1 in [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  $A_S \cong P_1(A_S) || P_2(A_S) = (Z, z_0, E, \delta_{||})$  if and only if it satisfies the following decomposability conditions (DC):  $\forall e_1 \in E_1 \setminus E_2, e_2 \in E_2 \setminus E_1, q \in Q, S \in E^*$ ,

- *DC1:*  $[\delta(q, e_1)! \wedge \delta(q, e_2)!] \Rightarrow [\delta(q, e_1 e_2)! \wedge \delta(q, e_2 e_1)!]$ ;
- *DC2:*  $\delta(q, e_1 e_2 S)! \Leftrightarrow \delta(q, e_2 e_1 S)!$ , and
- *DC3:*  $\forall S, S' \in E^*$ , sharing the same first appearing common event  $a \in E_1 \cap E_2$ ,  $S \neq S'$ ,  $q \in Q$ :  $\delta(q, S)! \wedge \delta(q, S')! \Rightarrow \delta(q, p_1(S) | p_2(S'))! \wedge \delta(q, p_1(S') | p_2(S))!$ ;
- *DC4:*  $\forall i \in \{1, 2\}$ ,  $x, x_1, x_2 \in Q_i$ ,  $x_1 \neq x_2$ ,  $e \in E_i$ ,  $t \in E_i^*$ ,  $\delta_i(x, e) = x_1$ ,  $\delta_i(x, e) = x_2$ :  $\delta_i(x_1, t)! \Leftrightarrow \delta_i(x_2, t)!$ .

**Remark 1:** Intuitively, the decomposability condition *DC1* means that for any adjacent pair of private events  $(e_1, e_2) \in \{(E_1 \setminus E_2, E_2 \setminus E_1), (E_2 \setminus E_1, E_1 \setminus E_2)\}$  (from different private event sets), both orders  $e_1 e_2$  and  $e_2 e_1$  should be legal from the same state, unless they are mediated by a common string.

While *DC1* addresses the decision on adjacent events, *DC2* deals with the decision on the order of successive events and states that if any order of a pair of adjacent private events  $(e_1, e_2) \in \{(E_1 \setminus E_2, E_2 \setminus E_1), (E_2 \setminus E_1, E_1 \setminus E_2)\}$  (from different private event sets) is necessary condition of occurrence of any string  $S \in E^*$ , then the other order also should be legal for such occurrence (see Example 3). Note that, as a special case,  $S$  could be  $\varepsilon$ .

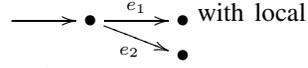
The condition *DC3* means that if two strings  $S$  and  $S'$  share the same first appearing common event, then any interleaving of these two strings should be legal in  $A_S$ . This requirement is due to synchronization of projections of these strings in  $P_1(A_S)$  and  $P_2(A_S)$ .

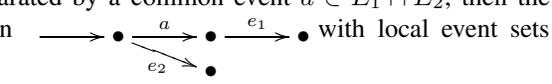
The last condition *DC4*, ensures the symmetry of mutual simulation relations between  $A_S$  and  $P_1(A_S) || P_2(A_S)$ . Given the determinism of  $A_S$ , this symmetry is guaranteed when each local task automaton bisimulates a deterministic

automaton, leading to the existence a deterministic automaton that is bisimilar to  $P_1(A_S)||P_2(A_S)$ . If the simulation relations are not symmetric, then some of the sequences that are allowed in  $A_S$  will be disabled in  $P_1(A_S)||P_2(A_S)$ .

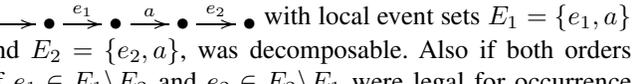
The following several examples are reviewed from [1] to illustrate the decomposable and undecomposable automata based on the decomposability conditions in Lemma 1.

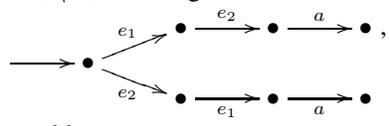
*Example 2:* This example shows an automaton that satisfies *DC2*, *DC3* and *DC4*, but not *DC1*, leading to undecomposability.

Let an automaton  $A_S$  to be  with local event sets  $E_1 = \{e_1\}$  and  $E_2 = \{e_2\}$ . The parallel composition of  $P_1(A_S) : \rightarrow \bullet \xrightarrow{e_1} \bullet$  and  $P_2(A_S) : \rightarrow \bullet \xrightarrow{e_2} \bullet$  is  $P_1(A_S)||P_2(A_S) : \rightarrow \bullet \xrightarrow{e_1} \bullet \xrightarrow{e_2} \bullet$ . Therefore,

$A_S$  is not decomposable, since two adjacent events  $e_1 \in E_1 \setminus E_2$  and  $e_2 \in E_2 \setminus E_1$  do not respect *DC1*. One can observe that, if in this example  $e_1 \in E_1 \setminus E_2$  and  $e_2 \in E_2 \setminus E_1$  were separated by a common event  $a \in E_1 \cap E_2$ , then the automaton  with local event sets  $E_1 = \{e_1, a\}$  and  $E_2 = \{e_2, a\}$ , was decomposable.

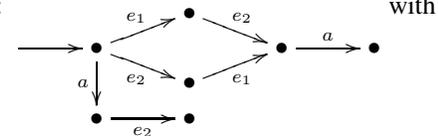
*Example 3:* This example shows an automaton which respects *DC1*, *DC3* and *DC4*, but is undecomposable due to violation of *DC2*. Consider automata  $A_S : \rightarrow \bullet \xrightarrow{e_1} \bullet \xrightarrow{e_2} \bullet \xrightarrow{a} \bullet$  with  $E_1 = \{a, e_1\}$ ,  $E_2 = \{a, e_2\}$ , leading to  $P_1(A_S)||P_2(A_S) : \rightarrow \bullet \xrightarrow{e_1} \bullet \xrightarrow{e_2} \bullet \xrightarrow{a} \bullet$ . The transition

$\delta_{||}(z_0, e_2e_1a)!$  in  $P_1(A_S)||P_2(A_S)$ , but  $-\delta(q_0, e_2e_1a)!$  in  $A_S$ . If  $e_1 \in E_1 \setminus E_2$  and  $e_2 \in E_2 \setminus E_1$  were separated by a common event  $a \in E_1 \cap E_2$ , then the automaton  with local event sets  $E_1 = \{e_1, a\}$  and  $E_2 = \{e_2, a\}$ , was decomposable. Also if both orders of  $e_1 \in E_1 \setminus E_2$  and  $e_2 \in E_2 \setminus E_1$  were legal for occurrence of  $a$ , i.e., if  $A_S$  was

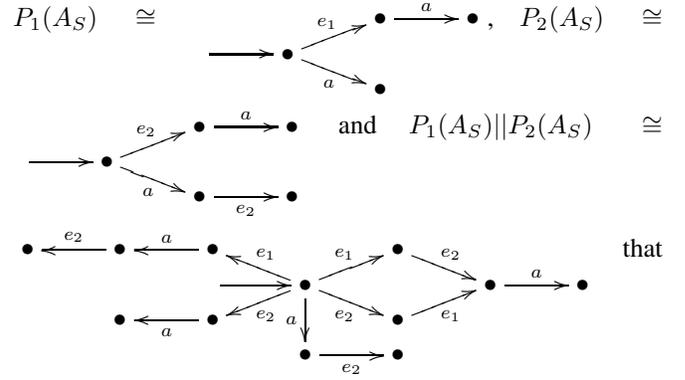


then again it was decomposable.

*Example 4:* This example illustrates an automaton that satisfies *DC1*, *DC2* and *DC4*, but it is undecomposable as it does not fulfil *DC3*, since new strings appear in  $P_1(A_S)||P_2(A_S)$  from the interleaving of two strings in  $P_1(A_S)$  and  $P_2(A_S)$ , but they are not legal in  $A_S$ . Consider the task automaton  $A_S : \rightarrow \bullet \xrightarrow{e_1} \bullet \xrightarrow{e_2} \bullet \xrightarrow{a} \bullet$  with

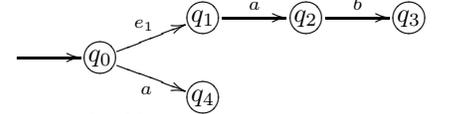


$E_1 = \{a, e_1\}$ ,  $E_2 = \{a, e_2\}$ , leading to

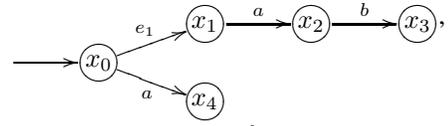


is not bisimilar to  $A_S$  since its two left branches are newly generated, while they do not appear in  $A_S$ , although both  $P_1(A_S)$  and  $P_2(A_S)$  are deterministic.

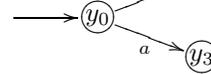
*Example 5:* This example illustrates an automaton that satisfies *DC1* and *DC2*, and *DC3*, but is undecomposable as it does not fulfil *DC4*. Consider the task automaton  $A_S : \rightarrow (q_0) \xrightarrow{e_1} (q_1) \xrightarrow{a} (q_2) \xrightarrow{b} (q_3)$



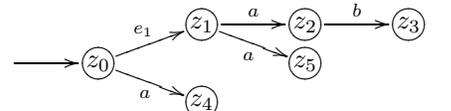
with  $E_1 = \{a, b, e_1\}$ ,  $E_2 = \{a, b\}$ , leading to  $P_1(A_S) : \rightarrow (x_0) \xrightarrow{e_1} (x_1) \xrightarrow{a} (x_2) \xrightarrow{b} (x_3)$ ,



$P_2(A_S) : \rightarrow (y_0) \xrightarrow{a} (y_1) \xrightarrow{b} (y_2)$ , and



$P_1(A_S)||P_2(A_S) : \rightarrow (z_0) \xrightarrow{e_1} (z_1) \xrightarrow{a} (z_2) \xrightarrow{b} (z_3)$



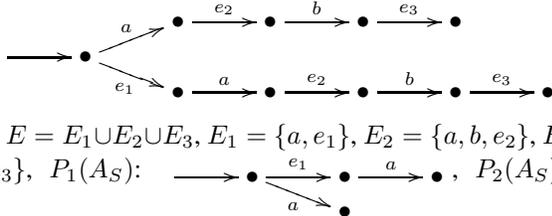
which is not bisimilar to  $A_S$ . This task automaton  $A_S$  satisfies *DC1* and *DC2* as contains no successive/adjacent transitions defined on different local event sets. It does satisfies *DC3* since any string in  $T = \{p_1(S)||p_2(S'), p_1(S')||p_2(S)\}$  ( $S$  and  $S'$  are the top and bottom strings in  $A_S$  and share the first appearing common event  $a \in E_1 \cap E_2$ ), appears in  $A_S$ . But  $A_S$  does not fulfil *DC4* since there does not exist a deterministic automaton that bisimulates  $P_2(A_S)$ . This results in the existence of a transition on string  $e_1a$  from  $z_0$  to  $z_5$  that stops in  $P_1(A_S)||P_2(A_S)$ , whereas, although  $e_1a$  transits from  $q_0$  in  $A_S$ , it does not stop afterwards. This illustrate dissymmetry in simulation relations between  $A_S$  and  $P_1(A_S)||P_2(A_S)$ . Note that  $A_S \prec P_1(A_S)||P_2(A_S)$  with the simulation relation  $R_1$  over all events in  $E$ , from all states in  $Q$  into some states in  $Z$ , as  $R_1 = \{(q_0, z_0), (q_1, z_1), (q_2, z_2), (q_3, z_3), (q_4, z_4)\}$ . Moreover,  $P_1(A_S)||P_2(A_S) \prec A_S$  with the simulation relation  $R_2$  over all events in  $E$ , from all states in  $Z$  into some states in  $Q$ , as  $R_2 = \{(z_0, q_0), (z_1, q_1), (z_2, q_2), (z_3, q_3), (z_4, q_4), (z_5, q_2)\}$ . Therefore, although  $A_S \prec P_1(A_S)||P_2(A_S)$  and  $P_1(A_S)||P_2(A_S) \prec A_S$ ,  $P_1(A_S)||P_2(A_S) \not\prec A_S$ , since  $\exists (z_5, q_2) \in R_2$ , whereas  $(q_2, z_5) \notin R_1$ . If similar to  $P_1(A_S)$ ,

$P_2(A_S)$  also had a deterministic bisimilar automaton, then for stopping of string  $e_1a$  in  $P_1(A_S)||P_2(A_S)$ , there was a state in  $Q$  reachable from  $q_0$  by  $e_1a$  and stopping there, then we had  $\forall q \in Q, z \in Z : (q, z) \in R_1 \Leftrightarrow (z, q) \in R_2$  and  $P_1(A_S)||P_2(A_S) \cong A_S$ .

### B. Decomposability Condition for $n$ agents

In practice, multi-agent systems are typically comprised of many individual agents that work as a team. The proposed procedure of decomposition can be generalized for more than two agents, considering any  $n$ -tuple of successive/adjacent private events from different private event sets. However, this approach becomes rapidly complex as the number of agents increases, as it should check  $n!$  possibilities for the order of these events. To tackle this problem one way is to use a hierarchical decomposition method to have only two individual event sets at a time for partitioning: a local event set; and the set of the rest of local event sets. In each step, the algorithm seeks these two sets such that they satisfy the decomposability conditions. This algorithm, however, depends strongly on the order of the event sets that we choose for decomposition, as it is elaborated in the following example.

*Example 6:* The automata  $A_S$ :



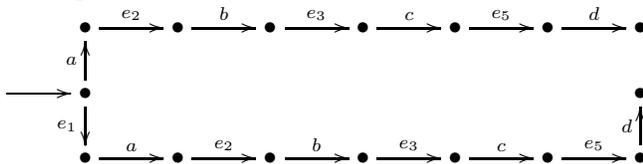
with  $E = E_1 \cup E_2 \cup E_3$ ,  $E_1 = \{a, e_1\}$ ,  $E_2 = \{a, b, e_2\}$ ,  $E_3 = \{b, e_3\}$ ,  $P_1(A_S) \cong$

$\rightarrow \bullet \xrightarrow{a} \bullet \xrightarrow{e_2} \bullet \xrightarrow{b} \bullet \xrightarrow{e_3} \bullet$ , and  $P_3(A_S) \cong \rightarrow \bullet \xrightarrow{b} \bullet \xrightarrow{e_3} \bullet$ , is decomposable as  $A_S \cong P_1(A_S)||P_2(A_S)||P_3(A_S) \cong P_1(A_S)||((P_2(A_S)||P_3(A_S))) \cong P_3(A_S)||((P_1(A_S)||P_2(A_S))) \cong P_2(A_S)||((P_1(A_S)||P_3(A_S)))$ .

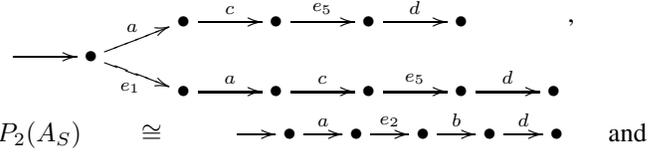
In this example,  $P_{E_2 \cup E_3}(A_S) \cong P_2(A_S)||P_3(A_S)$  and  $P_{E_1 \cup E_2}(A_S) \cong P_1(A_S)||P_2(A_S)$ , but  $P_{E_1 \cup E_3}(A_S) \not\cong P_1(A_S)||P_3(A_S)$ . This means that while choosing  $P_1(A_S)$  or  $P_3(A_S)$  as the first set allows the hierarchical algorithm to proceed up to  $A_S \cong P_1(A_S)||P_2(A_S)||P_3(A_S)$ , choosing  $P_2(A_S)$  will stuck the algorithm in the second step as  $A_S \cong P_2(A_S)||P_{E_1 \cup E_3}(A_S)$ , but  $P_{E_1 \cup E_3}(A_S) \not\cong P_1(A_S)||P_3(A_S)$ . Therefore, the algorithm depends on the order of selections.

Next, as illustrated in the following example, the above hierarchical decomposition method is only sufficient.

*Example 7:* The automata  $A_S$ :



with  $E = E_1 \cup E_2 \cup E_3$ ,  $E_1 = \{a, c, d, e_1, e_5\}$ ,  $E_2 = \{a, b, d, e_2\}$ ,  $E_3 = \{b, c, e_3\}$ ,  $P_1(A_S) \cong$



$P_2(A_S) \cong \rightarrow \bullet \xrightarrow{a} \bullet \xrightarrow{e_2} \bullet \xrightarrow{b} \bullet \xrightarrow{d} \bullet$  and  $P_3(A_S) \cong \rightarrow \bullet \xrightarrow{b} \bullet \xrightarrow{e_3} \bullet \xrightarrow{c} \bullet$ , is decomposable as  $A_S \cong P_1(A_S)||P_2(A_S)||P_3(A_S)$ . However,  $P_{E_2 \cup E_3}(A_S) \not\cong P_2(A_S)||P_3(A_S)$ ,  $P_{E_1 \cup E_2}(A_S) \not\cong P_1(A_S)||P_2(A_S)$  and  $P_{E_1 \cup E_3}(A_S) \not\cong P_1(A_S)||P_3(A_S)$ . This means that although  $A_S$  is decomposable with respect to  $P_1(A_S)$ ,  $P_2(A_S)$  and  $P_3(A_S)$ , choosing any of local event sets  $E_1$ ,  $E_2$  and  $E_3$  passes the first stage of hierarchical decomposition, but the algorithm will stuck at the second step.

Therefore, it would be very advantageous if we can find a necessary and sufficient condition for decomposability of a deterministic automaton with respect to an arbitrary finite number of local event sets. The method would be independent of the order of the local event sets and should be able to check the decomposability condition by a direct investigation. In the following, as the main result, we propose such a necessary and sufficient condition for task automaton decomposition for an arbitrary finite number of agents.

*Theorem 1:* A deterministic automaton  $A_S = (Q, q_0, E = \bigcup_{i=1}^n E_i, \delta)$  is decomposable with respect to parallel composition and natural projections  $P_i : A_S \rightarrow P_i(A_S)$ ,  $i = 1, \dots, n$  such that  $A_S \cong \prod_{i=1}^n P_i(A_S)$  if and only if  $A_S$  satisfies the following decomposability conditions (DC):  $\forall e_1, e_2 \in E, q \in Q, S \in E^*, \forall E_i \in \{E_1, \dots, E_n\}, \{e_1, e_2\} \not\subseteq E_i$ :

- DC1:  $[\delta(q, e_1)! \wedge \delta(q, e_2)!] \Rightarrow [\delta(q, e_1e_2)! \wedge \delta(q, e_2e_1)!]$ ;
- DC2:  $\delta(q, e_1e_2S)! \Leftrightarrow \delta(q, e_2e_1S)!$ ;
- DC3:  $\delta(q_0, \prod_{i=1}^n p_i(S_i))!$  for  $S_i \in \tilde{L}(A_S)$ , where,  $\tilde{L}(A_S) \subseteq L(A_S)$  is the largest subset of  $L(A_S)$  such that  $\forall S \in \tilde{L}(A_S) \exists S' \in \tilde{L}(A_S), \exists E_i, E_j \in \{E_1, \dots, E_n\}, i \neq j, p_{E_i \cap E_j}(S)$  and  $p_{E_i \cap E_j}(S')$  start with the same event, and
- DC4:  $\forall i \in \{1, 2\}, x, x_1, x_2 \in Q_i, x_1 \neq x_2, e \in E_i, t \in E_i^*, \delta_i(x, e) = x_1, \delta_i(x, e) = x_2: \delta_i(x_1, t)! \Leftrightarrow \delta_i(x_2, t)!$ .

*Proof:* In order for  $A_S \cong \prod_{i=1}^n P_i(A_S)$ , from the definition of bisimulation, it is required to have  $A_S \prec \prod_{i=1}^n P_i(A_S)$ ;  $\prod_{i=1}^n P_i(A_S) \prec A_S$ , and the simulation relations are symmetric. These requirements are provided by the following three lemmas. Due to limitation in space, the proofs for these lemmas are omitted from here, and will be provided in the extended version of this result.

Firstly, in general,  $\prod_{i=1}^n P_i(A_S)$  always simulates  $A_S$ .

*Lemma 2:* For a deterministic automaton  $A_S = (Q, q_0, E = \bigcup_{i=1}^n E_i, \delta)$  and natural projections  $P_i : A_S \rightarrow P_i(A_S)$ ,  $i = 1, \dots, n$ , it always holds that  $A_S \prec \prod_{i=1}^n P_i(A_S)$ .

This lemma shows that, in general,  $\prod_{i=1}^n P_i(A_S)$  simulates  $A_S$ . The similarity of  $\prod_{i=1}^n P_i(A_S)$  to  $A_S$ , however, is not always true (see Examples 2 and 3), and needs some conditions as stated in the following lemma.

*Lemma 3:* Consider a deterministic automaton  $A_S = (Q, q_0, E = \bigcup_{i=1}^n E_i, \delta)$  and natural projections  $P_i : A_S \rightarrow P_i(A_S)$ ,  $i = 1, \dots, n$ . Then,  $\prod_{i=1}^n P_i(A_S) \prec A_S$  if and only if *DC1*, *DC2* and *DC3* hold true for  $A_S$ .

Next, we need to show that the two simulation relations  $R_1$  (for  $A_S \prec \prod_{i=1}^n P_i(A_S)$ ) and  $R_2$  (for  $\prod_{i=1}^n P_i(A_S) \prec A_S$ ) defined by the above two lemmas are symmetric.

*Lemma 4:* Consider an automaton  $A_S = (Q, q_0, E = \bigcup_{i=1}^n E_i, \delta)$  with natural projections  $P_i : A_S \rightarrow P_i(A_S)$ ,  $i = 1, \dots, n$ . If  $A_S$  is deterministic,  $A_S \prec \prod_{i=1}^n P_i(A_S)$  with the simulation relation  $R_1$  and  $\prod_{i=1}^n P_i(A_S) \prec A_S$  with the simulation relation  $R_2$ , then  $R_1^{-1} = R_2$  (i.e.,  $\forall q \in Q, z \in Z: (z, q) \in R_2 \Leftrightarrow (q, z) \in R_1$ ) if and only if *DC4*:  $\forall i \in \{1, \dots, n\}, x, x_1, x_2 \in Q_i, x_1 \neq x_2, e \in E_i, t \in E_i^*, \delta_i(x, e) = x_1, \delta_i(x, e) = x_2: \delta_i(x_1, t) \Leftrightarrow \delta_i(x_2, t)!$ .

Now, according to Definition 4,  $A_S \cong \prod_{i=1}^n P_i(A_S)$  if and only if  $A_S \prec \prod_{i=1}^n P_i(A_S)$  (that is always true due to Lemma 2),  $\prod_{i=1}^n P_i(A_S) \prec A_S$  (that it is true if and only if *DC1*, *DC2* and *DC3* are true, according to Lemma 3) and the simulation relations are symmetric, i.e.,  $R_1^{-1} = R_2$  (that for a deterministic automaton  $A_S$ , when  $A_S \prec \prod_{i=1}^n P_i(A_S)$  with simulation relation  $R_1$  and  $\prod_{i=1}^n P_i(A_S) \prec A_S$  with simulation relation  $R_2$ , due to Lemma 4,  $R_1^{-1} = R_2$  holds true if and only if *DC4* is satisfied). Therefore,  $A_S \cong \prod_{i=1}^n P_i(A_S)$  if and only if *DC1*, *DC2*, *DC3* and *DC4* are satisfied. ■

The decomposability conditions in Theorem 1 are illustrated in the following example.

*Example 8:* Consider the automaton in Example 6. We denote the strings on the bottom and top branches of  $A_S$  as  $S_1$  and  $S_2$ , respectively. Since  $\{e_1, a\} \subseteq E_1$ ,  $\{a, e_2\} \subseteq E_2$ ,  $\{e_2, b\} \subseteq E_2$ ,  $\{b, e_3\} \subseteq E_3$ , then *DC1* and *DC2* are satisfied. Moreover,  $S_1, S_2$  contain  $a \in E_1 \cap E_2$ ,  $b \in E_2 \cap E_3$ , where  $a$  appears as the first event in both  $p_{E_1 \cap E_2}(S_1)$  and  $p_{E_1 \cap E_2}(S_2)$ . Then, according to *DC3*, following eight interleaving transitions are checked:  $\delta(q_0, p_1(S_i) | p_2(S_j) | p_3(S_k))!$ ,  $i \in \{1, 2\}, j \in \{1, 2\}, k \in \{1, 2\}$ , i.e., *DC3* is satisfied. Moreover, *DC4* is also satisfied as  $P_1(A_S), P_2(A_S)$  and  $P_3(A_S)$  are deterministic, and hence,  $A_S$  is decomposable.

## IV. CONCLUSION

The paper proposed a formal method for automaton decomposition, applicable in a top-down cooperative control design for multi-agent systems. For the class of parallel distributed systems whose global specification is represented as a deterministic automaton with given event distribution, we provide a necessary and sufficient condition for decomposability of an automaton with respect to parallel composition and natural projections into an arbitrary finite number of local event sets. Another research question is that when a task automaton is not decomposable, then how one can modify the event distribution to make the global task automaton decomposable. Another future work can be investigation of decomposability under event failure, namely, under what conditions a decomposable task automaton preserves its decomposability in spite of failure of some events.

## REFERENCES

- [1] M. Karimadini and H. Lin, "Guaranteed Global Performance Through Local Coordinations", Submitted to *Automatica*, 2010.
- [2] P. U. Lima and L. M. Custodio, *Multi-Robot Systems, Book Series Studies in Computational Intelligence*, Book Innovations in Robot, Mobility and Control, Chapter 1, vol. 8, Springer Berlin / Heidelberg, 2005.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] A. Jadbabaie, J. Lin and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transaction on Automatic Control*, vol. 48, no. 6, pp. 988C-1001, 2003.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Springer, U.S.A, 2008.
- [13] 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.
- [14] R. Morin, "Decompositions of asynchronous systems," *In CONCUR 98*, LNCS 1466, pp. 549-564, Springer, 1998.
- [15] 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.
- [16] R. Kumar and V. K. Garg, *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, Boston, 1995.