2010 8th IEEE International Conference on
Control and Automation
Xiamen, China, June 9-11, 2010

Optimal Task Automaton Decomposabilization for a Class of Global Specifications

Mohammad Karimadini and Hai Lin

Abstract— **This paper addresses the optimal task automaton decomposabilization applicable in top-down cooperative control of multi-agent systems. In [1], we proposed a divide-and-conquer approach for task automaton decomposition. In that result, given the global specification, represented as an automaton, and the logical behavior of the multi-agent system, modeled as a parallel distributed system, we proposed a necessary and sufficient condition for task automaton decomposition for two agents, such that the parallel composition of subtask automata is bisimilar to the original task automaton. This work completes the decomposition result of [1], by proposing a decomposabilization result. In this work, as the main contributions, firstly, we introduce a guideline for the basic design of private and common events, based on the specification. Secondly, after the basic event pattern attribution, the decomposability condition is checked, and if the task automaton is not decomposable, a sufficient condition is given to make it decomposable by assigning some of the private events to be common. The method is then proven to be optimal in the sense of minimum number of event conversions leading to minimum increment in the required communication capacity. An example is given to illustrate the concept and significance of optimal task decomposabilization.**

I. INTRODUCTION

Multi-agent system is a broad rapidly developing area with increasing interest of researches in many fields such as distributed surveillance, target following, distributed defence systems, underwater or space exploration, assembling and transportation, large scale manufacturing systems, and rapid emergency response [2]. The cooperative control of multi-agent systems, that are typically distributed in nature, is still in its infancy and possess significant technical and theoretical challenges that fall beyond the conventional control methods [3], [4].

The experimental results on multi-agent systems have shown that sophisticated collective capabilities can be achieved through the collaboration of simple agents with simple local interaction rules. Namely, a team of cooperative agents shows more functionality and reliability than a single or even a collection of multi-skilled agents that have no cooperations [5]-[8]. These studies induce increasing motivations towards more research efforts in this area. Most of these research activities in multi-agent systems, so far, have been devoted to bottom-up methods to know how and what kind of team behavior can be generated by a given set of local control rules [9], [10]. A more important problem, however, is the design of these local control and interaction rules such that given desired global behaviors can be achieved, cooperatively, by design [11]. The desired global specification could be very complicated, and the objectives and design may fall beyond the traditional output regulation or path planning [3], [4], [12], [13]. Moreover, the bottom-up scenario in swarming robotics may fail to guarantee the correctness by design, due to lack of understanding on how to manipulate the local rules to achieve the global behavior. To avoid the iterative trial and error design procedure, a new efficient and formal method is required to design the local control laws and interaction rules in order to achieve the global specification in a correct-by-design manner.

To achieve such an ambitious research goal, in [1] we proposed a "divide-and-conquer" approach by decomposing a global specification into sub-specifications for individual agents. The decomposition was developed in such a way that satisfaction of these sub-specifications lead to global desired behavior, by design. In this work, we assume that the global specification is given as an automaton, defined over the union of all local events. Accordingly, the logical behavior [14] of a multi-agent system can be modeled as a parallel distributed systems, the parallel composition of local plant automata [15].

In [1], the decomposition approach was developed based on a so-called natural projection scheme that obtains each local task automaton by ignoring the transitions that are not defied on its corresponding local event set. The cooperative collection (defined by parallel composition) of these sub-task automata would be equivalent (in the sense of bisimilarity) to the original task automaton. Given a task automaton and distribution of events to local event sets, it is always possible to do such kind of projections, but we have shown [1] that it is not always possible to decompose an automaton into sub-automata by natural projection, such that the parallel composition of these sub-automata is bisimilar to the original automaton. Furthermore, we have found [1] that an automaton is decomposable with respect to two local event sets if and only if for any two successive or adjacent private events from different private event sets, both orders of these events should be legal, and furthermore, the global automaton is required to contain the interleaving of local strings that share the first common event.

The results in [1], however, rely on the predefined sets of private and common events for each agent. This is a strong assumption and implies that the communication pattern between the agents is known. Usually, one needs to design the communication pattern as well in multi-agent systems. Furthermore, if a global task automaton is not decomposable, a natural follow-up question is how to modify

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

the event pattern for each agent to make the global task automaton decomposable. To address these two problems, this paper has two main contributions: Firstly, based on the global specification, we introduce a method to define the private and common events among the local event sets. In this framework, an event can be either an "observation"(sensor reading) or an "action" (the pair of actuator's command and feedback). The method starts with a preliminary attribution of the events, based on the basic requirements on coordination on some observations or actions. Secondly, after this setting of the functionality, the decomposability condition mentioned in [1] is checked, and if the task automaton is not decomposable, then, for a class of automata, the paper proposes a sufficient condition to make it decomposable by transforming some of the private events to the common ones. Furthermore, this decomposabilization is guaranteed to be optimal in the sense of minimum number of common events leading to minimum increasing on communication links between the agents.

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 concept of "common observation" as well as "common action" between two agents and shows that how one can transform a private event into a common event, in different situations. This section then proposes a sufficient condition for optimal decomposabilization of a class of undecomposable global task automata. An example is then given in Section IV to illustrate the concept of optimal task automaton decomposabilization. Finally, the paper concludes with remarks and discussions in Section V.

II. PROBLEM FORMULATION

We first recall the definition of an automaton [17].

Definition 1: (Automaton) An automaton is a tuple $A = (Q, Q_0, E, \delta)$ consisting of

- a set of states Q;
- a set of initial states $Q_0 \subseteq Q$;
- 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 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 behavior of the task automaton and the collective behavior of its decomposed automata, we use the simulation relation [16], defined as follows.

Definition 4: (Simulation) Let two automata $A_i = \left(Q_i, Q_i^0, E, \delta_i\right)$, $i = 1, 2$. 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) $\forall q_1^0 \in Q_1^0, \exists q_2^0 \in Q_2^0 : \left(q_1^0, q_2^0\right) \in R$
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) [17] 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

- $Q = Q_1 \times Q_2$;
- $Q_0 = Q_1^0 \times Q_2^0$;
- $E = E_1 \cup E_2$;
- $\forall (q_1, q_2) \in Q, e \in E : \delta((q_1, q_2), e) =$
$$\begin{cases} \left(\delta_1(q_1, e), \delta_2(q_2, e)\right), & \text{if } \begin{cases} \delta_1(q_1, e)!, \delta_2(q_2, e)! \\ e \in E_1 \cap E_2 \end{cases}; \\ \left(\delta_1(q_1, e), q_2\right), & \text{if } \delta_1(q_1, e)!, e \in E_1 \backslash E_2; \\ \left(q_1, \delta_2(q_2, e)\right), & \text{if } \delta_2(q_2, e)!, e \in E_2 \backslash E_1; \\ undefined, & \text{otherwise} \end{cases}$$

In [1], we used natural projection to obtain the local task automata from global task automaton. Natural projection can be defined over strings as $p_{E_i} = p_i : E^* \rightarrow E_i^*$, or over automata by $P_i(A_S) : A_S \rightarrow A_{S_i}$, 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 as follows.

Definition 6: (Natural Projection) 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 on strings as $p_i(\varepsilon) = \varepsilon$, and $\forall S \in E^*, e \in E$: $p_i(Se) = \begin{cases} p_i(S)e & \text{if } e \in E_i; \\ p_i(S) & \text{otherwise.} \end{cases}$

Now, 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, the natural projection of A_S with respect to E_i is defined as $P_i(A_S) = (Q_i = Q_{/\sim E_i}, Q_i^0 = Q_{/\sim E_i}^0, E_i, \delta_i)$, 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, $[q]$ 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 $\delta(q, e) = q' \wedge e \notin E_i \Rightarrow q \sim_{E_i} q'$.

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 7: Consider two sequences $q_1 \stackrel{e_1}{\rightarrow} q_2 \stackrel{e_2}{\rightarrow} ... \stackrel{e_n}{\rightarrow} q_n$ and $q_1' \stackrel{e_1'}{\rightarrow} q_2' \stackrel{e_2'}{\rightarrow} ... \stackrel{e_m'}{\rightarrow} q_m'$, 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 8: (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 $P_1(A_S)||P_2(A_S) \cong A_S$.

In [1], we have shown that not all automata are decomposable with respect to parallel composition and natural projections. Consequently, a necessary and sufficient condition was proposed for task automaton decomposition with respect to two local event sets as

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 $A_S \cong P_1(A_S)||P_2(A_S)$ if and only if it satisfies the following decomposability conditions (DC): $\forall e_1 \in E_1 \backslash E_2, e_2 \in E_2 \backslash E_1, q \in Q$, $S \in E^*$,

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

The next follow-up question is that if an automaton is not decomposable then how we can design the event pattern such that it becomes decomposable. Moreover, if it is possible to do so, how we can do it in an optimal way such that it needs minimum number of common events. This problem is addressed in the next section and it is formally stated as

Problem 1: Consider a deterministic task automaton A_S with event set E and 2 agents. If A_S is not decomposable, how can we optimally design the private and common events of local event sets such that A_S becomes decomposable with respect to parallel composition and natural projections $P_i : A_S \rightarrow P_i(A_S)$, $i = 1, 2$ with the minimum number of common events?

III. TASK AUTOMATON DECOMPOSABILIZATION

This section is devoted to Problem 1. This problem can be arisen in either the first stage of the design, that the sensors and actuators should be designed for each agent; or when a global task automaton is given with a fixed event pattern and its decomposition fails due to lack of communication on some of the private events. In this case, the private and common event sets for some agents should be modified to make the global task decomposable. To formulate this problem, we synthetically distinguish the events by two types:

- observation: which is a sensor reading from an external measurement;
- action: which is a pair of a command to, and a feedback from an actuator.

An observation is a private event for an agent in which possesses the corresponding sensor. This observation can be a common event with the other agent if

- it is shared through communication;
- the second agent also is equipped with an identical sensor to read the same measurement, and both agents communicate on this event and interpret it as the same label, or
- two sensors in two agents read two different measurements, but inform each other about these readings and interpret them with same label.

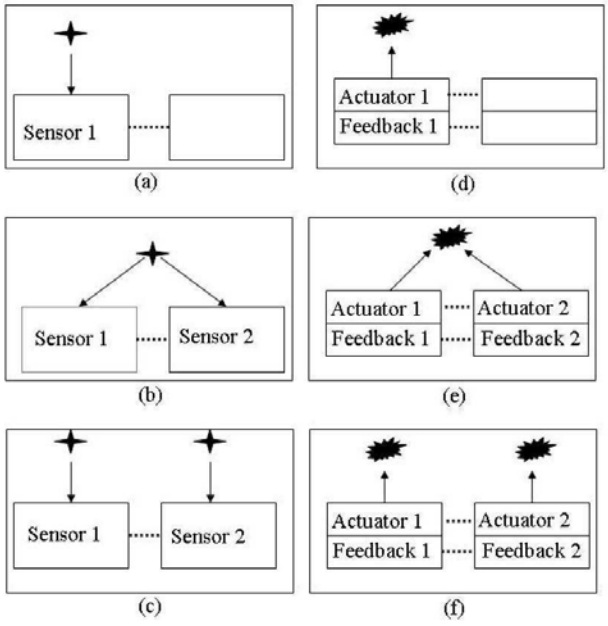


Fig. 1. Illustration of observation and action common events.

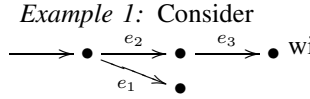
The example of the first case is when one robot observes a target and alerts the other robot (Figure 1 (a)). This event is used in both robots as "target detected". It can be also the case that both robots are equipped with sensors and observe the same target. This "target detected" event also can be considered as a common event (Figure 1 (b)). The third case for instance refers to the case that two robots with two cameras, observe two different targets. But, if only the existence of a target is important rather than a specific target, again this event can be shared through the communication to be considered as a common event (Figure 1 (c)). By "label" we mean the symbol of the event that is used as an input in each local event set. For example, target recognition by two sensors in two robots can be either adopted as two different labels (private events) "Robot 1 recognized the target" and "Robot 2 recognized the target", or, by the same label as "target recognized".

Another type of event could be action that consists of a pair of command and feedback; a command is sent to an actuator, and its activation is sensed and feedbacked to the agent. Similar to the "observation", an "action" is initially private event and it could be a common event with the other agent, if

- its command and feedback are shared through communication;
- the second agent is also equipped with an identical actuator to activate and feedback the activation of the same operation, and both agents communicate on this event and interpret it as the same label, or
- two actuators in two agents are synchronized to accomplish two different operations, but inform each other about these commands and feedbacks, and interpret them with same label.

The example of the first case is when one robot opens a screw and the other robot is informed about the command for this screwing and about the accomplishment of this operation (Figure 1 (d)). The example for the second case is when two robots synchronously push a door (Figure 1 (e)). In this case, each robot needs an actuator (motor driver here), and should synchronize on the pushing command as well as on the feedback from the accomplishment of this task. Both robots interpret this event as the same label of "pushing the door". The third case refers, for example, to the case that robots push two different objects. Here, both robots are equipped with actuators, synchronize on their commands and feedbacks and interpret both actions with the same label of "pushing" (Figure 1 (f)).

After setting this basic required functionality of the agents and their capabilities (the set of sensors and actuators for each agent), the number of sensors and actuators are fixed, and still, some of the private events may need to become common via the communication links, to make A_S decomposable. The core idea is to find the set of private events between two local event sets, such that they violate the decomposability conditions $DC1$ and $DC2$, stated in Lemma 1, and let one event among each such event pair to become a common event. This method, however, is applicable to the class of automata that satisfy $DC3$. Moreover, this approach may offer different sets of private events to become common events. Since conversion of each private event to a common event demands an extra communication link capacity, the optimal decomposition is then introduced based on minimum number of such conversions. Assuming the satisfaction of $DC3$, for all sets of private events that may convert to common events to remove the violation of DC, the optimal decomposition algorithm will therefore seek the set with the minimum cardinality. This concept is illustrated in the following simple example and serves as the essential idea in the main result.

Example 1: Consider the automaton 

with local event sets $E_1 = \{e_1, e_3\}$ and $E_2 = \{e_2\}$. This automaton is undecomposable due to violation of DC by $e_2 \in E_2 \backslash E_1$ and $\{e_1, e_3\} \in E_1 \backslash E_2$. To make it decomposable, one event among the set $\{e_1, e_2\}$ and another event among the set $\{e_2, e_3\}$ should become common. This implies that either $\{e_2\}$ or $\{e_1, e_3\}$ should become common. Therefore, in order for optimal decomposition, $\{e_2\}$ is chosen to become common due to its minimum cardinality. It is obvious that in this case only one event should become common while if $\{e_1, e_3\}$ was chosen, then two events were required to become common.

In order to automate this procedure, following operators are introduced to capture the sets of private events that cause the violation of DC. Consider the global task automaton A_S with local event sets E_i for $n = 2$ agents such that $E = E_1 \cup E_2$, and suppose that A_S satisfies $DC3$. In this case, the satisfaction of DC is reduced to $DC1$ and $DC2$. Then, the violating set operator $V_{1,2} : A_S \rightarrow (E_1 \backslash E_2 \times E_2 \backslash E_1) \cup (E_2 \backslash E_1 \times E_1 \backslash E_2)$, indicates the set of private event pairs that violate DC (violating pairs), and is defined as $V_{1,2}(A_S) := \{(e_1, e_2) \in \{(E_1 \backslash E_2, E_2 \backslash E_1), (E_2 \backslash E_1, E_1 \backslash E_2)\} | \exists q \in Q$ s.t. $[[\delta(q, e_1)! \wedge \delta(q, e_2)!] \vee \delta(q, e_1 e_2)!] \wedge \neg[\delta(q, e_1 e_2)! \wedge \delta(q, e_2 e_1)!]$ or $\neg[\delta(q, e_1 e_2 S)! \Leftrightarrow \delta(q, e_2 e_1 S)!]\}$, for some $S \in E^*$. Moreover, $W_{1,2} : A_S \rightarrow (E_1 \backslash E_2) \cup (E_2 \backslash E_1)$ is defined as $W_{1,2}(A_S) := \{e | \exists e' s.t. (e, e') \in V_{1,2}(A_S) \vee (e', e) \in V_{1,2}(A_S)\}$, and shows the set of private events that contribute in $V_{1,2}(A_S)$ (violating events). Furthermore, $V^e_{1,2} : A_S \rightarrow (E_1 \backslash E_2 \times E_2 \backslash E_1) \cup (E_2 \backslash E_1 \times E_1 \backslash E_2)$ is defined as $V^e_{1,2}(A_S) := \{(e_1, e_2) \in V_{1,2}(A_S) | e \in \{e_1, e_2\}\}$, and finally, we define $W^e_{1,2} : A_S \rightarrow (E_1 \backslash E_2) \cup (E_2 \backslash E_1)$ as $W^e_{1,2}(A_S) := \{e' \in (E_1 \backslash E_2) \cup (E_2 \backslash E_1) | (e, e') \in V^e_{1,2}(A_S) \vee (e', e) \in V^e_{1,2}(A_S)\}$. $W^e_{1,2}(A_S)$ is the set of events that have been paired up with e in A_S, violating DC.

To remove the violation of DC, it suffices that for every violating pair, at least one of its events become common between E_1 and E_2. Since some of the pairs share events, the minimum number of event conversion would be obtained by forming a set of events that are most frequently paired

to form the violating pairs. Such choice of events offers a set of private events that span all violating pairs. Therefore, the maximum appearance of an event in violation of DC can be captured by comparing $|W^e_{1,2}(A_S)|$ for all events in $W_{1,2}(A_S)$. Here, $|.|$ denotes the set's cardinality.

This procedure is stated formally in the following Algorithm to be used in the main result for optimal automaton decomposition.

Algorithm 1:

1) Set ${}^0V_{1,2}(A_S) = V_{1,2}(A_S)$, ${}^0W_{1,2}(A_S) = W_{1,2}(A_S)$, ${}^0D_{1,2}(A_S) = \emptyset$, $k = 1$;
2) color the events in ${}^0W_{1,2}(A_S)$ whose conversion to common events violates $DC3$;
3) mark an uncolored unmarked event e_k from the set ${}^{k-1}W_{1,2}(A_S)$ with the maximum $|{}^{k-1}W^e_{1,2}(A_S)|$;
4) form ${}^kV_{1,2}(A_S)$ and ${}^kW_{1,2}(A_S)$ by removing all elements in ${}^{k-1}V_{1,2}(A_S)$ that correspond to e_k;
5) update ${}^kW^e_{1,2}(A_S)$ for all events in ${}^kW_{1,2}(A_S)$, update ${}^kD_{1,2}(A_S) = {}^{k-1}D_{1,2}(A_S) \cup \{e_k\}$, set $k = k+1$ and got to set 2;
6) continue, until there exists no uncolored unmarked event, at $k = K$;
7) $D_{1,2}(A_S)$ is the set of all the marked events, i.e., $D_{1,2}(A_S) = {}^KD_{1,2}(A_S)$.

Where, ${}^kV_{1,2}(A_S)$ and ${}^kW_{1,2}(A_S)$ are $V_{1,2}(A_S)$ and $W_{1,2}(A_S)$ in the $k-th$ iteration.

Based on this formulation, an undecomposable task automaton can become decomposable with minimum number of conversions of private events to common events, as the following theorem.

Theorem 1: Consider a deterministic task automaton A_S, satisfying $DC3$, with local event plants A_{P_i}, and local event sets E_i, $i = 1, 2$, such that $E = E_1 \cup E_2$. If A_S is not decomposable with respect to parallel composition and natural projections $P_i : A_S \rightarrow P_i(A_S)$, $i = 1, 2$, and ${}^0V_{1,2}(A_S)$ contains no event pair with both colored, it becomes optimally decomposable if all events in $D_{1,2}(A_S)$, derived from Algorithm 1, become common events between E_1 and E_2.

Proof: We first define sets ${}^kF_{1,2}(A_S)$ and ${}^kF^e_{1,2}(A_S)$ to be the sets of unordered pairs from ${}^kV_{1,2}(A_S)$ and ${}^kV^e_{1,2}(A_S)$, respectively, by removing the repetitive pairs, indicating the links between the violating events (violating links) contributing in ${}^kV_{1,2}(A_S)$ and ${}^kV^e_{1,2}(A_S)$, respectively. In each iteration k, the event e_k with the maximum $|{}^{k-1}W^e_{1,2}(A_S)|$ is marked, and its corresponding violating pairs are removed until in the $K-th$ step all violating pairs of $V_{1,2}$ are removed. Therefore, ${}^kD_{1,2}(A_S) - {}^{k-1}D_{1,2}(A_S) = \{e_k\}$, $|{}^kD_{1,2}(A_S)| = |{}^{k-1}D_{1,2}(A_S)| + 1$, $|D_{1,2}(A_S)| = K$, ${}^{k-1}F_{1,2}(A_S) - {}^kF_{1,2}(A_S) = {}^k\Delta F_{1,2}(A_S)$ and $|{}^kF_{1,2}(A_S)| - |{}^{k-1}F_{1,2}(A_S)| = -|{}^k\Delta F_{1,2}(A_S)| = -|W^{e_k}_{1,2}(A_S)|$. Here, ${}^kD_{1,2}(A_S)$ and ${}^k\Delta F_{1,2}(A_S)$ are the set of marked events and removed violating links in the $k-th$ iteration, respectively.

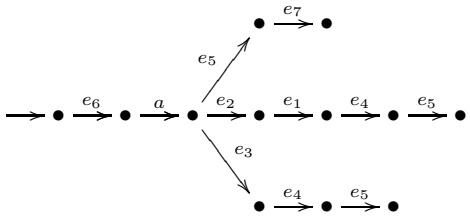
Furthermore, since the removing violating links in each iteration will not appear in the next iterations then ${}^k\Delta F_{1,2}(A_S) \cap {}^l\Delta F_{1,2}(A_S) = \emptyset, \forall k, l \in \{1, ..., K\}, k \neq l$, and since the algorithm continues until all violating pairs are removed, then, $\bigcup_{k=1}^{K} {}^k\Delta F_{1,2}(A_S) = F_{1,2}(A_S)$, and hence, $\{{}^k\Delta F_{1,2}(A_S)\}, k \in \{1, ..., K\}$ partitions $F_{1,2}(A_S)$.

Since in each step k, one event e_k is added to ${}^kD_{1,2}$ and $|W^{e_k}_{1,2}|$ violating links are removed from the set ${}^{k-1}F_{1,2}(A_S)$, then the number of violating links is reduced with the minimum gradient decent $-\max_{e_k \in {}^{k-1}W_{1,2}} |W^{e_k}_{1,2}|$. When all violating links are removed in the $K-th$ step, then $D_{1,2}(A_S) = K$ will be minimum due to minimum gradient decent of ${}^kF_{1,2}$ with respect to iteration number k. Therefore, the number of these partitions is minimum if the size of each partition is maximum in each iteration, i.e., $|{}^k\Delta F_{1,2}(A_S)| = |{}^k\Delta F_{1,2}(A_S)|_{max} \forall k \in \{1, ..., K\} \Rightarrow K = K_{min}$, that is, $\forall k \in \{1, ..., K\} : |W^{e_k}_{1,2}| = \max_{e_k \in {}^{k-1}W_{1,2}} |W^{e_k}_{1,2}| \Rightarrow |D_{1,2}(A_S)| = |D_{1,2}(A_S)|_{min}$, and the proof is completed. ■

IV. EXAMPLE

This section examines an example to illustrate the concept of optimal task automaton decomposabilization.

Example 2: Consider two robots in a manufacturing station, that are supposed to perform operations e_1, e_2, e_3, e_4, e_5, e_6 and e_7, according to the order specified in the task automaton A_S:



with the local event sets $E_1 = \{e_1, e_3, e_5, e_7, a\}$, $E_2 = \{e_2, e_4, e_6, a\}$. The set of pairs that violate DC (the set of violating pairs) is obtained as $V_{1,2}(A_S) = \{(e_1, e_2), (e_1, e_4), (e_2, e_3), (e_2, e_5), (e_3, e_4), (e_4, e_5)\}$, with $W_{1,2}(A_S) = \{e_1, e_2, e_3, e_4, e_5\}$. It can be seen that the private events e_6 and e_7 and the common event a are not included in $W_{1,2}(A_S)$ as they have no contribution in violation of DC. Algorithm 1 is applied to find $D_{1,2}(A_S)$ as follows. During the algorithm we denote $W_{1,2}(A_S)$ and $V_{1,2}(A_S)$ with $W_{1,2}$ and $V_{1,2}$, for simplicity. The initial sets ${}^0V^e_{1,2}$ and ${}^0W^e_{1,2}$ for violating events are obtained as:

e	${}^0V^e_{1,2}$	${}^0W^e_{1,2}$
e_1	$\{(e_1, e_2), (e_1, e_4)\}$	$\{e_2, e_4\}$
e_2	$\{(e_1, e_2), (e_2, e_3), (e_2, e_5)\}$	$\{e_1, e_3, e_5\}$
e_3	$\{(e_2, e_3), (e_3, e_4)\}$	$\{e_2, e_4\}$
e_4	$\{(e_1, e_4), (e_3, e_4), (e_4, e_5)\}$	$\{e_1, e_3, e_5\}$
e_5	$\{(e_2, e_5), (e_4, e_5)\}$	$\{e_2, e_4\}$

Two events e_2 and e_4 have the same $|{}^0W^e_{1,2}|$ which equals to $\max_{e \in {}^0W_{1,2}} |{}^0W^e_{1,2}| = 3$. Therefore, we can chose either ${}^1D_{1,2}(A_S) = \{e_2\}$ or ${}^1D_{1,2}(A_S) = \{e_4\}$. Note that none of them are colored, i.e., they do not violate $DC3$. The reason is that e_2 appears not in more than one branch after a.

Moreover, the transition after e_4 in two distinct branches, is the same, and hence, there will be no violating interleaving of these two branches. If one choose ${}^1D_{1,2}(A_S) = \{e_2\}$, then ${}^1V_{1,2}(A_S) = \{(e_1, e_4), (e_3, e_4), (e_4, e_5)\}$ and ${}^1W^e_{1,2}(A_S) = \{e_1, e_3, e_4, e_5\}$, and

e	${}^1V^e_{1,2}$	${}^1W^e_{1,2}$
e_1	$\{(e_1, e_4)\}$	$\{e_4\}$
e_3	$\{(e_3, e_4)\}$	$\{e_4\}$
e_4	$\{(e_1, e_4), (e_3, e_4), (e_4, e_5)\}$	$\{e_1, e_3, e_5\}$
e_5	$\{(e_4, e_5)\}$	$\{e_4\}$

Since $\max_{e \in {}^1W_{1,2}} |{}^1W^e_{1,2}| = |{}^1W^{e_4}_{1,2}| = 3$, then ${}^2D_{1,2}(A_S) = \{e_2, e_4\}$, and consequently, ${}^2V_{1,2}(A_S) = \emptyset$, ${}^2W_{1,2}(A_S) = \emptyset$, $|{}^2W_{1,2}(A_S)| = 0$ and the algorithm ends, here. If, however, e_4 was chosen, instead of e_2, then ${}^1V_{1,2}(A_S) = \{(e_1, e_2), (e_2, e_3), (e_2, e_5)\}$ and ${}^1W_{1,2}(A_S) = \{e_1, e_2, e_3, e_5\}$, and

e	${}^1V^e_{1,2}$	${}^1W^e_{1,2}$
e_1	$\{(e_1, e_2)\}$	$\{e_2\}$
e_2	$\{(e_1, e_2), (e_2, e_3), (e_2, e_5)\}$	$\{e_1, e_3, e_5\}$
e_3	$\{(e_2, e_3)\}$	$\{e_2\}$
e_5	$\{(e_2, e_5)\}$	$\{e_2\}$

Since $\max_{e \in {}^1W_{1,2}} |{}^1W^e_{1,2}| = |{}^1W^{e_2}_{1,2}| = 3$, then ${}^2D_{1,2}(A_S) = \{e_4, e_2\}$, and consequently, ${}^2V_{1,2}(A_S) = \emptyset$, ${}^2W_{1,2}(A_S) = \emptyset$, $|{}^2W_{1,2}(A_S)| = 0$ and the algorithm ends, here, resulting the same $D_{1,2}(A_S) = \{e_4, e_2\}$ as previous choice. Different choices for ${}^kD_{1,2}(A_S)$ may result in different final sets, however, the cardinality of final $D_{1,2}(A_S)$ will be unique due to update of ${}^kV_{1,2}(A_S)$ based on maximum $|{}^kW_{1,2}(A_S)|$ in each iteration.

V. CONCLUSION

The paper proposed a formal method for optimal automaton decomposabilization, to complete the previous result [1] on task automaton decomposition, applicable in a top-down decentralized cooperative control of distributed discrete event systems. Given a set of agents whose logical behaviors are modeled in a parallel distributed system with fixed event pattern (private and common events) for each agent, and a global task automaton, [1] provided a necessary and sufficient condition for decomposability of an automaton with respect to parallel composition and natural projections into two local event sets. This work completes the results of [1], as following two contributions: Firstly, this paper gives a procedure to initially define the private and common events based on the atomic propositions of the global specification, and shows that how a private event can become a common one. Secondly, once the initial event pattern is attributed for sensors and actuators based on the required functionality, if the global task automaton is not decomposable, we have proposed a sufficient condition for optimal task automaton decomposabilization by converting some (as minimum as possible) private events to common events between two agents. This optimality would be appreciated for design as it leads to less number of communication links. This result is given for a team of two agents. Generalization of this method into an arbitrary finite number of local plants is under development. Furthermore, this method is restricted to the class of global task automata that satisfy $DC3$. Another future work could be the extension of this result into the whole class of deterministic task automata.

VI. ACKNOWLEDGMENTS

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

REFERENCES

[1] M. Karimadini and H. Lin, "Synchronized Task Decomposition for two Cooperative Agents," Accepted by *IEEE International Conference on Robotics, Automation and Mechatronics (RAM 2010)*, 2010.

[2] 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.

[3] 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.

[4] 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.

[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] 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.

[7] 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.

[8] 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. 320-330, 2007.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Springer, U.S.A, 2008.

[17] R. Kumar and V. K. Garg, *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, Boston, 1995.