Learning-based Formal Synthesis of Cooperative Multi-agent Systems with an Application to Robotic Coordination

Jin Dai, Alessandro Benini, Hai Lin, Panos J. Antsaklis, Matthew J. Rutherford and Kimon P. Valavanis

Abstract—We propose a computationally efficient framework to automatically synthesize coordination and control strategies for cooperative multi-agent systems from global tasks via “divide-and-conquer”: first the global tasks are decomposed into local ones; then local control strategies for each agent are synthesized to satisfy local tasks; finally, compositional verification technique is adopted to coordinate the collective behavior of the agents such that the fulfillment of local tasks will guarantee the satisfaction of the global ones. Modified $L^\infty$ algorithms are adapted to both the local synthesis and the compositional verification. In this paper, the co-design framework is applied to multi-robot coordination. The paper concludes with the presentation of a series of computational and software tools developed to integrate automatic supervisor synthesis and inter-robot communication so that the robots can cooperatively satisfy a request-response team performance through coordination.

I. INTRODUCTION

Cooperative multi-agent systems, which consist of a team of spatially-distributed agents collaborating through wireless communication networks to achieve desirable team tasks, represent a typical class of cyber-physical systems (CPS). Strongly motivated by its great potential in both academic and industrial applications, such as power grids, transportation networks and robotic systems, multi-agent systems has emerged rapidly as a hot research area at the intersection of control theory, communication and computer science in recent years [3], [8]–[10], [15], [16], [21].

One of the most highlighted problems in multi-agent system is how to design local control policies for each agent such that certain desirable global specifications can be accomplished through multi-agent coordination. The bottom-up design methodologies, in which sophisticated collective behaviors of the agent team emerge through predefined control and coordination policies, have been applied for various coordination purposes using different modeling formalisms, such as flocking [15], consensus [16], rendezvous [6] and synchronization [24] of multi-agent teams. Despite its popularity and simplicity, the bottom-up design methodologies lack flexibility to varying control specifications and lack guarantees for achieving high level specifications when dealing with uncertain and dynamic environment. By contrast, top-down design approaches adopt a “divide-and-conquer” scheme to accomplish more complex global specifications by a team of agents. Such methodologies have been widely adopted in the symbolic planning problems of multi-agent systems for various forms of specifications, such as linear temporal logic formulas [8] [12] and “trace-closed” regular languages [3]. Correct-by-construction design of hybrid controllers for reactive motion and mission planning problems were also addressed in [9] and [10]. [11] studied multi-agent task decomposition problems and necessary and sufficient conditions, under which the global tasks can be retrieved by the assigned local ones, were presented; however, limited results of local control design has been proposed.

In this paper, we present a study of the feasibility of applying formal design approaches to coordination of robotic teams. By modeling each robot’s behavior as a deterministic finite automaton [2] and the behavior of the robotic team as the concurrent operation among them [23], we propose a top-down framework to synthesize both local discrete-event controllers (termed as supervisors) for each agent and the coordination schemes: (i) the global specification is decomposed into local subtasks; (ii) a learning-based formal synthesis scheme is proposed such that each robot synthesizes its local supervisor to satisfy the local tasks; (iii) compositional verification using assume-guarantee paradigm [17] is deployed to check whether the collective behaviors of the local controlled robots can satisfy the global specification, while mitigating the state explosion issues by avoiding the computation of the composed system. Once the checking fails, a counterexample is generated and used to modify the task decomposition and the automatic synthesis process repeats until success.

The remainder of the paper is organized as follows. Section II introduces the coordination and control co-design problem for multi-agent systems with a motivating example in multi-robot coordination. The automatic supervisor synthesis framework is proposed in Section III. The motivating example is reviewed and experimental results are presented in Section IV. Finally, Section V concludes the paper.

II. PROBLEM FORMULATION

A. Multi-robot Coordination

Consider a cooperative multi-agent systems that consists of three robots, namely, $R_1$, $R_2$ and $R_3$, all of which have the same communication and localization capabilities. Furthermore, $R_2$ is assumed to be equipped with the rescue
and fire-fighting capabilities, while \( R_1 \) and \( R_3 \) are normal robots with the pushing capability. All doors are equipped with spring to be closed automatically, whenever there is no force to keep them open. Initially, all of the three robots are positioned in Room 1. Rooms 2 and 3 are accessible from Room 1 by one-way door \( D_2 \) and two-way doors \( D_1 \) and \( D_3 \), as shown in Figure 1.

![Fig. 1: The scenario of the multi-robot coordination example.](image)

The multi-robot coordination requires that, after a help request from Room 2 for fire extinguishing, \( R_2 \) needs to go to Room 2, urgently from the one-way door \( D_2 \) and accomplish its task there and come back immediately to Room 1 from the two-way, but heavy door \( D_1 \) that needs the cooperation of two robots \( R_1 \) and \( R_3 \) to be opened. To save time, as soon as the robots hear the help request from Room 2, \( R_2 \) heads to Room 2 through \( D_2 \) and either \( R_1 \) or \( R_3 \) goes to Room 3 from the two-way door \( D_3 \), while the rest of them stays at Room 1. After \( R_2 \) accomplishes its task, the robot in Room 3 and the one staying at Room 1 approach the door \( D_1 \), synchronously open it and wait for \( R_2 \) to return to Room 1. Afterwards, \( R_1 \) and \( R_3 \) move backwards to close \( D_1 \) and the one in Room 3 returns to Room 1 through \( D_3 \). All three robots then keep staying at Room 1 for next service request. In summary, the team specifications for the multi-robot coordination include the following two aspects.

- **Request-response** Robot \( R_2 \) should respond to the request by entering Room 2 through \( D_2 \) and returning to Room 1 through \( D_1 \) when it accomplishes its task;
- **Coordination** \( D_1 \) should be opened jointly by \( R_1 \) and \( R_3 \).

### B. Automata Models of Multi-robot Systems

We show how to use a discrete-event system formalism to characterize the behavior of the robot team. Towards this end, we model the behavior of robot \( R_i, i \in \{1, 2, 3\} \) as a deterministic finite automaton (DFA) \( G_i = (Q_i, \Sigma_i, q_0, i, \delta_i) \), where \( Q_i \) is the set of local states, \( \Sigma_i \) is the local event set, \( \delta_i : Q_i \times \Sigma_i \rightarrow Q_i \) is the (partial) transition function, and \( q_0, i \in Q_i \) is the local initial state, \( \delta_i \) can be extended to \( Q_i \times \Sigma_i^* \rightarrow Q_i \) in a natural way. The states of a robot may represent different regions of the partitioned working space (in motion planning) and/or the service status (in mission planning), while the event alphabet \( \Sigma_i \) may include sensing, motion and service capabilities of the underlying robot. The behavior generated by robot \( G_i \) is given by \( L(G_i) = \{ s \in \Sigma_i^* | \delta(q_0, i, s) \} \) is defined. The prefix closure \[2\] of a language \( K_i \subset \Sigma_i^* \) is written as \( \overline{K_i} \). \( K_i \) is called prefix-closed if \( \overline{K_i} = K_i \).

**TABLE I: Events in the Multi-robot Coordination Example.**

| Event       | Explanation                      |
|-------------|----------------------------------|
| \( h_i \)  | Robot \( R_i \) receives the service request, \( i = 1, 2, 3 \). |
| \( R_i to D_1 \) | Robot \( R_i \) approaches the door \( D_1 \), \( i = 1, 3 \). |
| \( R_i on D_1 \) | Robot \( R_i \) localizes itself at the door \( D_1 \), \( i = 1, 3 \). |
| \( R_i tok \) | Robot \( R_i \) heads for Room \( k \), \( i, k = 1, 2, 3 \). |
| \( R_iink \) | Robot \( R_i \) stays at Room \( k \), \( i, k = 1, 2, 3 \). |
| Open        | command for moving forward to open \( D_1 \). |
| Close       | command for moving backward to close \( D_1 \). |
| \( D_1 open \) | The status that \( D_1 \) is opened by \( R_1 \) and \( R_3 \). |
| \( D_1 closed \) | The status that \( D_1 \) is closed by \( R_1 \) and \( R_3 \). |
| r           | All the robots return to and stay at Room 1 for next request. |

The global event set is given by \( \Sigma = \bigcup_{i=1}^{3} \Sigma_i \), and is listed in Table 1. We assume the full knowledge of the local events of each robot, which are defined to be

\[
\Sigma_1 = \{ h_1, R_1 to D_1, R_1 on D_1, Open, Close, R_2 in 1, R_1 to 3, R_1 in 3, D_1 closed, D_1 open, R_1 to 1, R_1 in 1, r \}
\]

\[
\Sigma_2 = \{ h_2, R_2 to 2, R_2 in 2, D_2 open, R_2 to 1, R_2 in 1, r \}
\]

\[
\Sigma_3 = \{ h_3, R_3 to 3, R_3 in 3, R_3 to D_1, R_3 on D_1, Open, Close, D_1 open, R_2 in 1, D_1 closed, R_3 to 3, R_1 in 1, r \}
\]

Collective behavior of the multi-robot team \( G \) is captured by the concurrent operations of each robot, i.e., \( G = \bigotimes_{i=1}^{3} G_i \). Define \( P_i : \Sigma^* \rightarrow \Sigma_i^* \) as the natural projection \[2\] for robot \( G_i \) with the inverse projection \( P_i^{-1} \). The collective behavior of the multi-agent system from a language perspective is given by the synchronous product of \( L(G_i) \):

\[
L(G) = \bigotimes_{i=1}^{3} L(G_i) = \bigcap_{i \in \mathbb{N}} P_i^{-1}(L(G_i)).
\]

### C. Robotic Coordination and Control Problem

For the purpose of supervisory control \[18\], \( \Sigma_i \) is partitioned into the set of (local) controllable events and the set of uncontrollable events, i.e., \( \Sigma_i = \Sigma_{i, uc} \cup \Sigma_{i, c} \). For the three robots, we assume that \( \Sigma_{1, uc} = \{ h_1, R_2 in 1 \} \), \( \Sigma_{3, uc} = \{ h_3, R_2 in 1 \} \) and \( \Sigma_{2, uc} = \{ h_2, D_1 open \} \).

A local supervisor \( S_i \) associated with \( G_i \) is another automaton that operates in parallel with \( G_i \), and the local controlled behaviors can then be modeled as \( L(S_i|G_i) \), where \( | \) stands for the parallel composition of automata \[2\].

Given a non-empty prefix-closed local specification \( K_i \subseteq L(G_i) \), a local supervisor \( S_i \) exists such that \( L(S_i|G_i) = K_i \) if and only if \( K_i \) is controllable with respect to \( L(G_i) \) and \( \Sigma_{i, uc}, i.e., \overline{K_i} \Sigma_{i, uc} \subseteq L(G_i) \subseteq \overline{K_i} \). If not, a supervisor is synthesized for the supremal controllable (also prefix-closed) sublanguage of \( K_i \), i.e., \( \sup C_i(K_i) := \sup C_{\Sigma_{i, uc}}(K_i) \).

In this example, we furthermore assume that any events shared by more than one robot agree on the status of controllability, i.e., \( \forall i, j \in \{1, 2, 3\}, i \neq j, \Sigma_{i, uc} \cap \Sigma_{j, c} = \emptyset \). The "global"
controllable and uncontrollable event sets are given by \( \Sigma_c = \bigcup_{i=1}^{3} \Sigma_{c,i} \) and \( \Sigma_{uc} = \Sigma - \Sigma_c \), respectively.

Two regular language specifications, denoted as \( K_s, K_d \subseteq \Sigma^* \) (subscripts \( s \) and \( d \) stand for “service” and “door”, respectively), that correspond to the informal performance requirements, are shown in Fig. 10 and 11 respectively, represented by DFA, where \( h_1 \| h_3 \) stands for the interleaving of single-event traces \( h_1 \) and \( h_3 \).

The overall specification for the multi-agent system is to satisfy \( K_s \) and \( K_d \) simultaneously, i.e., \( K = K_s \cap K_d \). Our design objective is to synthesize local supervisors \( S_i \) for Robot \( R_i \) such that \( \| L_{i=1}^{3} S_i \| R_i = K \), i.e., \( P_{\Sigma(K)}(L(\| L_{i=1}^{3} S_i \| R_i)) \subseteq K \), where \( P_{\Sigma(K)} \) is the projection from \( \Sigma = \bigcup_{i=1}^{3} \Sigma_i \) to \( \Sigma(K) \).

Our design objective is concerned with the following distributed control and coordination problem for the multi-agent systems.

\[ \text{Problem 1: Distributed Coordination and Control Co-design Problem (DCCCP): Consider the multi-robot team} \]

\[ G \text{ with robots} \ G_i \text{ and associated local controllable events} \Sigma_{i,c} \text{ and uncontrollable events} \Sigma_{i,uc}, \ i = 1, 2, 3. \] Given the performance requirement \( K \subseteq L(G) \):

1) systematically find local control specification \( K_i \) for each agent \( G_i \);

2) synthesize the local supervisor \( S_i \) for each \( G_i \) to achieve \( K_i \) using only the local control information \( G_i \) and \( \Sigma_i \), such that the controlled behaviors of the system, \( \| L_{i=1}^{3} S_i \| G_i \) \subseteq K, with no prior knowledge of \( G_i \)'s.

III. AUTOMATIC SYNTHESIS FRAMEWORK FOR MULTI-AGENT SYSTEMS

A. \( L^* \) learning-based Automatic Synthesis Framework

This section solves DCCCP under the case that \( G_i \) is not necessarily known. To compensate for the lack of prior model knowledge, \( L^* \) learning algorithm, which learns an unknown regular language \( U \) over the event set \( \Sigma \), is adapted to the coordination and control purposes. \( L^* \) creates a series of observation tables to incrementally record and maintain the information whether traces in \( \Sigma^* \) belong to \( U \). An observation table is a three-tuple \((S, E, T)\) consisting of: a non-empty finite set \( S \) of prefix-closed traces, a non-empty finite set \( E \) of suffix-closed traces and a Boolean function, called a membership query, \( T : (S \cup \Sigma \Sigma)E \rightarrow \{0, 1\} \). Once the observation table is closed and consistent [1], a candidate DFA \( M(S, E, T) = (Q, q_0, \delta, Q_m) \) over the alphabet \( \Sigma \) is constructed. If \( L(M) = U \), then the oracle returns “True” with the current DFA \( M \); otherwise, a counterexample \( c \in (U - L(M)) \cup (L(M) - U) \) is generated by the oracle. \( L^* \) then adds all its prefixes \( \tau \) to \( S \), which reflects the difference in the next conjecture by splitting states in \( M \), and \( L^* \) iterates the aforementioned process to update \( M \) with respect to \( S \).

We propose a two-layer learning-based framework to solve DCCCP for the multi-robot team as depicted in Fig. 3.

The automatic synthesis framework solves DCCCP by executing the following steps iteratively.

- **Task decomposition** Obtain a prefix-closed local specification by \( K_i = K_i \subseteq L(G_i) \) for robot \( G_i \).
- **Local synthesis** Given the local specification \( K_i \), we use a modified \( L^* \) algorithm \( L^*_i, S \) to automatically synthesize the local supervisors \( S_i \) such that \( L(S_i) \subseteq \Sigma_i \) for robot \( G_i \).
- **Compositional verification** After achieving local control specifications, we go one step further to check whether or not the collective behaviors of each agent can satisfy the global specification. For such purpose, a compositional verification [17] procedure is summoned by setting \( M_i = S_i \subseteq \Sigma_i \) be component modules, and \( P = K \) be the property to be checked.
- **Counterexample-guided synthesis** If the local behaviors do not satisfy the global specification jointly, then the compositional verification fails and provides a counterexample trace \( t \subseteq \Sigma^* \), which implies that all the \( M_i, i \in \mathcal{N} \) share a same illegal trace that violates property \( P \). We present such counterexample \( t \) back to the Task Decomposition by modifying the global specification from \( K \) to \( K - \bigcap_{i=1}^{n} P_i^{-1} P_i(t) \) and use the new specification to re-synthesize the local supervisors.
B. Task Decomposition

In the robotic coordination problem stated in Section II, local specifications \( K_i \), \( i = 1, 2, 3 \), are obtained by \( K_i = P_i(K) \). \( K_i \) can be implemented according to the following theorem.

Theorem 1: [23] Consider a multi-agent system \( G = \biguplus_{i=1}^{n} G_i \) that consists of \( n \) concurrent agents. If \( K \subseteq L(G) \) is a non-empty, prefix-closed specification, then \( K_i = P_i(K) \subseteq L(G_i) \).

Given the local specification \( G \), Fig. 4, and 5, we propose the following membership queries \( T_j^i \) for \( i \in I \) and \( j \in \mathbb{N} \) [4]. For \( t \in \Sigma_i \), let \( T_j^i(t) \) denote the \( i \)-th local membership Boolean function, initially,

\[
T_j^i(t) = \begin{cases} 
0, & \text{if } t \notin K_i \\
1, & \text{otherwise} 
\end{cases}
\]

For \( j > 1 \)

\[
T_j^i(t) = \begin{cases} 
0, & \text{if } T_{j-1}^i(t) = 0 \text{ or } t \in D_i(C_j^i)\Sigma_i^* \\
1, & \text{otherwise} 
\end{cases}
\]

The correctness and finite convergence properties of \( L_{LS}^* \) is guaranteed by the following theorem.

Theorem 2: [4] For \( K_i \subseteq L(G_i) \) being a prefix-closed local specification, then \( L_{LS}^* \) with membership queries (2) and (3) converges to a supervisor \( S_i \) such that \( L(S_i||G_i) = \sup C_i(K_i) \) within a finite number of counterexample tests.

We apply \( L_{LS}^* \) to the synthesis of local supervisors for each robot. For instance, a candidate supervisor \( S_2 \) for robot \( G_2 \) is demonstrated as follows.

C. \( L_{LS}^* \): Learning-based Synthesis of Local Supervisors

Given the local specification \( K_i \), we aim to synthesis a local supervisor \( S_i \) even no prior knowledge of \( G_i \) is given. The synthesis algorithm, namely \( L_{LS}^* \), is designed by adjusting \( L^* \) with modified dynamical membership queries that is capable of learning the supremal controllable sublanguage of the local specification such that \( L(S_i||G_i) = \sup C_i(K_i) \).

Construction of local supervisors depends on local illegal behaviors. A local behavior \( s \in \Sigma_i^* \) for the \( i \)-th agent is said to be \( i \)-locally uncontrollably illegal if \( s \in K_i \), \( t \in \Sigma_i^{\ast, uc} \) and \( st \notin K_i \). Let \( C_i \) denote the collection of observed locally uncontrollably illegal behaviors of the agent \( G_i \). We define \( D_i(\cdot) \) as \( D_i(C_i) = \{ s \in L(G_i) : \exists t \in \Sigma_i^{\ast, uc} \text{ such that } st \in C_i \} \) to represent the collection of the strings formed by discarding the uncontrollable suffixes of strings in \( C_i \), and let \( C_i^j \) denote the set of \( i \)-locally uncontrollably illegal behaviors after the \( j \)-th update, then if a new \( i \)-locally uncontrollably illegal behavior \( s_j \) is generated, we then update \( C_i^j \) to \( C_i^{j+1} = \{ s_j \} \cup C_i^j \). Finally, we apply \( D_i(\cdot) \) as \( D_i(C_i) = \{ s \in L(G_i) : \exists t \in \Sigma_i^{\ast, uc} \text{ such that } st \in C_i \} \) to represent the collection of the strings formed by discarding the uncontrollable suffixes of strings in \( C_i \). If the team specification \( K \) for the multi-robot team is separable [23], i.e., \( K = \biguplus_{i=1}^{3} P_i(K) \), then DCCCP can be solved trivially by synthesizing local supervisors independently with respect to local specifications \( K_i, i = 1, 2, 3 \) using \( L_{LS}^* \) algorithm. To evaluate the joint work of the local supervisors, a compositional verification procedure is summoned and we use the “assume-guarantee” paradigm to justify whether or not the global specification is satisfied. Moreover, to avoid “state explosion”, we use an “assume-guarantee” paradigm [17] for the verification.

1) Proof rules and assume-guarantee reasoning

If the team specification \( K \) for the multi-robot team is separable [23], i.e., \( K = \biguplus_{i=1}^{3} P_i(K) \), then DCCCP can be solved trivially by synthesizing local supervisors independently with respect to local specifications \( K_i, i = 1, 2, 3 \) using \( L_{LS}^* \) algorithm. To evaluate the joint work of the local supervisors, a compositional verification procedure is summoned and we use the “assume-guarantee” paradigm to justify whether or not the global specification is satisfied. Moreover, to avoid “state explosion”, we use an “assume-guarantee” paradigm [17] for the verification.

D. Compositional verification and assume-guarantee reasoning

If the team specification \( K \) for the multi-robot team is separable [23], i.e., \( K = \biguplus_{i=1}^{3} P_i(K) \), then DCCCP can be solved trivially by synthesizing local supervisors independently with respect to local specifications \( K_i, i = 1, 2, 3 \) using \( L_{LS}^* \) algorithm. To evaluate the joint work of the local supervisors, a compositional verification procedure is summoned and we use the “assume-guarantee” paradigm to justify whether or not the global specification is satisfied. Moreover, to avoid “state explosion”, we use an “assume-guarantee” paradigm [17] for the verification.

1) Proof rules and assume-guarantee reasoning

If the team specification \( K \) for the multi-robot team is separable [23], i.e., \( K = \biguplus_{i=1}^{3} P_i(K) \), then DCCCP can be solved trivially by synthesizing local supervisors independently with respect to local specifications \( K_i, i = 1, 2, 3 \) using \( L_{LS}^* \) algorithm. To evaluate the joint work of the local supervisors, a compositional verification procedure is summoned and we use the “assume-guarantee” paradigm to justify whether or not the global specification is satisfied. Moreover, to avoid “state explosion”, we use an “assume-guarantee” paradigm [17] for the verification.

1) Proof rules and assume-guarantee reasoning

If the team specification \( K \) for the multi-robot team is separable [23], i.e., \( K = \biguplus_{i=1}^{3} P_i(K) \), then DCCCP can be solved trivially by synthesizing local supervisors independently with respect to local specifications \( K_i, i = 1, 2, 3 \) using \( L_{LS}^* \) algorithm. To evaluate the joint work of the local supervisors, a compositional verification procedure is summoned and we use the “assume-guarantee” paradigm to justify whether or not the global specification is satisfied. Moreover, to avoid “state explosion”, we use an “assume-guarantee” paradigm [17] for the verification.
due to \( R_2 \)'s distinct capabilities. Based on this structure, we first define \( M_{-2} = M_1 || M_3 \), and use the assume-guarantee proof rule with respect to \( M_2 \) and \( M_{-2} \) to obtain the appropriate assumption \( A_{-2} \) for \( M_{-2} \) such that \( M_2 || M_{-2} \models K \); secondly, we use the proof rules once more with respect to \( M_1 \) and \( M_3 \) to verify if \( M_{-2} = M_1 || M_3 \models A_{-2} \), which is summarized as follows.

\[
\begin{align*}
1. & \quad (A_{-2})M_2(K) \\
2. & \quad (A_3)M_1(A_{-2}) \\
3. & \quad (true)M_3(A_3)
\end{align*}
\]

\( (true)M_1||M_2||M_3(P) \)

Solving DCCCP relies on two aspects: first, we use a second modified \( L^* \) procedure, called \( L^*_CV \) to generate appropriate assumptions associate with each module \( M_i \), \( i \in N \); secondly, if the verification fails, we have to analyze the counterexample provided by the model checker to determine whether or not the concurrent operations of \( M_i \), \( i \in N \) indeed violate the property \( P \).

2) \( L^*_CV \): learning assumptions for compositional verification: The correctness of the learning loop for solving compositional verification is studied. The following theorem guarantees that the circular proof rules are sound and complete.

**Theorem 3:** [17] The asymmetric and circular assume-guarantee proof rules are sound and complete.

At each iteration, \( L^*_CV \) is used to learn appropriate assumptions for each component module \( M_i \), based on querying the system and on the results of the previous iteration. The construction of \( L^*_CV \) relies on the following membership queries for the \( L^*_CV \). For \( t \in \Sigma \),

\[
T_i(t) = \begin{cases} 
0, & \text{if } L(DFA(A)|M_i) \not\subseteq P \\
1, & \text{otherwise}
\end{cases}
\]

\[
(3)
\]

It is shown in [5] that \( L^*_CV \) is correct and convergent.

We take the learning process of an appropriate assumption for robot 2. The learning-based compositional verification algorithm \( L^*_CV \) answers the conjecture \( A_{-2} \) by checking \( \langle A_{-2}\rangle M_2\langle K \rangle \), and it turns out that \( A_{-2} \) is an appropriate assumption for \( M_2 \). Then the assumption \( A_{-2} \) can serve as the “property” for the composed system \( M_1||M_3 \) to be satisfied.

To reduce the computational complexity, we use the following \( A^{ref}_{-2} \) as a refined assumption for \( M_1||M_3 \), indicating that \( R_1 \) and \( R_3 \) should open and close \( D_1 \) from Room 1 and 3 respectively. Note that \( A^{ref}_{-2} \models A_{-2} \), which suggests that, by using AG-NC rules, if \( M_1||M_3 \models A^{ref}_{-2} \), then \( M_1||M_2||M_3 \models K \).

3) Counterexample analysis: If Oracle 3 returns true, then the framework returns that \( |\{3\}M_i \models P \) and therefore the DCCCP synthesis succeeds. If Oracle 3 returns false, it provides a counterexample \( t \in \Sigma^* \) that violates \( P \). We simulate \( t \) on \( M_i|coP \) to check whether \( t \) is a violating trace of all components \( M_i \), \( i = 1, 2, 3 \). If \( t \) is not a violating trace for at least one component \( M_i \), then we use \( t \) to modify the corresponding assumptions, otherwise \( \{3\}M_i \) indeed violates \( P \).

Different from verification problems, the framework proposed in this paper takes advantage of the counterexample to re-synthesize the local supervisors by eliminating all the “indistinguishable” traces with respect to \( t \) from the global specification \( K \), and let \( K := K - \cap_{i=1}^3 P_i^{-1}t_i(t) \) be the new global specification.

The following theorem asserts the correctness and convergence of the proposed DCCCP synthesis framework.

**Theorem 4:** [5] The framework depicted in Fig. 3 will eventually synthesize local supervisors \( S_i \) such that collective behaviors of the multi-agent system achieve a separately controllable sublanguage of \( K \).

### IV. EXPERIMENTAL RESULTS

This section reviews and solves the DCCCP of the motivating example. The overall specification for the multi-agent system is to satisfy \( K_s \) and \( K_d \) simultaneously, i.e., \( K = K_s \cap K_d \).

#### A. Theoretical Results

Through the automatic supervisor synthesis framework, the learned supervisors are as shown in Fig. 9, 10 and 11 respectively.

![Fig. 8: Assumption \( A^{ref}_{-2} \).](image)

![Fig. 9: Local supervisor \( S_1 \).](image)

![Fig. 10: Local supervisor \( S_2 \).](image)
B. Experimental Setup

The proposed supervisors, implemented as MatLab StateFlow machines, have been tested and evaluated on a swarm composed by three small robots. The experimental setup for the testing of the derived supervisors is composed by three main elements, described as follows.

1) Localization System: The localization system is based on the Aruco library [25] and is able to provide the status of each robot (position and heading) and the status of each door (open or close) with a frame-rate of $\sim 20\text{Hz}$. The information concerning the position of the robots and the status of each door is codified using the JSON format. The software is based on a multi-threading architecture:

- The main thread implements a UDP server waiting for information requests from the robots.
- The second thread computes the position of each marker and updates the list of detected markers along with their position and orientation.

The sharing of data between the threads is based on a mutex locking mechanism. Figure 12 shows the output of the localization system. As examples, the position of the robot 1, along with the status of the three doors is represented as follows:

{"R1":{"+0460.86","+0113.43","+0079.15","0","0","1"}}.

2) Local supervisors: Each of the three supervisors is implemented as StateFlow machine and manages a predefined robot of the swarm.

The logic implemented follows the theoretical approach described in section III: The transition between the states is based on the occurrence of one of the possible pre-defined events, like the opening of a door. The communication between the StateFlow machine and the corresponding robot is based on a wireless communication system based on a UART over IP module. The TCP protocol is used to provide a reliable exchange of messages between the state flow and the robot. Figures 13 and 14 show the StateFlow implementation of the Local supervisor S2.

3) Ground Robots: The fleet is composed by three small ground vehicle with the same hardware characteristics. The hardware is designed as a multi-layer architecture:

- The low-level system (LL board), based on xMOS processors, is responsible to gather the data from the sensors and to apply the control action to the actuators (Figure 15). The xMOS processor provides event-driven, hard real-time performances and up to 4 parallel threads at 100 MIPS. The hard real-time scheduling of the tasks running on the xMOS devices is guaranteed regardless of the state of other threads. Furthermore, each xMOS processor has up to 64 I/O pins that can provide 10ns timing resolution. This allows for the
creation of software-defined peripherals. The software for the management of the xMOS devices has been developed by the authors and it is based on [26]. Thanks to the hard-real time performance of the xMOS devices, the library provides deterministic real-time access to all the sensors and actuators connected to the processor.

- The high-level system (HL board) is based on the BeagleBone board and is responsible to interact with the corresponding local supervisor (Figure 16). The goal of the high-level board is to provide a system for the development of high-level control system such as obstacle avoidance, delegating the low-level real-time board for the interaction with the sensors.

A unique marker (Figure 17) for each robot is used for detecting its position using the developed localization system (Figure 12). The communication system between the two hardware layer is realized using a serial port. The data exchanged are also codified using the JSON format.

C. Experimental Results

The flow chart in Figure 18 describes the interaction between the supervisor and the corresponding ground robot. Since each supervisor manages only one robot (agent), the three supervisors are implemented as distinct stateflow machines running simultaneously.

The initialization of the system is carried out using an ad-hoc application that sends the start signal to all the agents (using the UDP protocol).

After the reception of the START signal, the HL board of the agent initializes the communication system and sends the first request to the supervisor. The supervisor responds sending the command back to the HL board, which interacts with the LL board to accomplish the task. When the task is completed, the HL board sends the ACK back to the Supervisor which responds with the next action.

The path followed by robot 2, as defined by the local supervisor S2 is shown in Figure 19.
The framework is illustrated through multi-robot coordination. Agents’ models are not given a priori by using compositional verification techniques even if the synthesis of local supervisors and check its correctness robot is expressed in pixel coordinates. The position of the robot is sent from the local supervisor. The position of robot 2 reached a predefined waypoint and waited for the next cloud of points corresponds to the instants in which the robot 2 followed. The actual path shown in Figure 20 corresponds to the instants in which the robot 2 reached a predefined waypoint and waited for the next command sent from the local supervisor. The position of the robot is expressed in pixel coordinates.

V. CONCLUSION

We follow the top-down design approaches and present a distributed coordination and control framework for multi-agent systems that can be modeled as a concurrent discrete-event systems accounting for the collective behaviors of individual agents. By modifying the $L^*$ algorithm, the framework synthesizes the local supervisors and check its correctness by using compositional verification techniques even if the agents’ models are not given a priori. Effectiveness of the framework is illustrated through multi-robot coordination.

REFERENCES

[1] D. Angluin, “Learning regular sets from queries and counterexamples,” Int. J. Inform. and Computation, vol. 75, no. 1, pp. 87-106, 1987.
[2] C. G. Cassandras, S. Lafortune, Introduction to Discrete Event Systems, USA: Springer, 2008.
[3] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, “Formal approach to the deployment of distributed robotic teams,” IEEE Trans. Robot., vol. 28, no. 1, pp. 158-171, 2012.
[4] J. Dai and H. Lin, “A learning-based synthesis approach to decentralized supervisory control of discrete event systems with unknown plants,” Control Theory Tech., vol. 12, no. 3, pp. 218-233, 2014.
[5] J. Dai and H. Lin, “Automatic synthesis of cooperative multi-agent systems,” Proc. 53rd IEEE Conf. Decision Control, pp. 6175-6178, 2014.
[6] D. V. Dimarogonas and K. J. Kyriakopoulos, “On the rendezvous problem for multiple nonholonomic agents,” IEEE Trans. Autom. Control, vol. 52, no. 5, pp. 916-922, 2007.
[7] X. Ding, M. Kloetzer, Y. Chen, and C. Belta, “Automatic deployment of robotic teams,” IEEE Robot. Autom. Mag., vol. 18, no. 3, pp. 75-86, 2011.
[8] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, “Temporal logic motion planning for dynamic robots,” Automatica, vol. 45, no. 2, pp. 343-352, 2009.
[9] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, “Where’s Waldo? Sensor-based temporal logic motion planning,” Proc. IEEE Int. Conf. Robot. Autom., Rome, Italy, pp. 3116-3121, 2007.
[10] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, “Temporal-logic-based reactive mission and motion planning,” IEEE Trans. Robot., vol. 25, no. 6, pp. 1370-1381, 2009.
[11] M. Karimadini and H. Lin, “Guaranteed global performance through local coordinations,” Automatica, vol. 47, no. 5, pp. 890-898, 2011.
[12] M. Kloetzer and C. Belta, “Automatic deployment of distributed teams of robots from temporal logic motion specifications,” IEEE Trans. Autom. Control, vol. 53, no. 1, pp. 48-61, 2008.
[13] R. Kumar and V. K. Garg, Modeling and Control of Logical Discrete Event Systems, Boston: Kluwer, 1995.
[14] S. M. LaValle, Planning Algorithms, Cambridge, U.K.: Cambridge Univ. Press, 2006.
[15] R. Olfati-Saber, “Flocking for multi-agent dynamic systems: Algorithms and theory,” IEEE Trans. Autom. Control, vol. 51, no. 3, pp. 401-420, 2006.
[16] R. Olfati-Saber and R. M. Murray, “Consensus problems in networks of agents with switching topology and time-delays,” IEEE Trans. Autom. Control, vol. 49, no. 9, pp. 1520-1533, 2004.
[17] C. Păsăreanu, D. Giannakopoulou, M. G. Bobaru, J. M. Cobleigh, and H. Barringer, “Learning to divide and conquer: applying the $L^*$ algorithm to automate assume-guarantee reasoning,” Formal Methods Syst. Des., vol. 32, no. 3 pp. 175-205, Springer U.S., 2008.
[18] F. J. Ramadge and W. M. Wonham, “Supervisory control of a class of discrete event processes,” SIAM J. Control Optim., vol. 25, no. 1, pp. 206-230, 1987.
[19] W. Ren, “On consensus algorithms for fo double-integrator dynam- ics,” IEEE Trans. Autom. Control, vol. 53, no. 6, pp. 1503-1509, 2008.
[20] K. T. Seow, M. T. Pham, C. Ma, and M. Yokoo, “Coordination planning: Applying control synthesis methods for a class of distributed agents,” IEEE Trans. Control Syst. Technol., vol. 17, no. 2, pp. 405-415, 2009.
[21] K. P. Valavanis and G. N. Saridis, Intelligent robotic systems: theory, design and applications, vol. 182, Springer Science & Business Media, 2012.
[22] Wei, Q., Liu, D. and Lewis, F.L., 2015. Optimal distributed synchroniz- tion control for continuous-time heterogeneous multi-agent differential graphical games. Information Sciences, 317, pp.96-113.
[23] Y. Willner and M. Heymann, “Supervisory control of concurrent discrete-event systems,” Int. J. Contr., vol. 54, no. 5, pp. 1143-1169, 1991.
[24] H. Zhang, F. L. Lewis, and A. Das, “Optimal design for synchronization of cooperative systems: state feedback, observer and output feedback.” IEEE Trans. Autom. Control, vol. 56, no. 8, pp. 1948-1952, 2011.
[25] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco J. Madrid-Cuevas, Rafael Medina-Carnicer, “Generation of fiducial marker dic- tions using Mixed Integer Linear Programming” Journal of Pattern Recognition, vol. 47, no. 6, pp. 2280-2292, 2014.
[26] Goncalo Martins, Allistair Moses, Matthew J Rutherford, Kimon P Valavanis, “Enabling intelligent unmanned vehicles through XMOS Technology,” The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, vol. 9, pp. 71-82, 2012.
[27] G. Martins and D. Lacey and A. Moses and M. J. Rutherford and K. P. Valavanis,”A case for I/O response benchmarking of microprocessors, IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, pp. 3018-3023, 2012.