Need-based Coordination for Decentralized High-level Robot Control

Paper 43

ABSTRACT

We consider multiple robots operating in a shared workspace, each given a high-level task specification in the form of Linear Temporal Logic (LTL) formulas. The robots have no a priori knowledge about the tasks of the other robots and might run into conflicts during task execution. In this work, we develop algorithms that allow the robots to autonomously resolve conflicts and complete their tasks with correctness guarantees, when possible.

In this approach, the robots autonomously detect conflicts and trigger coordination within the subgroup of robots in conflict. The need-based coordination is achieved through executing a global robot controller on each robot in the subgroup until the robots' current goals are reached. The subgroup of robots then return to their local controllers and continue their execution.

Categories and Subject Descriptors

F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Temporal logic*; I.2.2 [**Artificial Intelligence**]: Automatic Programming—*Program modification, Program synthesis*; I.2.9 [**Artificial Intelligence**]: Robotics; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and coordination, Intelligent agents, Multiagent systems*

Keywords

Algorithms, Reliability, Verification

General Terms

Robotics, Controller Synthesis, Multi-agent control, Hybrid systems, Linear Temporal Logic

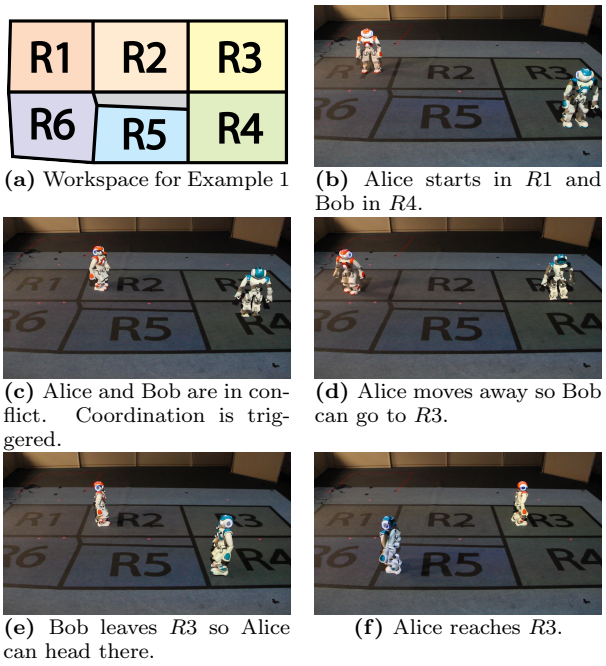
1 Introduction

Researchers have been interested in the automatic synthesis of provably correctly robot controllers from high-level task specifications [3, 4, 12, 25]. Compared with the manual construction of robot controllers, the synthesis of robot controllers is automatic starting from high-level task specifications. These controllers, if successfully generated, are guaranteed to satisfy the requirements in the specifications.

Recent work has used formal methods in multi-robot missions [11, 14, 15, 24]. In these scenarios, one can model the other robots as part of its environment and make assumptions about the neighbor robots' behaviors in order to achieve one's task.

When several robots are operating in a shared environment, even though each robot is executing a correct-by-construction controller, possible deadlock and livelock conditions emerge if the other robots do not behave as presumed.

In this work, we assume that each robot can detect where the other robots are and their completed actions, but each robot cannot affect the other robots' behavior. Consider the following example with two robots, Alice and Bob, operating in the workspace as shown in Fig. 1a.



(a) Workspace for Example 1

(b) Alice starts in $R1$ and Bob in $R4$.

(c) Alice and Bob are in conflict. Coordination is triggered.

(d) Alice moves away so Bob can go to $R3$.

(e) Bob leaves $R3$ so Alice can head there.

(f) Alice reaches $R3$.

Figure 1: Alice, the orange Aldebaran Nao, and Bob, the blue Aldebaran Nao, performing their tasks as described in Example 1.

Example 1. Alice starts in $R1$ and Bob starts in $R4$ (Fig. 1b). The robots should always maintain a distance of at least two regions from each other. For example, while Bob is in $R4$, Alice can only stay in $R1$, $R2$ and $R6$. The robots have the same goal: go to $R3$. Not knowing the behavior of the other robot, each robot assumes that they are always three regions apart. For instance, Alice assumes that when she is in $R2$, Bob is in $R5$.

In Example 1, the provably-correct controllers for Alice and Bob can be generated, as the tasks can be accomplished

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), John Thangarajah, Karl Tuyls, Stacy Marsella, Catholijn Jonker (eds.), May 9–13, 2016, Singapore.*

under the assumptions made about the other robot. However, during execution, as both robots head to $R3$, each robot can violate the assumptions the other robot made. For example, if Alice moves from $R1$ to $R2$ and Bob is in $R4$, then the assumption that they are always three regions apart, "when Alice is in $R2$, Bob is in $R5$", is violated (Fig. 1c). The assumptions used to generate the controllers are violated, therefore the robots are not longer guaranteed to perform correctly.

In this work, we propose a new approach that allows the robots to continue and accomplish their tasks through coordination. When a conflict is detected, a combined spec is automatically created for each robot and a global controller is synthesized with the current goals pursued by the robots. With our approach, the previous example is resolved by the robots taking turns to visit $R3$ and the tasks of the two robots are satisfied one after the other (Fig. 1d-1f). Once the goals are achieved, the robots return to their individual controllers and the task execution continues.

Example 1 is a simple illustration of the approach. The approach is general and can be applied to many complex high-level tasks (See Section 6) ranging from robot manipulation (Section 6.2), request fulfillment (Section 6.1) to operation in hazardous environment with more than two robots. Our proposed approach combines the advantage of decentralized robot control in which unnecessary information and the status of the other robots are omitted, reducing the state space of the decentralized controller; and the advantage of centralized robot control in which coordination of the robots is possible. The transition between centralized and decentralized high-level control of robot behaviors is seamless in our approach with guarantees provided during execution.

Related Work

Multi-robot systems pose a variety of problems, among which the multi-robot task allocation (MRTA) problem, the problem of finding an optimal task allocation for a team of robots, has been explored by researchers [6,8] over the years. Techniques from task swapping [13] to tackling uncertain situations [16] were developed.

Instead of assigning single-robot-tasks to different robots, another approach is to divide a multi-robot task into subtasks and partition the set of robots into different teams, known as coalitions, to perform each task [21,22]. As such, potential robot coalitions are analyzed to determine feasibility [26] with coordination mechanism generalized [9].

Once we have a policy for each robot, the low-level robot locomotion and the high-level task execution can be guaranteed. When deployed at runtime, a team of heterogeneous robots can be controlled with decentralized feedback motion controllers in a known environment with obstacles [1]. A team of robots may respond to motion inputs from humans with goal convergence and safety guaranteed [2]. Alternatively, the motion plan of a group of robots is represented as a set of safe linear temporal logic (LTL) formulas and solved with Satisfiability Modulo Theories (SMT) [20].

Current work on providing task-level guarantees uses centralized approaches [10,17]. A centralized controller synthesis framework for multiple heterogeneous robots was proposed [18], with the synthesis of robot controllers together with robot motion commands also developed [15]. In our work, we leverage the advantage of centralized robot controllers where subgroups of robots are under global control when the robots cannot continue their missions without coordination.

In order to avoid the high cost of computing centralized controllers, decomposing the global specification and decentralized robot control have been introduced, such as automatically generating decentralized strategies and communication for a group of robots from a global high-level task specification [11]. The synthesis and merger of a joint strategy into two local robot controllers [14] is also available, but it requires synchronous execution and multiple syntheses may occur before the robots reach their current goals. The negotiation approach with two robots [24] only modifies one robot's plan instead of both when conflict arises, and it does not resolve the situation in Example 1. Our work allows the robots to execute asynchronously leveraging the work from [19], and coordinates more than two robots through a temporary execution of global controllers for the robots in conflict.

The outline of this paper is as follows: We define the necessary preliminaries in Section 2. In Section 3, we describe the specifications of Example 1. In Section 4, we motivate and explain the problem being solved in this paper and in Section 5, we describe our approach. Section 6 illustrates and discusses our approach using two examples. In Section 7, we conclude our paper with possible future directions.

2 Preliminaries

Consider $\mathcal{R}$ to be a set of robots operating in a shared workspace. For each robot $r \in \mathcal{R}$,

Definition 1. **(Atomic Propositions)** Each robot r is capable of a variety of actions and its location can be represented in terms of regions in its workspace. For example, in Example 1, Alice is in $R1$ when her task begins. The surroundings may change as each robot conducts its task and this can be modeled. For instance, from Alice's perspective, Bob is part of the environment and his location can change.

The relevant continuous properties of the robot and the environment are abstracted into a set of atomic propositions $AP_r = \mathcal{X}_r \cup \mathcal{Y}_r$.

- $\mathcal{Y}_r = Act_r \cup Reg_r \cup Mem_r$ is the set of system propositions abstracted from the robot behaviors. Act_r represents the set of actions that can be actuated by the robot. Reg_r consists of the set of regions in the workspace. If $\pi_{R_i} \in Reg_r$ is true, then the robot is currently heading to region R_i in the workspace. Only one element in Reg_r is true at all times. Mem_r consists of the set of propositions used to remember events in execution. In Example 1, $\pi_{R1} \in Reg_A$ is true, i.e., Alice temporarily stays in place when she starts, with no actions started.
- $\mathcal{X}_r = envAct_r \cup envReg_r \cup \{Act_r^c \cup Reg_r^c\} \cup Sen_r \cup Sen_{\mathcal{G}}$ is the set of environment propositions based on the abstraction of environment and robot behaviors. $envAct_r$ represents the completion status of the other robots' actions. $envReg_r$ represents the other robots' current location in the workspace. In Example 1, $\pi_{R4}^B \in envReg_A$ is true as Bob starts in $R4$ when Alice begins. $\{Act_r^c \cup Reg_r^c\} \subseteq \mathcal{X}_r$ contains the same number of elements as $\{Act_r \cup Reg_r\} \subseteq \mathcal{Y}_r$, except $\pi_{R_i}^c \in Reg_r^c$ now represents the current location of the robot if $\pi_{R_i}^c$ is true; $\pi_A^c \in Act_r^c$ represents the completion of robot action A if π_A^c is true. Same as Reg_r, only one element

in Reg_r^c is true at all times. In Example 1, $\pi_{R1}^c \in Reg_A^c$ is true when Alice starts, with no actions completed. Sen_r is the set of abstracted robot sensors while $Sen_\mathcal{G}$ is the set of abstracted global sensors. Global sensors are sensors with status accessible to all robots operating in the workspace. There are no robot sensors or global sensors in Example 1.

Definition 2. **(Robot Controller)** The controller of each robot r in this work is a finite state nondeterministic automaton $\mathcal{A}_r = (\mathcal{X}_r, \mathcal{Y}_r, \mathcal{S}_r, \mathcal{S}_{0_r}, \delta_r^s, \gamma_r^s)$, where:

$\mathcal{X}_r$ and $\mathcal{Y}_r$ are the set of environment and system propositions respectively. $\mathcal{S}_r$ is the set of all states. $\mathcal{S}_{0_r} \subseteq \mathcal{S}_r$ is the set of initial states. $\delta_r^s : \mathcal{S}_r \times 2^{\mathcal{X}_r} \rightarrow 2^{\mathcal{S}_r}$ is the transition function that given a state and a subset of environment propositions, outputs a set of next states. $\gamma_r^s : \mathcal{S}_r \rightarrow 2^{\mathcal{X}_r} \times 2^{\mathcal{Y}_r}$ is the state labeling function that outputs the subset of propositions true in a state.

An automaton execution, or a trace, is an infinite sequence of states $\sigma_r = s_r^0, s_r^1, \dots$, where $s_r^0 \in \mathcal{S}_{0_r}$ and $s_r^i \in \delta_r^s(s_r^{i-1}, x_r^i)$ for $i \geq 1 \in \mathbb{N}$ with $x_r^i = \gamma_r^s(s_r^i) \cap \mathcal{X}_r$.

The set of all states S_r can also be presented in the form of an LTL formula (See Def. 3) as W_r.

Definition 3. **(Linear Temporal Logic (LTL))** LTL describes sequences of events and are defined recursively over a finite set of atomic propositions AP_r:

$$\varphi ::= \pi \in AP_r \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc \varphi \mid \varphi \,\mathcal{U}\, \varphi$$

LTL includes boolean operators such as disjunction ($\vee$) and negation ($\neg$), which derive the conjunction ($\wedge$), implication ($\rightarrow$) and biimplication ($\leftrightarrow$) operators. LTL also includes temporal operators "always" ($\Box$), "finally" ($\Diamond$) and "next" ($\bigcirc$). The "always" ($\Box$) and "finally" ($\Diamond$) operators are derived from the "until" ($\mathcal{U}$) operator.

A trace σ_r satisfies φ, written as $\sigma_r \models \varphi$, if φ holds at the first state s_r^0 in σ_r. A trace σ_r satisfies $\bigcirc \varphi$ ($\sigma_r \models \bigcirc \varphi$) if φ holds at the next state s_r^1 in trace σ_r. A trace σ_r satisfies $\Diamond \varphi$ ($\sigma_r \models \Diamond \varphi$) if at some state s_r^j in the trace φ is satisfied, with $j \geq 0$. A trace σ_r satisfies $\Box \varphi$ ($\sigma_r \models \Box \varphi$) if φ is satisfied in s_r^0 and every state in the trace σ_r. An automaton $\mathcal{A}_r$ satisfies φ if for all traces σ_r of $\mathcal{A}_r$, $\sigma_r \models \varphi$.

Definition 4. **(Mission Specification)** In this paper, the specifications are expressed using a subset of LTL formulas known as Generalized Reactivity (1) fragment [5]. Each specification is of the form:

$$\begin{aligned} \varphi_r &= \varphi_{e,r} \rightarrow \varphi_{s,r} \\ &= \varphi_{e,r}^i \wedge \varphi_{e,r}^t \wedge \varphi_{e,r}^g \rightarrow \varphi_{s,r}^i \wedge \varphi_{s,r}^t \wedge \varphi_{s,r}^g \end{aligned} \tag{1}$$

$\varphi_{e,r}^i$ and $\varphi_{s,r}^i$ are the environment and robot initial conditions respectively. Each condition is a Boolean formula defined over propositions in $\mathcal{X}_r \cup \mathcal{Y}_r$.

$\varphi_{e,r}^t$ is the environment safety assumptions. Each assumption in $\varphi_{e,r}^t$ is a formula of the form $\Box\, \psi_{e,r}^t$, where $\psi_{e,r}^t$ is a Boolean formula defined over $\mathcal{X}_r \cup \mathcal{Y}_r \cup \bigcirc \mathcal{X}_r$. $\varphi_{s,r}^t$ is the system safety guarantees. Each guarantee in $\varphi_{s,r}^t$ is a formula of the form $\Box\, \psi_{s,r}^t$, where $\psi_{s,r}^t$ is a Boolean formula defined over $\mathcal{X}_r \cup \mathcal{Y}_r \cup \bigcirc \mathcal{X}_r \cup \bigcirc \mathcal{Y}_r$. The environment safety assumptions $\varphi_{e,r}^t$ contains assumptions about the environment behaviors expected to hold at all times. If these requirements are always satisfied, then the system safety guarantees $\varphi_{s,r}^t$ has to always hold during execution. In Example 1, 'each robot assumes that they are always three regions apart' is an environment safety assumption; 'the robots should always maintain a distance of at least two regions from each other' is a system safety guarantee. The LTL formulas that specify the two requirements are shown in Section 3.

$\varphi_{e,r}^t$ is the environment livenesses while $\varphi_{s,r}^g$ is the system livenesses, which are also known as system goals. These two formulas specify the robot and the environment behaviors that should be satisfied infinitely often. Each liveness or goal is a formula of the form $\Box\Diamond\, \psi_{e,r}^g$ or $\Box\Diamond\, \psi_{s,r}^g$, with $\psi_{e,r}^g$ and $\psi_{s,r}^g$ being a Boolean formula defined over $\mathcal{X}_r \cup \mathcal{Y}_r \cup \bigcirc \mathcal{X}_r \cup \bigcirc \mathcal{Y}_r$. Examples of these formulas are given in Section 3.

From a specification, a controller $\mathcal{A}_r$ can automatically be generated, if one exists. A specification φ_r is realizable if a controller $\mathcal{A}_r$ satisfying φ_r can be generated; the specification φ_r is unrealizable otherwise. More details on the synthesis of a controller $\mathcal{A}_r$ from a specification φ_r can be found in [5] and [7] .

The controller synthesis from a specification in this work leverages the cooperative implementations in [7]. Due to the structure of Eq. 1 which contains an implication operator, the synthesis algorithm in [5] can generate a controller $\mathcal{A}_r$ that falsifies the environment specification $\varphi_{e,r}$ with the system specification $\varphi_{s,r}$ not satisfied. That is undesirable and the cooperative implementation in [7] ensures that if the specification φ_r is realizable, then in the controller $\mathcal{A}_r$ the environment satisfies its assumptions and livenesses. Since $\varphi_r = \varphi_{e,r} \rightarrow \varphi_{g,r}$, the system requirements $\varphi_{g,r}$ in φ_r are also satisfied in $\mathcal{A}_r$ if φ_r is realizable.

The specifications considered in this work follow the activation and completion paradigm as described in [19]. In this paradigm, each robot behavior is two-fold: the initiation of each robot actuation is abstracted into a system proposition, with the completion of an action sensed by the environment and is abstracted into an environment proposition. For example, when Example 1 starts, Alice locates in $R1$ so π_{R1}^c is true. Alice is activating $R1$ (π_{R1} is true) as she stays in place. Later on, Alice heads to region $R2$ and π_{R2} turns true, but since Alice is still in $R1$, π_{R1}^c is true.

3 Example 1 Revisited

In this section, the robots' specifications in Example 1[1] are outlined. Two partial specifications, one for each robot, are shown in Spec. 1 and 2.

Specification 1 Alice's specification φ_A for Example 1

1 $\varphi_{s,A}^i = \pi_{R1}$
2 $\varphi_{e,A}^i = \pi_{R1}^c \wedge \pi_{R4}^B$
3 $\varphi_{s,A}^g = \Box\Diamond(\bigcirc \pi_{R3} \wedge \bigcirc \pi_{R3}^c)$

Specification 2 Bob's specification φ_B for Example 1

1 $\varphi_{s,B}^i = \pi_{R4}$
2 $\varphi_{e,B}^i = \pi_{R4}^c \wedge \pi_{R1}^A$
3 $\varphi_{s,B}^g = \Box\Diamond(\bigcirc \pi_{R3} \wedge \pi_{R3}^c)$

In Spec. 1, $\{\pi_{R1}, \pi_{R3}\} \subseteq Reg_A$, $\{\pi_{R1}^c, \pi_{R3}^c\} \subseteq Reg_A^c$ and $\pi_{R4}^B \in envReg_A$. Alice starts in $R1$ (Line 1, 2 in Spec. 1) while Bob starts in $R4$ (Line 2 in Spec. 1). Her goal, or system liveness, is to go to $R3$ (Line 3 in Spec. 1). Line 1 and 2 are the environment and robot initial conditions respectively. In Spec. 2, $\{\pi_{R4}, \pi_{R3}\} \subseteq Reg_B$, $\{\pi_{R4}^c, \pi_{R3}^c\} \subseteq Reg_B^c$ and $\pi_{R1}^A \in envReg_B$. Bob starts in $R4$ (Line 1, 2 in Spec. 2) while Alice starts in $R1$ (Line 2 in Spec. 2). His goal is to go to $R3$ (Line 3 in Spec. 2).

In this work, the robots are assumed not to occupy the same region at any time during execution, as we are using simple low-level motion controllers that cannot handle

[1] An execution of Example 1 in Section 1 and 3, Example 2 in Section 6.1 and Example 3 in Section 6.2 can be found in the video: `https://youtu.be/QKk7HZZkbtA`.

navigation of multiple robots in the same region. This assumption, however, can easily be relaxed by replacing the motion controllers used. This property is encoded in each robot's specification as a set of system safety guarantees:

$$\varphi_{s,r}^{t[mutual]} = \bigwedge_{r' \in \mathcal{R}\setminus\{r\}} \bigwedge_{R_i \in Regions} \square(\bigcirc \pi_{R_i}^{r'} \rightarrow \neg \bigcirc \pi_{R_i}^{c}) \quad (2)$$

where $\pi_{R_i}^{r'} \in envReg_r$ and $\pi_{R_i}^{c} \in Reg_r^c$. $\pi_{R_i}^{r'}$ represents robot r' is currently in region R_i. $Regions$ is the set of regions in a given workspace.

In each specification, two other assumptions are made about the other robots: (i) the other robots can only be in one region at a time (Eq. 3); (ii) the topological constraints of the other robots are the same as the system robot (Eq. 4).

$$\varphi_{e,r}^{t[region]} = \bigwedge_{r' \in \mathcal{R}\setminus\{r\}} \square \bigvee_{R_i \in Regions} (\bigcirc \pi_{R_i}^{r'} \bigwedge_{R_j \in Regions\setminus\{R_i\}} \neg \bigcirc \pi_{R_j}^{r'}) \quad (3)$$

$$\varphi_{e,r}^{t[adj]} = \bigwedge_{r' \in \mathcal{R}\setminus\{r\}} \bigwedge_{R_i \in Regions} \square(\pi_{R_i}^{r'} \rightarrow \bigcirc \pi_{R_i}^{r'} \bigvee_{R_j \in Adj(R_i)} \bigcirc \pi_{R_j}^{r'}) \quad (4)$$

where $Adj(R_i)$ is a function that returns the set of regions adjacent to region R_i in the workspace.

In Example 1, each robot maintains a distance of at least two regions with the other robot (Line 1-6 in Spec. 3). It is also assumed that the other robot always maintains a distance of three regions with each robot (Line 7-9 in Spec. 3). This assumption is needed to synthesize a controller for each robot. Without this assumption, no controller can be generated, as the environment robot has no constraints and can fail to maintain a distance of at least two regions, the system safety requirement. For Spec. 3, in Alice's specification, $r = A$ and $r' = B$; in Bob's specification, $r = B$ and $r' = A$.

Specification 3 Additional specification for each robot r in Example 1

$$\begin{aligned}
1\ &\varphi_{e,r}^{t[maintain\ Distance]} = \square(\bigcirc\pi_{R1}^{r'} \rightarrow \neg(\bigcirc\pi_{R1} \vee \bigcirc\pi_{R6} \vee \bigcirc\pi_{R2})) \wedge \\
2\ &\square(\bigcirc\pi_{R2}^{r'} \rightarrow \neg(\bigcirc\pi_{R2} \vee \bigcirc\pi_{R1} \vee \bigcirc\pi_{R3})) \wedge \\
3\ &\square(\bigcirc\pi_{R3}^{r'} \rightarrow \neg(\bigcirc\pi_{R3} \vee \bigcirc\pi_{R2} \vee \bigcirc\pi_{R4})) \wedge \\
4\ &\square(\bigcirc\pi_{R4}^{r'} \rightarrow \neg(\bigcirc\pi_{R4} \vee \bigcirc\pi_{R3} \vee \bigcirc\pi_{R5})) \wedge \\
5\ &\square(\bigcirc\pi_{R5}^{r'} \rightarrow \neg(\bigcirc\pi_{R5} \vee \bigcirc\pi_{R4} \vee \bigcirc\pi_{R6})) \wedge \\
6\ &\square(\bigcirc\pi_{R6}^{r'} \rightarrow \neg(\bigcirc\pi_{R6} \vee \bigcirc\pi_{R5} \vee \bigcirc\pi_{R1})) \\
7\ &\varphi_{e,r}^{t[opposite]} = \square(\bigcirc \pi_{R1}^{c} \rightarrow \bigcirc \pi_{R4}^{r'}) \wedge \square(\bigcirc \pi_{R2}^{c} \rightarrow \bigcirc \pi_{R5}^{r'}) \wedge \\
8\ &\square(\bigcirc \pi_{R3}^{c} \rightarrow \bigcirc \pi_{R6}^{r'}) \wedge \square(\bigcirc \pi_{R4}^{c} \rightarrow \bigcirc \pi_{R1}^{r'}) \wedge \\
9\ &\square(\bigcirc \pi_{R5}^{c} \rightarrow \bigcirc \pi_{R2}^{r'}) \wedge \square(\bigcirc \pi_{R6}^{c} \rightarrow \bigcirc \pi_{R3}^{r'})
\end{aligned}$$

4 Problem Formulation

In a specification φ_r, one can model the other robots in the shared workspace as part of the environment and encode the other robots' possible behaviors in the environment safety assumptions $\varphi_{e,r}^t$ and environment livenesses $\varphi_{e,r}^g$. Consider the following problem:

Problem 1. Given a set of robots $\mathcal{R}$ each having its own task φ_r in a shared workspace, find an approach that guarantees the robots can in fact complete their tasks when some of them $\mathcal{R}' \subseteq \mathcal{R}$ run into conflicts with each other, or notify the user that the conflicts cannot be resolved.

5 Approach - Glocal Coordination

In this section, the set of robots $\mathcal{R}$ operating in the shared workspace is separated into $Self$ and $\overline{\{Self\}}$. $Self$ is the robot under control. $\overline{\{Self\}} = \mathcal{R} \setminus \{Self\}$ is the set of other robots operating in the workspace and part of $Self$'s environment. In this work, it is assumed that each robot can access the current location and the action completion status of the other robots.

To address Problem 1, consider the robots $Self$ and $\overline{\{Self\}}$ each conducting a different task encoded as the specification φ_{Self} for $Self$ and φ_{Other} for $Other \in \overline{\{Self\}}$. From the specifications, the automatons, $\mathcal{A}_{Self}$ for $Self$ and $\mathcal{A}_{Other}$ for $Other \in \overline{\{Self\}}$, are successfully synthesized.

During execution, as described in [23], $Self$'s controller can run in parallel with a runtime monitor on the set of environment safety assumptions $\varphi_{e,Self}^t$ to detect any violations. When any violations occur, the set of violated environment safety assumptions $\varphi_{e,r}^{t[violated]}$ of robot r is identified.

If the violated assumption $\varphi_{e,Self}^{t[violated]}$ only involves the global sensors $Sen_{\mathcal{G}}$ or the robot sensors Sen_{Self}, then the Environment Characterization approach of [23] can be used to resolve the problem. In this work we assume the conflict is with the other robots in the environment. Furthermore, we assume the conflict cannot be resolved by one robot changing its task to accommodate the other robot as done in [24].

Since we assume that the robots must coordinate in order to resolve the conflict, $Self$ triggers coordination by first identifying the set of conflicting robots with $Self$, $\mathcal{R}^*$.

The approach is named the glocal coordination as it forms connections between global and local robot controllers.

5.1 Glocal Specification

To coordinate, each robot first obtains the specifications of the conflicting robots and creates a glocal specification $\varphi_{\mathcal{G}}$. The set of coordinating robots is $\mathcal{R}^{coop} = \{Self\} \cup \mathcal{R}^*$. For example, if Alice and Bob are coordinating, then $\mathcal{R}^{coop} = \{A, B\}$. The set of atomic propositions $AP_{\mathcal{G}}$ used in the Glocal Coordination is $AP_{\mathcal{G}} = \mathcal{X}_{\mathcal{G}} \cup \mathcal{Y}_{\mathcal{G}}$, where:

$$\mathcal{X}_{\mathcal{G}} = \bigcup_{r\# \in \mathcal{R}^{coop}} (\mathcal{X}_{r\#} \setminus env_{r\#}^{\mathcal{R}^*}) \qquad \mathcal{Y}_{\mathcal{G}} = \bigcup_{r\# \in \mathcal{R}^{coop}} \mathcal{Y}_{r\#}$$, with

$$env_{r\#}^{\mathcal{R}^*} = \{\pi^{r'} \mid r' \in \mathcal{R}^* \text{ and } \pi^{r'} \in \{envAct_{r\#} \cup envReg_{r\#}\}\}$$

The set of atomic propositions $AP_{\mathcal{G}}$ consists of all the system and environment propositions from the coordinating robots $\mathcal{R}^{coop}$, excluding the environment propositions that reason about the location and the action completion of the other coordinating robots ($env_{r\#}^{\mathcal{R}^*}$). These propositions are excluded as they are duplicates. For example, in Example 1, $\pi_{R2}^{A} \in envReg_B$ in Bob's environment propositions $\mathcal{X}_B$ is a duplicate of the environment proposition $\pi_{R2}^{c} \in Reg_A^c$ in Alice's environment propositions $\mathcal{X}_A$.

The glocal specification $\varphi_{\mathcal{G}}$ is as follows:

$$\varphi_{\mathcal{G}} = \varphi_{e,\mathcal{G}}^i \wedge \varphi_{e,\mathcal{G}}^t \wedge \varphi_{e,\mathcal{G}}^g \rightarrow \varphi_{s,\mathcal{G}}^i \wedge \varphi_{s,\mathcal{G}}^t \wedge \varphi_{s,\mathcal{G}}^g$$

The glocal specification $\varphi_{\mathcal{G}}$ only considers the current goals of the coordinating robots together with all the other requirements given in the local specifications φ_{Self} and φ_{Other} for $Other \in \mathcal{R}^*$.

$\varphi_{e,\mathcal{G}}^i$ and $\varphi_{s,\mathcal{G}}^i$, as shown Eq. 5 and Eq. 6 respectively, are the environment and system initial conditions based on the current state of the robots. In Eq. 5 and 6, $s_{r\#}$ is the current state of robot $r^{\#}$ and $\gamma_{r\#}^s$ is the state labeling function of robot $r^{\#}$.

$$\varphi_{e,\mathcal{G}}^i = \bigwedge_{r\# \in \mathcal{R}^{coop}} (\bigwedge_{\pi \in \gamma_{r\#}^s(s_{r\#}) \setminus \mathcal{Y}_{r\#}} \pi \wedge \bigwedge_{\pi \in \mathcal{X}_{r\#} \setminus \gamma_{r\#}^s(s_{r\#})} \neg\pi) \quad (5)$$

$$\varphi_{s,\mathcal{G}}^i = \bigwedge_{r\# \in \mathcal{R}^{coop}} (\bigwedge_{\pi \in \gamma_{r\#}^s(s_{r\#}) \setminus \mathcal{X}_{r\#}} \pi \wedge \bigwedge_{\pi \in \mathcal{Y}_{r\#} \setminus \gamma_{r\#}^s(s_{r\#})} \neg\pi) \quad (6)$$

The glocal environment safety assumptions $\varphi^t_{e,\mathcal{G}}$ contains all the environment safety assumptions $\varphi^t_{e,r^\#}$ in each specification $\varphi_{r^\#}$ of the coordinating robots $r^\# \in \mathcal{R}^{coop}$, excluding those made exclusively about the coordinating robots. The assumptions $\varphi^t_{e,\mathcal{G}}$ is an LTL formula of the form:

$$\varphi^t_{e,\mathcal{G}} = \bigwedge_{r^\# \in \mathcal{R}^{coop}} (\varphi^t_{e,r^\#} \setminus \varphi^{t[coop]}_{e,r^\#}) \tag{7}$$

where:

$$\begin{aligned}\varphi^{t[coop]}_{e,r^\#} =& \{\Box\, \psi^t_{e,r^\#} \mid \psi^t_{e,r^\#} \text{ s.t. } \forall \pi \in \psi^t_{e,r^\#}, \\ & \exists \pi^{r' \in \mathcal{R}^*} \in \{envReg_{r^\#} \cup envAct_{r^\#}\} \text{ and} \\ & \nexists \pi^{r' \in \mathcal{R} \setminus \mathcal{R}^{coop}} \in \{envReg_{r^\#} \cup envAct_{r^\#}\}\}\end{aligned} \tag{8}$$

$\varphi^{t[coop]}_{e,r^\#}$ in Eq. 7 is all assumptions made only about the conflicting robots $\mathcal{R}^*$ or together with robot $r^\#$ in robot $r^\#$'s specification (the exist clause in (8)), and are unrelated to the robots not coordinating (the does-not-exist clause in (8)). For example, when coordinating, $\varphi^{t[opposite]}_{e,r}$ (Line 7-9 in Spec. 3) is included in the set of assumptions $\varphi^{t[coop]}_{e,r^\#}$ for both robots in Example 1.

The assumptions $\varphi^{t[coop]}_{e,r^\#}$ relating the coordinating robots $\mathcal{R}^*$ and robot $r^\#$ can be removed as the glocal automaton $\mathcal{A}_\mathcal{G}$, if successfully synthesized, now controls all the coordinating robots. No assumptions about the behaviors of the coordinating robots are needed anymore.

Similarly, in the glocal environment livenesses $\varphi^g_{e,\mathcal{G}}$ (Eq. 9) the livenesses relating the coordinating robots $\mathcal{R}^*$ and robot $r^\#$ are removed for the same reason.

$$\varphi^g_{e,\mathcal{G}} = \bigwedge_{r^\# \in \mathcal{R}^{coop}} (\varphi^g_{e,r^\#} \setminus \varphi^{g[coop]}_{e,r^\#}) \tag{9}$$

where:

$$\begin{aligned}\varphi^{g[coop]}_{e,r^\#} =& \{\Box\Diamond\, \psi^g_{e,r^\#} \mid \psi^g_{e,r^\#} \text{ s.t. } \forall \pi \in \psi^g_{e,r^\#}, \\ & \exists \pi^{r' \in \mathcal{R}^*} \in \{envReg_{r^\#} \cup envAct_{r^\#}\} \text{ and} \\ & \nexists \pi^{r' \in \mathcal{R} \setminus \mathcal{R}^{coop}} \in \{envReg_{r^\#} \cup envAct_{r^\#}\}\}\end{aligned}$$

Since all the system safety properties in the local controllers $\mathcal{A}_{r^\#}$ of the coordinating robots $\mathcal{R}^{coop}$ should be passed to the glocal controller $\mathcal{A}_\mathcal{G}$, the glocal system safety guarantees $\varphi^t_{s,\mathcal{G}}$ (Eq. 10) includes all the system safety requirements $\varphi^t_{s,r^\#}$ of the coordinating robots $r^\# \in \mathcal{R}^{coop}$.

$$\varphi^t_{s,\mathcal{G}} = \bigwedge_{r^\# \in \mathcal{R}^{coop}} \varphi^t_{s,r^\#} \tag{10}$$

The glocal system goals $\varphi^g_{s,\mathcal{G}}$ is as follows:

$$\varphi^g_{s,\mathcal{G}} = \bigwedge_{r^\# \in \mathcal{R}^{coop}} \Box\Diamond(\psi^g_{s,r^\#}) \wedge \Box\Diamond(\bigwedge_{r^\# \in \mathcal{R}^{coop}} W_{r^\#}) \tag{11}$$

where $\psi^g_{s,r^\#}$ is the current goal of robot $r^\#$. $W_{r^\#}$ is an LTL formula that encodes the set of all states $S_{r^\#}$ in the controller $\mathcal{A}_{r^\#}$ for robot $r^\#$.

Since the glocal automaton $\mathcal{A}_\mathcal{G}$ is only a temporary controller, used until a conflict is resolved, the coordinating robots $\mathcal{R}^{coop}$ must be able to resume the execution of their original controllers $\mathcal{A}_{Self}$ and $\mathcal{A}_{Other}$ for $Other \in \mathcal{R}^*$ once their goals are reached. In this case, besides the current goals of the coordinating robots, another system goal is added to the glocal system goals $\varphi^g_{s,\mathcal{G}}$. This system goal $\Box\Diamond(\bigwedge_{r^\# \in \mathcal{R}^{coop}} W_{r^\#})$ includes all the states $S_{r^\#}$ in each robot controller $\mathcal{A}_{r^\#}$ for $r^\# \in \mathcal{R}^{coop}$. By including the formulas of all states as one of the system livenesses, the resulting automaton $\mathcal{A}_\mathcal{G}$, if successfully synthesized, first satisfies the robots' current goals and then prepares the robots to return to their local automatons by leading the robots into their set of all states in their local controllers.

When creating the glocal specification, the propositions are renamed to reflect the correct physical mapping. In Example 1, π^B_{R4} in Alice's specification φ_A is renamed to π^c_{B-R4} in the glocal specification. π^c_{R1} in φ_A is renamed to π^c_{A-R1}.

5.2 Synthesis and Execution

Each coordinating robot $r^\# \in \mathcal{R}^{coop}$ then synthesizes its own glocal controller $\mathcal{A}_\mathcal{G}$ with the same specification $\varphi_\mathcal{G}$. The controller $\mathcal{A}_\mathcal{G}$ is a finite state deterministic automaton, where the transition function only outputs one next state given a state and a subset of environment propositions: $\delta^s_\mathcal{G} : \mathcal{S}_\mathcal{G} \times 2^{\mathcal{X}_\mathcal{G}} \to \mathcal{S}_\mathcal{G}$.

The glocal controller $\mathcal{A}_\mathcal{G}$ is deterministic to ensure each coordinating robot is in the same state during the execution of $\mathcal{A}_\mathcal{G}$. The local controller of each robot $\mathcal{A}_r$ is nondeterministic to ensure when the robot returns to its local automaton, it can find a valid state to continue its task.

If the controller $\mathcal{A}_\mathcal{G}$ cannot be synthesized, then the tasks are aborted. However, if the controller $\mathcal{A}_\mathcal{G}$ is successfully synthesized, each robot will inform the other coordinating robots that it is ready to resume execution. Once every robot in $\mathcal{R}^{coop}$ is ready, each robot will start executing the glocal controller $\mathcal{A}_\mathcal{G}$. The other robots not coordinating $\mathcal{R} \setminus \mathcal{R}^{coop}$ stay in place and wait until all the robots in $\mathcal{R}^{coop}$ are ready again, then they continue their tasks unaffected.

During this decentralized execution of the strategy $\mathcal{A}_\mathcal{G}$, the coordinating robots continue to update all robots $\mathcal{R}$ with their current location and action completion status. The robots can update asynchronously with the completion modeling of continuous low-level controllers described in [19].

While executing the glocal controller $\mathcal{A}_\mathcal{G}$, each coordinating robot monitors both the completion of the coordinating robots' current goals $\bigwedge_{r^\# \in \mathcal{R}^{coop}} \Box\Diamond(\psi^g_{s,r^\#})$ and the local states goal $\Box\Diamond(\bigwedge_{r^\# \in \mathcal{R}^{coop}} W_{r^\#})$.

Each coordinating robot also checks if (i) new coordination request is received from any non-coordinating robot $\mathcal{R} \setminus \mathcal{R}^{coop}$ or (ii) any assumption in $\varphi^t_{e,\mathcal{G}}$ is violated. In these cases, the robot follows the same procedures as before and creates a new glocal controller with more robots included. Environment characterization of [23] can be carried out if the violation of $\varphi^t_{e,\mathcal{G}}$ only involves the global sensors $Sen_\mathcal{G}$ or the robot sensors $Sen_{r^\#}$ of coordinating robots $r^\# \in \mathcal{R}^{coop}$. If one robot detects another coordination request but the other coordinating robots do not, then the others are notified and all of them stop and resynthesize. The upper bound is a global coordination involving all the robots in $\mathcal{R}$.

The execution of the glocal automaton $\mathcal{A}_\mathcal{G}$ continues until the system goals in $\varphi_\mathcal{G}$ are accomplished at least once and the robots $r^\# \in \mathcal{R}^{coop}$ are within the set of all states in their local controllers $\mathcal{A}_{r^\#}$. Afterwards, the coordinating robots $\mathcal{R}^{coop}$ resume their local automaton execution.

To resume the execution of $\mathcal{A}_{Self}$ and $\mathcal{A}_{Other}$ for $Other \in \mathcal{R}^*$, each robot searches for an initial state in their local automaton based on its exit state from the glocal controller $\mathcal{A}_\mathcal{G}$ and heads to its next system goal. Since the sets of all states of the robots are included as a system goal when synthesizing the glocal automaton, the robots should find a valid state in their local automatons. The execution continues and coordination can be triggered again when the environment safety

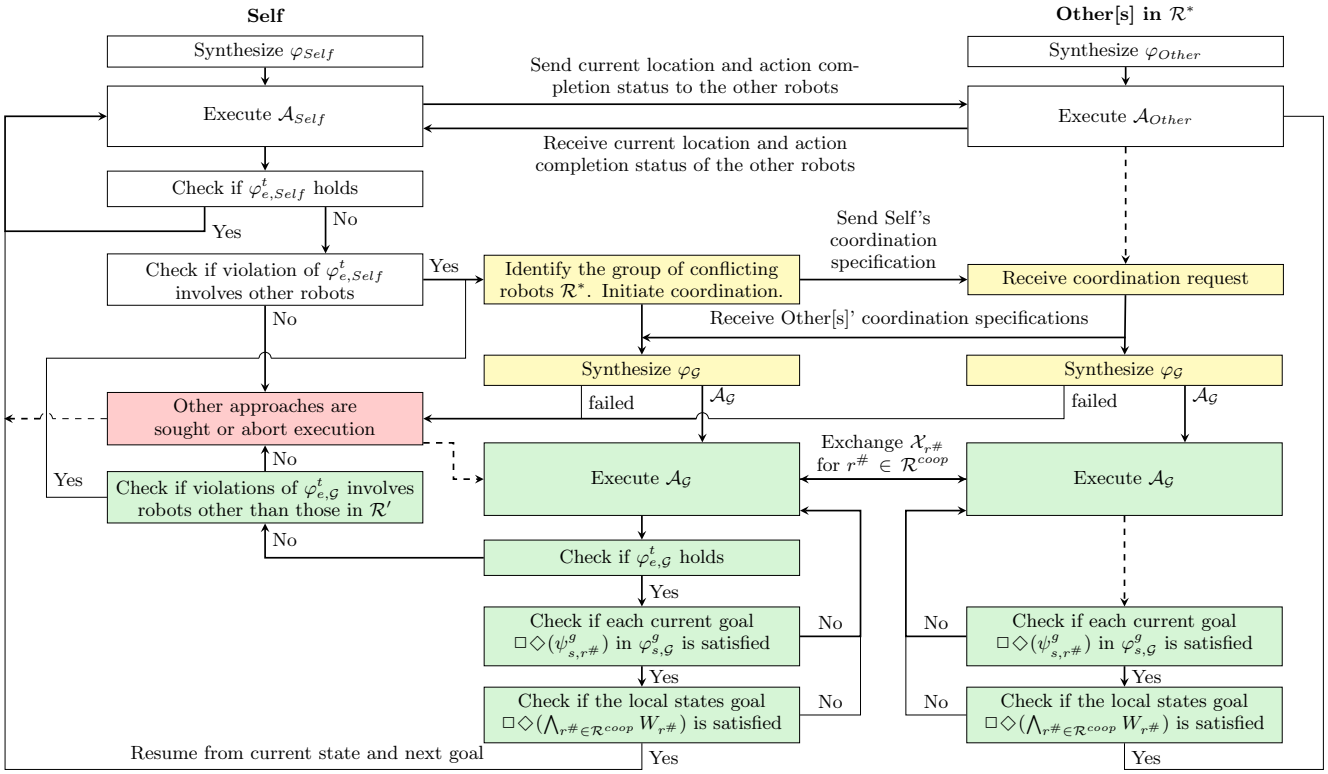


Figure 2: Description of the Glocal Coordination approach. The yellow boxes describe the steps taken when preparing for $\mathcal{A}_{\mathcal{G}}$ while the green boxes refer to the glocal controller execution.

assumptions of at least one of the robots is violated.

In φ_{Self} and φ_{Other} for $Other \in \mathcal{R}^*$, there might be multiple system goals and these goals might be satisfied without running into any violations of the environment safety requirements. The approach here only considers the current goal pursued by each robot when conducting the glocal synthesis. By excluding the other goals, this reduces the synthesis time and the size of the glocal automaton. Fig. 2 gives an overview of the approach described here.

5.3 Guarantees

THEOREM 1. *If the glocal specification $\varphi_{\mathcal{G}}$ is realizable, then:*

1. *the local system guarantees $\varphi^t_{s,r\#}$ of the coordination robots $\mathcal{R}^{coop}$ are satisfied in the glocal automaton $\mathcal{A}_{\mathcal{G}}$, provided that the glocal environment specification $\varphi_{e,\mathcal{G}}$ holds in every trace σ in $\mathcal{A}_{\mathcal{G}}$;*
2. *the glocal system goals $\varphi^g_{s,\mathcal{G}}$ is satisfied in the glocal automaton $\mathcal{A}_{\mathcal{G}}$, provided that the glocal environment specification $\varphi_{e,\mathcal{G}}$ holds in every trace σ in $\mathcal{A}_{\mathcal{G}}$;*
3. *the robots $r^{\#} \in \mathcal{R}^{coop}$ can find initial states in their local automatons $\mathcal{A}_{r\#}$ and resume execution when the glocal system goal $\square\diamond(\bigwedge_{r\#\in\mathcal{R}^{coop}} W_{r\#})$ is satisfied.*

PROOF. By definition, the glocal specification $\varphi_{\mathcal{G}}$ is realizable if an automaton $\mathcal{A}_{\mathcal{G}}$ satisfying the specification $\varphi_{\mathcal{G}}$ can be generated.

Claim 1. Follows from the definition of logical implication, the glocal specification formula $\varphi_{\mathcal{G}} = \varphi_{e,\mathcal{G}} \rightarrow \varphi_{s,\mathcal{G}}$ is satisfied by either falsifying $\varphi_{e,\mathcal{G}}$ or satisfying both $\varphi_{e,\mathcal{G}}$ and $\varphi_{s,\mathcal{G}}$.

The work of Ehlers et al. showed that we can exclude the states that falsify $\varphi_{e,\mathcal{G}}$ when extracting the automaton $\mathcal{A}_{\mathcal{G}}$ (The reader is referred to [7] for more details.). If an automaton $\mathcal{A}_{\mathcal{G}}$ is successfully synthesized, then $\varphi_{e,\mathcal{G}}$ and $\varphi_{s,\mathcal{G}}$ are both satisfied. It follows that $\varphi^t_{s,\mathcal{G}}$ is satisfied.

Since $\varphi^t_{s,\mathcal{G}}$ includes all the local system safety guarantees $\varphi^t_{s,r\#}$ of the coordination robots $\mathcal{R}^{coop}$ (See Eq. 10), these guarantees are satisfied in the glocal automaton $\mathcal{A}_{\mathcal{G}}$.

Claim 2. Following the same approach in the first claim, we can exclude the states that falsify $\varphi_{e,\mathcal{G}}$ when extracting the automaton $\mathcal{A}_{\mathcal{G}}$. If an automaton $\mathcal{A}_{\mathcal{G}}$ is successfully synthesized, then $\varphi_{e,\mathcal{G}}$ and $\varphi_{s,\mathcal{G}}$ are both satisfied. It follows that the glocal system goals $\varphi^g_{s,\mathcal{G}}$ is satisfied.

Claim 3. Following from the second claim, the local states formulas $W_{r\#}$ for the robots $r^{\#} \in \mathcal{R}^{coop}$ are satisfied when the system goal $\square\diamond(\bigwedge_{r\#\in\mathcal{R}^{coop}} W_{r\#})$ in $\varphi^g_{s,\mathcal{G}}$ is satisfied.

Each local states formula $W_{r\#}$ encodes the possible states in each robot's local automaton $\mathcal{A}_{r\#}$. When the system goal $\square\diamond(\bigwedge_{r\#\in\mathcal{R}^{coop}} W_{r\#})$ in $\varphi^g_{s,\mathcal{G}}$ is satisfied, the robots are within the set of all states in their local automatons $\mathcal{A}^{\#}_r$.

At this position in the glocal automaton $\mathcal{A}_{\mathcal{G}}$, the robots can find an initial state in their local automatons $\mathcal{A}_{r\#}$ and resume their execution.

□

6 Examples

6.1 Patrol and Request Fulfillment

Consider the workspace shown in Fig. 3a. Here Alice (A) and Bob (B) are conducting patrol tasks while Charlie (C) is conducting a request fulfillment task. The partial specifications of the three robots are shown in Spec. 4, 5 and 6.

Example 2. [1]Alice starts in $L2$, Bob starts in $R2$ and Charlie starts in $M1$ (Line 1, 2 of Spec. 4, 5 and 6). Alice's

Specification 4 Alice's specification φ_A for Example 2

$$
\begin{aligned}
1\ \varphi^{i}_{s,A} &= \pi_{L2} \\
2\ \varphi^{i}_{e,A} &= \pi^{c}_{L2} \wedge \pi^{B}_{R2} \wedge \pi^{C}_{M1} \\
3\ \varphi^{g}_{s,A} &= \square\lozenge(\bigcirc \pi_{L1} \wedge \bigcirc \pi^{c}_{L1}) \wedge \\
4 &\quad\ \square\lozenge(\bigcirc \pi_{L3} \wedge \bigcirc \pi^{c}_{L3}) \\
5\ \varphi^{t}_{e,A} &= \square\neg(\bigcirc \pi^{B}_{L1} \vee \bigcirc \pi^{B}_{L2} \vee \bigcirc \pi^{B}_{L3}) \wedge \\
6 &\quad\ \square\neg(\bigcirc \pi^{C}_{L1} \vee \bigcirc \pi^{C}_{L2} \vee \bigcirc \pi^{C}_{L3})
\end{aligned}
$$

Specification 5 Bob's specification φ_B for Example 2

$$
\begin{aligned}
1\ \varphi^{i}_{s,B} &= \pi_{R2} \\
2\ \varphi^{i}_{e,B} &= \pi^{c}_{R2} \wedge \pi^{A}_{L2} \wedge \pi^{C}_{M1} \\
3\ \varphi^{g}_{s,B} &= \square\lozenge(\bigcirc \pi_{R1} \wedge \bigcirc \pi^{c}_{R1}) \wedge \\
4 &\quad\ \square\lozenge(\bigcirc \pi_{R3} \wedge \bigcirc \pi^{c}_{R3}) \\
5\ \varphi^{t}_{e,B} &= \square\neg(\bigcirc \pi^{B}_{R1} \vee \bigcirc \pi^{B}_{R2} \vee \bigcirc \pi^{B}_{R3}) \wedge \\
6 &\quad\ \square\neg(\bigcirc \pi^{C}_{R1} \vee \bigcirc \pi^{C}_{R2} \vee \bigcirc \pi^{C}_{R3})
\end{aligned}
$$

Specification 6 Charlie's specification φ_C for Example 2

$$
\begin{aligned}
1\ \varphi^{i}_{s,C} &= \pi_{M1} \\
2\ \varphi^{i}_{e,C} &= \pi^{c}_{M1} \wedge \pi^{A}_{L2} \wedge \pi^{B}_{R2} \\
3\ \varphi^{g}_{s,C} &= \square\lozenge(\pi_{requestL3} \rightarrow (\bigcirc \pi^{c}_{L3} \wedge \bigcirc \pi_{L3})) \wedge \\
4 &\quad\ \square\lozenge(\pi_{requestR1} \rightarrow (\bigcirc \pi^{c}_{R1} \wedge \bigcirc \pi_{R1})) \wedge \\
5 &\quad\ \square\lozenge(\neg(\pi_{requestL3} \vee \pi_{requestR1}) \rightarrow (\bigcirc \pi^{c}_{M1} \wedge \bigcirc \pi_{M1})) \\
6\ \varphi^{t}_{e,C} &= \square(\bigcirc \pi^{A}_{L1} \vee \bigcirc \pi^{A}_{L2} \vee \bigcirc \pi^{A}_{L3}) \wedge \\
7 &\quad\ \square(\bigcirc \pi^{B}_{R1} \vee \bigcirc \pi^{B}_{R2} \vee \bigcirc \pi^{B}_{R3}) \wedge \\
8 &\quad\ \square\neg(\bigcirc \pi_{requestR1} \wedge \bigcirc \pi_{requestL3}) \wedge \\
9 &\quad\ \square((\bigcirc \pi^{c}_{ML} \wedge \bigcirc \pi_{requestL3}) \rightarrow \neg \bigcirc \pi^{A}_{L2}) \wedge \\
10 &\quad\ \square(\bigcirc \pi^{c}_{L2} \rightarrow \neg(\bigcirc \pi^{A}_{L2} \vee \bigcirc \pi^{A}_{L3})) \wedge \\
11 &\quad\ \square(\bigcirc \pi^{c}_{L3} \rightarrow \neg(\bigcirc \pi^{A}_{L2} \vee \bigcirc \pi^{A}_{L3})) \wedge \\
12 &\quad\ \square((\bigcirc \pi^{c}_{MR} \wedge \bigcirc \pi_{requestR1}) \rightarrow \neg \bigcirc \pi^{B}_{R2}) \wedge \\
13 &\quad\ \square(\bigcirc \pi^{c}_{R2} \rightarrow \neg(\bigcirc \pi^{B}_{R2} \vee \bigcirc \pi^{B}_{R1})) \wedge \\
14 &\quad\ \square(\bigcirc \pi^{c}_{R1} \rightarrow \neg(\bigcirc \pi^{B}_{R2} \vee \bigcirc \pi^{B}_{R1}))
\end{aligned}
$$

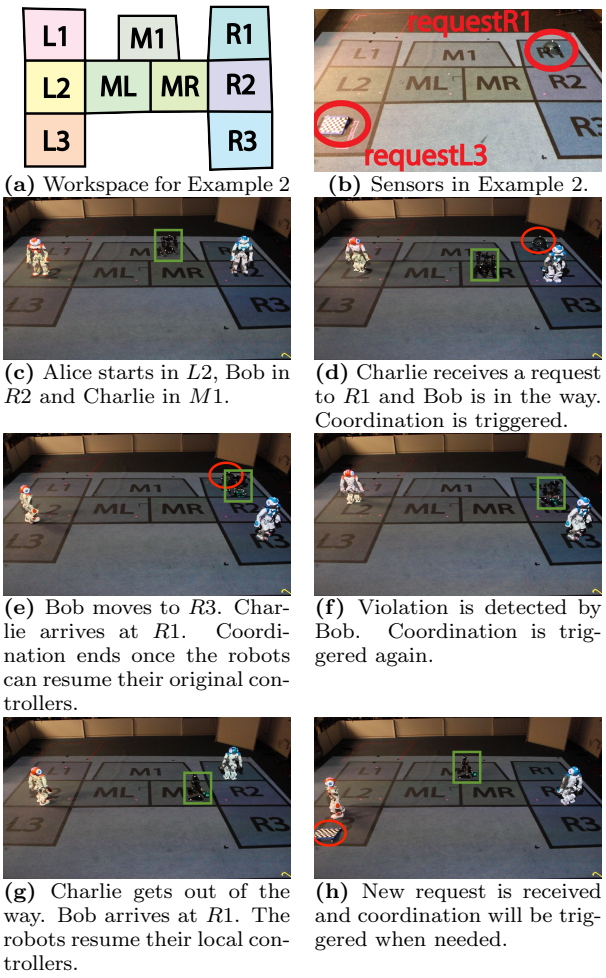
task is to patrol between $L1$ and $L3$ (Line 3, 4 of Spec. 4), while Bob's task is to patrol between $R1$ and $R3$ (Line 3, 4 of Spec. 5). To ensure they can complete their task, Alice assumes that the other two robots will never go to $L1$, $L2$ or $L3$ (Line 5, 6 of Spec. 4); Bob assumes that the other two robots will never go to $R1$, $R2$ or $R3$ (Line 5, 6 of Spec. 5).

Charlie's task is to go to $L3$ if $requestL3$ is received and go to $R1$ if $requestR1$ is received. Otherwise, Charlie visits $M1$ (Line 3-5 of Spec. 6). Charlie cannot sense both $requestL3$ and $requestR1$ at the same time (Line 8 in Spec. 6). Charlie assumes that the other two robots will never go to $M1$, ML or MR (Line 6, 7 of Spec. 6). On his way to his goals, the other robots will be out of his way (Line 9-14 of Spec. 6). The sensors $requestL3$ and $requestR1$ are shown in Fig. 3b.

In Spec. 4- 6, we omit the topology constraints of the neighboring robots (See $\varphi^{t[adj]}_{e,r}$ in Eq. 4), the assumption that the environment robots can only be in one region at a time (See $\varphi^{t[region]}_{e,r}$ in Eq. 3), and the mutual exclusion constraints in each of the specifications (See $\varphi^{t[mutual]}_{s,r}$ in Eq. 2). As each specification is realizable, we can synthesize controllers one for each robot.

Initially, Charlie stays in $M1$ as he detects no requests, while Alice patrols between $L1$ and $L3$, and Bob patrols between $R1$ and $R3$ (Fig. 3c).

Once Charlie receives a request to $R1$ ($\pi_{requestR1}$ is True), he heads to MR. There, Bob is in $R2$ and Charlie detects a violation of Line 9 in Spec. 6: if you are sensing $requestR1$ and you are in MR then Bob should not be in $R2$. In Charlie's specification, it is assumed that the other robots will not be in his way when he is heading to his goals (Fig. 3d).



(a) Workspace for Example 2 **(b)** Sensors in Example 2.

(c) Alice starts in $L2$, Bob in $R2$ and Charlie in $M1$.

(d) Charlie receives a request to $R1$ and Bob is in the way. Coordination is triggered.

(e) Bob moves to $R3$. Charlie arrives at $R1$. Coordination ends once the robots can resume their original controllers.

(f) Violation is detected by Bob. Coordination is triggered again.

(g) Charlie gets out of the way. Bob arrives at $R1$. The robots resume their local controllers.

(h) New request is received and coordination will be triggered when needed.

Figure 3: Alice, the orange Aldebaran Nao; Bob, the blue Aldebaran Nao; and Charlie, the black Johnny5 highlighted with a green box, performing their tasks as described in Example 2.

Detecting the violation, Charlie then asks Bob to coordinate and achieve their goals together. Charlie sends Bob his specification, his current goal and his local states formula. Receiving Charlie's request, Bob sends Charlie his specification, his current goal and his local states formula as well.

Bob's current goal is to visit $R3$ (Line 4 in Spec. 5), while Charlie's goal is to go to $R1$ if $requestR1$ is true (Line 4 in Spec. 6). Both robots check if the glocal specification $\varphi_{\mathcal{G}}$ constructed by joining φ_B and φ_C is realizable and each synthesizes a glocal controller $\mathcal{A}_{\mathcal{G}}$.

The joint specification of Bob and Charlie $\varphi_{\mathcal{G}}$ is realizable so the two robots continue their tasks with the new controllers $\mathcal{A}_{\mathcal{G}}$. Alice continues her task unaffected. With the controller $\mathcal{A}_{\mathcal{G}}$, first Bob moves from $R2$ to $R3$. Now Bob is not in the way, Charlie moves to $R1$ through $R2$ (Fig. 3e).

After the two robots have accomplished their current goals once, they head to states where the local states goal is satisfied. In this case, the state with π^{c}_{B-R3}, π_{B-R3}, π^{c}_{C-R2}, π_{C-R1} and π^{A}_{L2} being true satisfies the local states goal.

Once they are within one of the states in the local robot controller $\mathcal{A}_{r^{\#}}$ for $r^{\#} \in \mathcal{R}^{coop}$, Bob and Charlie return to their local automatons. Alice pauses her execution until all the robots' controllers are ready for execution again.

Bob and Charlie move on to the next goal. Bob's next goal is to visit $R1$ (Line 3 in Spec 5). Charlie's next goal is to go to $M1$ if no requests are received (Line 5 in Spec 6).

When the execution resumes, $reqestR1$ is true so Charlie's goal to go to $M1$ if no requests are received is satisfied (Line 5 in Spec. 6), through falsifying the first clause in the implication formula. Charlie's goal to go to $L3$ when $requestL3$ is true is also satisfied (Line 3 in Spec. 6) .

$requestR1$ is now off. The local automatons execution is asynchronous and Bob detects a violation: Charlie is not expected to be in $R1$, $R2$ or $R3$ (Line 6 in Spec. 5, Fig. 3f). Coordination is triggered and a new glocal controller $\mathcal{A}_\mathcal{G}$ is successfully synthesized. In the video, Charlie's current goal is to go to $R1$ if $requestR1$ is true again (Line 4 in Spec. 6) since the other two goals were satisfied.

With the controller $\mathcal{A}_\mathcal{G}$, Charlie first moves from $R2$ to MR. After Charlie is out of the way, Bob can go from $R3$ to $R1$ (Fig. 3g). Once the robots reach their goals, they satisfy the local states goal with the state of π^c_{B-R1}, π_{B-R1}, π^c_{C-MR}, π_{C-R2} and π^A_{L2} being true.

Bob and Charlie then return to their local automatons. Alice and Bob continue their patrol task. Charlie heads to $M1$ and stays there until a new request is received (Fig. 3h).

6.2 Manipulation

To illustrate the generality of this approach, consider a different domain. Two robots Alice and Bob are required to perform a manipulation task, as shown in Spec. 7 and Spec. 8. The workspace consists of two regions $R1$ and $R2$ right next to each other.

Example 3. [1]Alice and Bob start in $R1$ and $R2$ respectively (Line 1-2 in Spec. 7, 8). Both of them stay in place at all times (Line 5, 7 in Spec. 7, 8). Alice conducts a block painting task: when she is asked to start the task, she picks up the block (Line 8 in Spec. 7), paints it (Line 9 in Spec. 7) and remembers that the task is done until she is told that she can stop (Line 10-13 in Spec. 7). Bob conducts a similar task which he stamps the block instead when he is asked to start his task (Line 8-13 in Spec. 7). It is assumed that when one robot tries to pick up the block, the other robot has not picked up the block yet (Line 6 in Spec. 7 and 8).

Specification 7 Alice's specification φ_A for Example 3

$$
\begin{aligned}
1\ &\varphi^i_{s,A} = \pi_{R1}\\
2\ &\varphi^i_{e,A} = \pi^c_{R1} \wedge \pi^B_{R2}\\
3\ &\varphi^g_{s,A} = \Box\Diamond(\pi_{startPainting} \rightarrow \bigcirc \pi_{blockPainted})\\
4\ &\varphi^g_{e,A} = \Box\Diamond(\bigcirc \pi_{startPainting} \wedge \bigcirc \pi_{blockPainted})\\
5\ &\varphi^t_{e,A} = \Box(\bigcirc \pi^B_{R2}) \wedge\\
6\ &\qquad \Box(\pi_{takeBlock} \rightarrow \neg \bigcirc \pi^B_{takeBlock})\\
7\ &\varphi^t_{e,A} = \Box(\pi_{R1} \rightarrow \bigcirc \pi_{R1}) \wedge\\
8\ &\qquad \Box((\neg \bigcirc \pi^B_{takeBlock} \wedge \neg \bigcirc \pi_{blockPainted} \wedge\\
&\qquad\qquad \bigcirc \pi_{startPainting}) \leftrightarrow \bigcirc \pi_{takeBlock}) \wedge\\
9\ &\qquad \Box(\bigcirc \pi^c_{takeBlock} \leftrightarrow \bigcirc \pi_{paint}) \wedge\\
10\ &\qquad \Box(((\pi^c_{paint} \wedge \pi^c_{takeBlock}) \wedge \bigcirc \pi_{startPainting})\\
&\qquad\qquad \rightarrow \bigcirc \pi_{blockPainted}) \wedge\\
11\ &\qquad \Box(\neg \bigcirc \pi_{startPainting} \rightarrow \neg \bigcirc \pi_{blockPainted}) \wedge\\
12\ &\qquad \Box((\pi_{blockPainted} \wedge \bigcirc \pi_{startPainting}) \rightarrow\\
&\qquad\qquad \bigcirc \pi_{blockPainted}) \wedge\\
13\ &\qquad \Box((\neg\pi_{blockPainted} \wedge \neg(\pi^c_{paint} \wedge \pi^c_{takeBlock})) \rightarrow\\
&\qquad\qquad \neg \bigcirc \pi_{blockPainted})
\end{aligned}
$$

In Spec. 7, for the system propositions, $\{\pi_{takeBlock}, \pi_{paint}\} \subseteq Act_A$, $\pi_{R1} \in Reg_A$ and $\pi_{blockPainted} \in Mem_A$; for the environment propositions, $\pi^B_{R2} \in envReg_A$, $\pi^B_{takeBlock} \in envAct_A$, $\pi^c_{R1} \in Reg^c_A$, $\{\pi^c_{takeBlock}, \pi^c_{paint}\} \subseteq Act^c_A$ and $\pi_{startPainting} \in Sen_A$. In Spec. 8, for the system propositions, $\{\pi_{takeBlock}, \pi_{stamp}\} \subseteq Act_B$, $\pi_{blockStamped} \in Mem_B$ and $\pi_{R2} \in Reg_B$; for the environment propositions, $\pi^A_{R1} \in envReg_B$, $\pi^A_{takeBlock} \in envAct_B$, $\pi^c_{R2} \in Reg^c_B$, $\{\pi^c_{takeBlock}, \pi^c_{stamp}\} \subseteq Act^c_B$ and $\pi_{startStamping} \in Sen_B$.

Specification 8 Bob's specification φ_B for Example 3

$$
\begin{aligned}
1\ &\varphi^i_{s,B} = \pi_{R2}\\
2\ &\varphi^i_{e,B} = \pi^c_{R2} \wedge \pi^A_{R1}\\
3\ &\varphi^g_{s,B} = \Box\Diamond(\pi_{startStamping} \rightarrow \bigcirc \pi_{blockStamped})\\
4\ &\varphi^g_{e,B} = \Box\Diamond(\bigcirc \pi_{startStamping} \wedge \bigcirc \pi_{blockStamped})\\
5\ &\varphi^t_{e,B} = \Box(\bigcirc \pi^A_{R1}) \wedge\\
6\ &\qquad \Box(\pi_{takeBlock} \rightarrow \neg \bigcirc \pi^A_{takeBlock})\\
7\ &\varphi^t_{e,B} = \Box(\pi_{R2} \rightarrow \bigcirc \pi_{R2}) \wedge\\
8\ &\qquad \Box((\neg \bigcirc \pi^A_{takeBlock} \wedge \neg \bigcirc \pi_{blockStamped} \wedge\\
&\qquad\qquad \bigcirc \pi_{startStamping}) \leftrightarrow \bigcirc \pi_{takeBlock}) \wedge\\
9\ &\qquad \Box(\bigcirc \pi^c_{takeBlock} \leftrightarrow \bigcirc \pi_{stamp}) \wedge\\
10\ &\qquad \Box(((\pi^c_{stamp} \wedge \pi^c_{takeBlock}) \wedge \bigcirc \pi_{startStamping})\\
&\qquad\qquad \rightarrow \bigcirc \pi_{blockStamped}) \wedge\\
11\ &\qquad \Box(\neg \bigcirc \pi_{startStamping} \rightarrow \neg \bigcirc \pi_{blockStamped}) \wedge\\
12\ &\qquad \Box((\pi_{blockStamped} \wedge \bigcirc \pi_{startStamping}) \rightarrow\\
&\qquad\qquad \bigcirc \pi_{blockStamped}) \wedge\\
13\ &\qquad \Box((\neg\pi_{blockStamped} \wedge \neg(\pi^c_{stamp} \wedge \pi^c_{takeBlock})) \rightarrow\\
&\qquad\qquad \neg \bigcirc \pi_{blockStamped})
\end{aligned}
$$

Two controllers, one for each robot, are successfully generated. When started, Alice is asked to paint the block while Bob is asked to stamp it. Both Alice and Bob initiates their *takeBlock* actions; Alice completes her action before Bob.

In this case, Bob detects the violation 'If you are trying to take the block, then Alice has not picked up the block' (Line 6 in Spec. 8). Coordination is triggered.

Once coordination started, Bob stops his *takeBlock* action and waits until Alice has finished painting her block. Bob then picks up the block and stamps it. Once their goals are achieved and the two robots can return to their local controllers, coordination ends.

The two robots can continue to conduct their task separately unless violations are detected again.

7 Conclusions and Future Work

This work presents an approach to coordinate different subgroups of robots operating in a shared workspace when conflicts occur. The approach creates glocal automatons such that the robots in conflict can coordinate until all of them satisfy their current system goals at least once. The approach combines the advantage of centralized control, which allows coordination of robots, and that of decentralized robot control, which is more computationally efficient.

The synthesis of correct-by-construction controllers is computationally expensive and the extension to multi-robot scenarios can make the problem intractable. Our approach is novel in that it takes advantage of both centralized and decentralized robot control and leverages the provably-correct aspect of these controllers, while at the same time minimizing the computation hurdle.

We plan to explore other effective solutions to the robot coordination problem with high-level control in the future, mainly to reduce the global information needed. We also want to investigate the possibility to finish a portion of the task if the full task cannot be completed.

REFERENCES

[1] N. Ayanian and V. Kumar. Decentralized Feedback Controllers for Multiagent Teams in Environments With Obstacles. *IEEE Transactions on Robotics*, 26(5):878–887, 2010.

[2] N. Ayanian, A. Spielberg, M. Arbesfeld, J. Strauss, and D. Rus. Controlling a team of robots with a single input. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pages 1755–1762, 2014.

[3] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas. Symbolic planning and control of robot motion [Grand Challenges of Robotics]. *IEEE Robot. Automat. Mag.*, 14(1):61–70, 2007.

[4] A. Bhatia, L. E. Kavraki, and M. Y. Vardi. Sampling-based motion planning with temporal goals. In *ICRA*, pages 2689–2696, 2010.

[5] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of Reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3):911–938, 2012.

[6] S. S. da Costa Botelho and R. Alami. M+: A Scheme for Multi-Robot Cooperation Through Negotiated Task Allocation and Achievement. In *1999 IEEE International Conference on Robotics and Automation, Marriott Hotel, Renaissance Center, Detroit, Michigan, May 10-15, 1999, Proceedings*, pages 1234–1239, 1999.

[7] R. Ehlers, R. Könighofer, and R. Bloem. Synthesizing cooperative reactive mission plans. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[8] B. P. Gerkey and M. J. Mataric. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *I. J. Robotic Res.*, 23(9):939–954, 2004.

[9] N. Kalra, D. Ferguson, and A. Stentz. A Generalized Framework for Solving Tightly-coupled Multirobot Planning Problems. In *2007 IEEE International Conference on Robotics and Automation, ICRA 2007, 10-14 April 2007, Roma, Italy*, pages 3359–3364, 2007.

[10] M. Kloetzer and C. Belta. Temporal Logic Planning and Control of Robotic Swarms by Hierarchical Abstractions. *IEEE Transactions on Robotics*, 23(2):320–330, 2007.

[11] M. Kloetzer and C. Belta. Distributed implementations of global temporal logic motion specifications. In *ICRA*, pages 393–398, 2008.

[12] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-Logic-Based Reactive Mission and Motion Planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

[13] L. Liu, N. Michael, and D. A. Shell. Fully Decentralized Task Swaps with Optimized Local Searching. In *Robotics: Science and Systems X, University of California, Berkeley, USA, July 12-16, 2014*, 2014.

[14] S. C. Livingston and P. Prabhakar. Decoupled formal synthesis for almost separable systems with temporal logic specifications. In *Distributed Autonomous Robotic Systems DARS Daejeon, Korea, 2-5 Nov,2014*, 2014.

[15] S. G. Loizou and K. J. Kyriakopoulos. Automatic synthesis of multiagent motion tasks based on ltl specifications. In *CDC*, pages 153–158, 2004.

[16] C. Nam and D. A. Shell. When to do your own thing: Analysis of cost uncertainties in multi-robot task allocation at run-time. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 1249–1254, 2015.

[17] V. Raman. Reactive switching protocols for multi-robot high-level tasks. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, pages 336–341, 2014.

[18] V. Raman and H. Kress-Gazit. Synthesis for Multi-Robot Controllers with Interleaved Motion. In *ICRA*, pages 4316–4321, 2014.

[19] V. Raman, N. Piterman, and H. Kress-Gazit. Provably correct continuous control for high-level robot behaviors with actions of arbitrary execution durations. In *ICRA*, pages 4075–4081, 2013.

[20] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia. Automated composition of motion primitives for multi-robot systems from safe LTL specifications. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, pages 1525–1532, 2014.

[21] P. M. Shiroma and M. F. M. Campos. CoMutaR: A framework for multi-robot coordination and task allocation. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009, St. Louis, MO, USA*, pages 4817–4824, 2009.

[22] L. Vig and J. A. Adams. Multi-robot coalition formation. *IEEE Transactions on Robotics*, 22(4):637–649, 2006.

[23] K. W. Wong, R. Ehlers, and H. Kress-Gazit. Correct High-level Robot Behavior in Environments with Unexpected Events. In *RSS*, 2014.

[24] K. W. Wong and H. Kress-Gazit. Let's talk: Autonomous conflict resolution for robots carrying out individual high-level tasks in a shared workspace. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 339–345, 2015.

[25] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon control for temporal logic specifications. In *HSCC*, pages 101–110, 2010.

[26] Y. Zhang and L. E. Parker. IQ-ASyMTRe: Forming Executable Coalitions for Tightly Coupled Multirobot Tasks. *IEEE Transactions on Robotics*, 29(2):400–416, 2013.