Coupled Multi-Robot Systems under Linear Temporal Logic and Signal Temporal Logic Tasks

Lars Lindemann, *Student Member, IEEE*, Jakub Nowak, Lukas Schönbächler, Meng Guo, *Member, IEEE*, Jana Tumova, *Member, IEEE*, and Dimos V. Dimarogonas, *Senior Member, IEEE*

***Abstract*—This paper presents the implementation and experimental results of two frameworks for multi-agent systems under temporal logic tasks, which we have recently proposed. Each agent is subject to either a local linear temporal logic or a local signal temporal logic task where each task may further be coupled, i.e., satisfaction of a task may depend on more than one agent. The agents are represented by mobile robots with different sensing and actuation capabilities. We propose to combine the two aforementioned frameworks to use the strengths of both linear temporal logic and signal temporal logic. For the implementation, we take into account practical issues such as collision avoidance and, in particular for the signal temporal logic framework, input saturations, the digital implementation of continuous-time feedback control laws, and a controllability assumption that was made in the original work. The experimental results contain three scenarios that show a wide variety of tasks.**

***Index Terms*—Autonomous mobile robots; decentralized robotic networks; formal methods-based control synthesis; linear temporal logic (LTL); signal temporal logic (STL).**

I. Introduction

A multi-agent system is a collection of independent agents with the goal to achieve global or local (individual) tasks. Collaborative control deals with achieving global tasks such as consensus [1], formation control [2], connectivity maintenance [3], and collision avoidance [4], see also [5] for an overview. To impose more complex tasks such as recurrence, request-response, and time-constrained tasks, ideas from formal verification [6] have been used where the tasks are written as temporal logic formulas. Control methods for systems under temporal logic tasks can be seen as multi-purpose tools and are hence of high interest for practical applications, i.e., tasks can easily be changed while automatically obtaining correct-by-design controllers. Linear temporal logic (LTL) is a proposition-based logic used for single-agent systems [7]–[12] as well as for multi-agent systems [13]–[18] where, especially for multi-agent systems, the computational complexity is known to be high. LTL tasks can include combinations of surveillance ("periodically visit regions A, B, and C"), safety ("always avoid region D"), and many others. Signal temporal logic (STL) is a predicate-based logic interpreted over continuous-time signals [19]. STL entails space robustness [20], a form of the robust semantics [21], stating how robustly a signal satisfies a temporal logic formula. Control synthesis under STL tasks has been considered for single-agent systems [22]–[30] and for multi-agent systems [31], [32]. A common trade-off is to find a restricted, yet expressive STL fragment that allows for computationally-efficient control as in [25], [31]. STL tasks can again include combinations of surveillance ("visit regions A, B, and C every $10-60$ sec while agents form a triangular formation"), safety ("always between $5-25$ sec stay at least 1 m away from D"), and many others. Compared with LTL, STL now allows to impose desired quantitative temporal and spatial properties on the system. Multi-agent systems under temporal logic tasks are separated into two classes: top-down and bottom-up. In the former, a global task is assigned to the team of agents, while in the latter each agent is subject to a local (individual) task regardless of what other agents are assigned. These local tasks may be coupled, i.e., the satisfaction of a local task may depend on the behavior of other agents. Top-down approaches usually resort to decomposing the global task into local ones. We argue hence that a bottom-up approach is more general and considered here.

Fig. 1: TurtleBots (left) and Nexus 4WD Mecanum Robotic Cars (right) are used in the experiments in Section IV.

We consider a heterogenous multi-agent system consisting of mobile robots as in Fig. 1. Each robot is subject to either a local LTL or a local STL task. Our main contribution is the implementation and experimental results of the frameworks [14] and [31] to demonstrate the advantages of the computationally-efficient and robust methods presented therein. To account for the computational burdens of the LTL

L. Lindemann (llindem@kth.se), J. Nowak (jakubn@kth.se), and D. V. Dimarogonas (dimos@kth.se) are currently with the Division of Decision and Control Systems, KTH Royal Institute of Technology, Stockholm, Sweden.

L. Schönbächler (luki.schoenb@hotmail.de) is currently a research engineer at Bucher Hydraulics, Neuheim, Switzerland.

M. Guo (Meng.Guo2@de.bosch.com) is currently a research scientist at the Bosch Center for Artificial Intelligence, Renningen, Germany.

J. Tumova (tumova@kth.se) is currently with the Division of Robotics, Perception, and Learning, KTH Royal Institute of Technology, Stockholm, Sweden.

This work was supported in part by the Swedish Research Council (VR), the European Research Council (ERC), the Swedish Foundation for Strategic Research (SSF), the EU H2020 Co4Robots project, and the Knut and Alice Wallenberg Foundation (KAW).

plan synthesis, poses of interest are considered in [14] for the abstraction of the workspace, while the approach for STL control synthesis in [31] is inherently computationally-efficient due to the use of time-varying feedback control laws. We also aim at demonstrating the practical advantages of using both LTL and STL at the same time. Note that STL is a more complex task specification language than LTL. In fact, STL allows to impose timed tasks such as strict deadlines while admitting predicates as opposed to simple propositions in LTL. For multi-agent systems this implies that predicates can be used to couple agents with each other in a straightforward manner. Control approaches for LTL are mature and allow to use the full LTL fragment while STL control methods are still being investigated, currently only with results for fragments of STL [25], [31]. Existing methods for the full STL fragment [22], [28] discretize time (hence are, in some sense, equivalent to LTL) and are computationally demanding. We propose to use STL when timed tasks or tasks involving the coupling of agents are of interest while, otherwise, methods for LTL can be used. One of the emerging features is the following: the STL fragment in [31] does not allow to specify periodic tasks, but by coupling an agent, by means of an STL task, to agents under LTL tasks, periodic and complex behaviors can be induced. Agents also need to avoid collisions so that both dynamical and task level couplings are present.

Section II summarizes [14] and [31], while Section III describes our implementation. Experimental results are shown in Section IV, followed by conclusions in Section V.

II. Preliminaries

Let $\mathbb{R}^n$ be the n-dimensional real vector space. The set of real and non-negative real numbers are $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$, respectively; $\mathbf{0}_n$ is a vector containing n zeros.

Consider M agents where $M_{\text{LTL}} \leq M$ agents are subject to local LTL tasks, while $M_{\text{STL}} := M - M_{\text{LTL}}$ agents are subject to local STL tasks. Each agent i is described by three states $x_{i,1}$, $x_{i,2}$, and $x_{i,3}$ where $x_{i,1}$ and $x_{i,2}$ describe the agent's two-dimensional position, while $x_{i,3}$ describes the agent's orientation with respect to the x_1-axis. Let $\boldsymbol{x}_i := \begin{bmatrix}\boldsymbol{z}_i^T & x_{i,3}\end{bmatrix}^T \in \mathcal{X} \subseteq \mathbb{R}^3$ where $\boldsymbol{z}_i := \begin{bmatrix}x_{i,1} & x_{i,2}\end{bmatrix}^T$ and $\mathcal{X}$ is the workspace. We model each agent $i \in \{1, \dots, M\}$ by

$$\dot{\boldsymbol{x}}_i(t) = f_i(\boldsymbol{x}_i(t)) + g_i(\boldsymbol{x}_i(t))\boldsymbol{u}_i(t) + \boldsymbol{w}_i(t) \tag{1}$$

where $\boldsymbol{u}_i \in \mathbb{R}^{m_i}$ and $\boldsymbol{w}_i \in \mathbb{R}^3$ are the control input and additive disturbance, respectively. Furthermore, define $\boldsymbol{x} := \begin{bmatrix}\boldsymbol{x}_1{}^T & \dots & \boldsymbol{x}_M{}^T\end{bmatrix}^T$. The multi-agent system is modeled by an undirected graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ [5]. The set of agents is $\mathcal{V} := \{1, 2, \dots, M\}$, while the edge set $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ indicates communication links. We define the *behavior* of agent i to be agent i's trajectory, i.e., the solution $\boldsymbol{x}_i : \mathbb{R}_{\geq 0} \to \mathbb{R}^3$ to (1).

A. Linear Temporal Logic (LTL)

For each $i \in \{1, \dots, M_{\text{LTL}}\}$, consider the set of atomic propositions AP_i; $\alpha_i \in AP_i$ can either be true ($\top$) or false ($\bot$) and, for instance, indicate whether or not an agent is in a certain region or performing a certain action. We assume that the satisfaction of α_i does only depend on the behavior of agent i. The set of linear temporal logic (LTL) formulas is

$$\phi_i ::= \top \mid \alpha_i \mid \neg\phi_i \mid \circ\phi_i \mid \phi_i' \wedge \phi_i'' \mid \phi_i' \, U \, \phi_i'', \tag{2}$$

where ϕ_i' and ϕ_i'' are LTL formulas associated with agent i and $\wedge$, $\neg$, $\circ$, and U denote the conjunction, negation, next, and until operators, respectively. The disjunction ($\vee$), eventually (F), and always (G) operators can be derived similarly [6]. An infinite *word* over the alphabet 2^{AP_i}, where 2^{AP_i} is the power set of AP_i, is an infinite sequence $\sigma_i := \sigma_{i,0}\sigma_{i,1}\sigma_{i,2}\cdots \in (2^{AP_i})^\omega$ where $\sigma_{i,k} \in 2^{AP_i}$ is the set of atomic propositions that are true at time step k. The semantics of LTL are given in [6, Def. 5.6] and stated as a relation $(\sigma_i, k) \models \phi_i$, which means that σ_i satisfies a formula ϕ_i at time step k. For instance, $(\sigma_i, 0) \models \phi_i$ with $\phi_i := G\neg\alpha_i' \wedge F\alpha_i''$ implies that $\forall k \geq 0$, $\alpha_i' \notin \sigma_{i,k}$ (α_i' is always avoided) and $\exists l \geq 0$ such that $\alpha_i'' \in \sigma_{i,l}$ (eventually α_i'' holds). The set of words that satisfy ϕ_i is given by $Words(\phi_i) := \{\sigma_i \in (2^{AP_i})^\omega | (\sigma_i, 0) \models \phi_i\}$. Each ϕ_i can be translated into a language equivalent Büchi Automaton $\mathcal{A}_{\phi_i} := (Q_i, 2^{AP_i}, \Delta_i, Q_{i,0}, \mathcal{F}_i)$ where Q_i is a finite set of states; $Q_{i,0} \subseteq Q_i$ is a set of initial states; 2^{AP_i} is the alphabet; $\Delta_i : Q_i \times 2^{AP_i} \to 2^{Q_i}$ is a transition relation; $\mathcal{F}_i \subseteq Q_i$ is a set of accepting states. The sequence $q_i := q_{i,j_0}q_{i,j_1}q_{i,j_2}\cdots \in Q_i^\omega$ is a *run* of $\mathcal{A}_{\phi_i}$ for σ_i if $q_{i,j_0} \in Q_{i,0}$ and $\Delta_i(q_{i,j_k}, \sigma_{i,k}) = q_{i,j_{k+1}}$ for all $k \geq 0$. The run q_i is accepting if $q_{i,j_k} \in \mathcal{F}_i$ for infinitely many k. Let $\mathcal{L}_\omega(\mathcal{A}_{\phi_i}) := \{\sigma_i \in (2^{AP_i})^\omega | q_i \text{ is an accepting run of } \mathcal{A}_{\phi_i} \text{ for } \sigma_i\}$. There always exists a $\mathcal{A}_{\phi_i}$ with $Words(\phi_i) = \mathcal{L}_\omega(\mathcal{A}_{\phi_i})$ [6, Thm. 5.41]. For more intuition on LTL and this terminology, we refer to [17, Ex. 1] and [6, Sec. 5] as well as the three experiments that we present in Section IV. Each agent $i \in \{1, \dots, M_{\text{LTL}}\}$ has a set of propositions $\Pi_i \subseteq AP_i$ where $\alpha_{i,m} \in \Pi_i$ is associated with a set $\mathcal{X}_{i,m} \subseteq \mathbb{R}^3$ so that $\alpha_{i,m} = \top$ if $\boldsymbol{x}_i \in \mathcal{X}_{i,m}$. A transition from $\alpha_{i,m}$ to $\alpha_{i,n}$ is enabled if a navigation controller $\boldsymbol{u}_i$ exists that is able to drive the agent from any pose in $\mathcal{X}_{i,m}$ to some pose in $\mathcal{X}_{i,n}$ in finite time. Based on this, define a weighted finite-state transition system as a tuple $\mathcal{T}_i := (\Pi_i, \to_i, \Pi_{i,0}, L_i, W_i)$ where Π_i are the poses of interest; $\to_i \subseteq \Pi_i \times \Pi_i$ is the transition relation when there exists a navigation controller $\boldsymbol{u}_i$; $\Pi_{i,0} \subseteq \Pi_i$ is the set of initial regions; $L_i : \Pi_i \to 2^{AP_i}$ is the labeling function; $W_i : \Pi_i \times \Pi_i \to \mathbb{R}_{\geq 0}$ is the weight function associated with a transition. Note that propositions in $AP_i \setminus \Pi_i$ are generic propositions that may hold for some poses of interest. We define an infinite *path* of $\mathcal{T}_i$ as an infinite state sequence $\tau_i = \pi_{i,j_0}\pi_{i,j_1}\dots$ such that $\pi_{i,j_0} \in \Pi_{i,0}$ and a transition $\to_i$ exists from π_{i,j_k} to $\pi_{i,j_{k+1}}$ for all $k \geq 0$. A state sequence induces an infinite word $Word(\tau_i) = L_i(\pi_{i,j_0})L_i(\pi_{i,j_1})\dots$. For a given formula ϕ_i and a path τ_i, we have that $(Word(\tau_i), 0) \models \phi_i$ if $Word(\tau_i) \in Words(\phi_i)$. A weighted product Büchi automaton, capturing the behavior of an agent that satisfies ϕ_i, is defined as $\mathcal{A}_{\phi_i}' := \mathcal{T}_i \times \mathcal{A}_{\phi_i} = (Q_i', 2^{AP_i}, \Delta_i', Q_{i,0}', \mathcal{F}_i', W_i')$, where $Q_i' := \Pi_i \times Q_i, q_i' := \langle \pi_{i,m}, q_{i,m} \rangle \in Q_i'$, for all $\pi_{i,m} \in \Pi_i$ and for all $q_{i,m} \in Q_i$; $\Delta_i' : Q_i' \to 2^{Q_i'}$, $\langle \pi_{i,n}, q_{i,n} \rangle \in \Delta_i'(\langle \pi_{i,m}, q_{i,m} \rangle)$ if and only if $(\pi_{i,m}, \pi_{i,n}) \in \to_i$ and $q_{i,n} \in \Delta_i(q_{i,m}, L_i(\pi_{i,m}))$; $Q_{i,0}' := \Pi_{i,0} \times Q_{i,0}$ is the set of initial states; $\mathcal{F}_i' := \Pi_i \times \mathcal{F}_i$ is the set of accepting

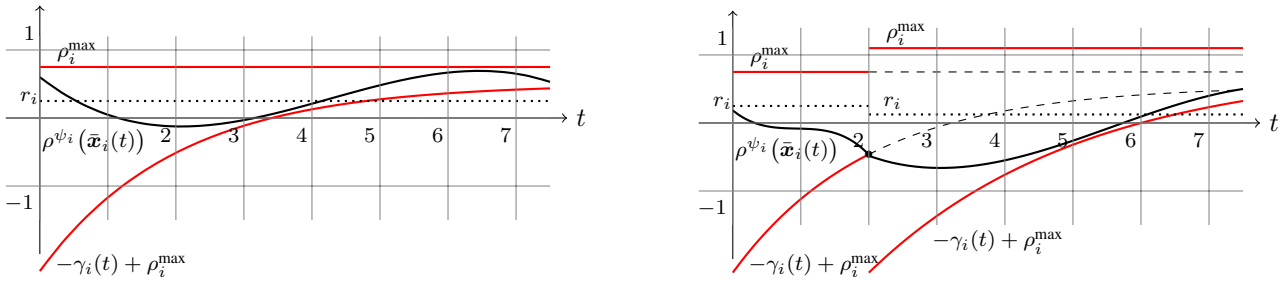


Fig. 2: Funnel-based control law for $G_{[5,7]}\psi_i$ with $r_i := 0.25$ (left). Funnel repairs in the first repair stage (right).

states; $W_i' : \Delta_i' \to \mathbb{R}_{\geq 0}$ is the weight function indicating the Euclidean distance for a transition. By following the approach in [14, Sec. IV] and applying a modified Dijkstra search algorithm [15, Alg. 3], we find a run in prefix-suffix structure $\tilde{q}_i := q'_{i,j_0}q'_{i,j_1}\cdots = \langle \pi_{i,j_0}, q_{i,j_0}\rangle\langle \pi_{i,j_1}, q_{i,j_1}\rangle\ldots$ with $\tau_i := \pi_{i,j_0}\pi_{i,j_1}\ldots$ so that $(Word(\tau_i), 0) \models \phi_i$. This then implies that agent i has to move from region π_{i,j_k} to region $\pi_{i,j_{k+1}}$ for each $k \geq 0$ by applying the corresponding navigation controller $\boldsymbol{u}_i$ to satisfy ϕ.

B. Signal Temporal Logic (STL)

Considering a predicate μ_i associated with an agent $i \in \{M_{\text{LTL}}, +1, \ldots, M\}$, we use the STL fragment of [31] as

$$\psi_i \;::=\; \top \mid \mu_i \mid \neg\mu_i \mid \psi_i' \wedge \psi_i'' \tag{3a}$$
$$\phi_i \;::=\; G_{[a_i,b_i]}\psi_i \mid F_{[a_i,b_i]}\psi_i \tag{3b}$$

where ψ_i' and ψ_i'' in (3a) are formulas of class ψ_i given in (3a). Note that, unlike LTL operators, $G_{[a_i,b_i]}$ (always) and $F_{[a_i,b_i]}$ (eventually) are time restricted on $[a_i, b_i]$ where $a_i, b_i \in \mathbb{R}_{\geq 0}$ with $a_i \leq b_i$. The satisfaction of μ_i depends on the behavior of agent i and possibly on the behavior of other agents $j \in \mathcal{V} \setminus \{i\}$. If the satisfaction of ϕ_i depends on the behavior of $j \in \mathcal{V}$, i.e., on $\boldsymbol{x}_j(t)$, we say that agent v_j is participating in ϕ_i. The formula ϕ_i consequently depends on a set of agents $\mathcal{V}_i := \{v_{j_1}, \ldots, v_{j_{P_i}}\} \subseteq \mathcal{V}$ where P_i indicates the total number of participating agents. Define $\bar{\boldsymbol{x}}_i(t) := \left[\boldsymbol{x}_{j_1}(t)^T \;\ldots\; \boldsymbol{x}_{j_{P_i}}(t)^T\right]^T \in \mathbb{R}^{3P_i}$ and let $\bar{\boldsymbol{x}}_i : \mathbb{R}_{\geq 0} \to \mathbb{R}^{3P_i}$ be the solution to (1). A predicate μ_i is evaluated at time t by its predicate function $h_i : \mathbb{R}^{3P_i} \to \mathbb{R}$ as $\mu_i := \begin{cases} \top \text{ if } h_i(\bar{\boldsymbol{x}}_i(t)) \geq 0 \\ \bot \text{ if } h_i(\bar{\boldsymbol{x}}_i(t)) < 0. \end{cases}$ The semantics of STL are given in [19, Def. 1] and stated as a relation $(\bar{\boldsymbol{x}}_i, t) \models \phi_i$, which means that $\bar{\boldsymbol{x}}_i$ satisfies ϕ_i at t. For instance, $(\bar{\boldsymbol{x}}_i, 0) \models \phi_i$ with $\phi_i := G_{[5,10]}\mu_i' \wedge F_{[10,15]}\mu_i''$ implies that $\forall t' \in [5, 10]$, $h_i'(\bar{\boldsymbol{x}}_i(t')) \geq 0$ and $\exists t'' \in [10, 15]$ such that $h_i''(\bar{\boldsymbol{x}}_i(t'')) \geq 0$. Note that μ_i' and μ_i'' in the above example can encode agent formations, connectivity constraints, or other collaborative tasks that couple agents due to the use of the stacked vector $\bar{\boldsymbol{x}}_i$. STL admits robust semantics [20], while we use a modified version thereof similarly to [31] by under-approximating the min-operator of conjunctions in the original semantics as

$$\rho^{\mu_i}(\bar{\boldsymbol{x}}_i, t) := h_i(\bar{\boldsymbol{x}}_i(t))$$
$$\rho^{\neg\mu_i}(\bar{\boldsymbol{x}}_i, t) := -h_i(\bar{\boldsymbol{x}}_i(t))$$
$$\rho^{\psi_i' \wedge \psi_i''}(\bar{\boldsymbol{x}}_i, t) := -\frac{1}{\eta}\ln\big(\exp(-\eta\rho^{\psi_i'}(\bar{\boldsymbol{x}}_i, t)) + \exp(-\eta\rho^{\psi_i''}(\bar{\boldsymbol{x}}_i, t))\big)$$
$$\rho^{G_{[a_i,b_i]}\psi_i}(\bar{\boldsymbol{x}}_i, t) := \min_{t_1 \in [t+a_i, t+b_i]} \rho^{\psi_i}(\bar{\boldsymbol{x}}_i, t_1)$$
$$\rho^{F_{[a_i,b_i]}\psi_i}(\bar{\boldsymbol{x}}_i, t) := \max_{t_1 \in [t+a_i, t+b_i]} \rho^{\psi_i}(\bar{\boldsymbol{x}}_i, t_1)$$

where $\eta > 0$ with the property that $\lim_{\eta\to\infty} \rho^{\psi_i' \wedge \psi_i''}(\bar{\boldsymbol{x}}_i, t) = \min(\rho^{\psi_i'}(\bar{\boldsymbol{x}}_i, t), \rho^{\psi_i''}(\bar{\boldsymbol{x}}_i, t))$; $\rho^{\phi_i}(\bar{\boldsymbol{x}}_i, t)$ states how robustly $\bar{\boldsymbol{x}}_i$ satisfies ϕ_i at time t and we have $(\bar{\boldsymbol{x}}_i, t) \models \phi_i$ if $\rho^{\phi_i}(\bar{\boldsymbol{x}}_i, t) > 0$ [21, Prop. 16]. Formulas of class ψ_i in (3a) are boolean formulas and t is contained in $\rho^{\psi_i}(\bar{\boldsymbol{x}}_i, t)$ through the composition of ρ^{ψ_i} with $\bar{\boldsymbol{x}}_i$ so that we use the shorthand notation $\rho^{\psi_i}(\bar{\boldsymbol{x}}_i(t))$.

Assumption 1: Each ψ_i in (3b) is: 1) s.t. $\rho^{\psi_i}(\bar{\boldsymbol{x}}_i)$ is concave; 2) well-posed in the sense that $\rho^{\psi_i}(\bar{\boldsymbol{x}}_i) > 0$ implies $\|\bar{\boldsymbol{x}}_i\| \leq \bar{C}$ for some $\bar{C} \geq 0$; 3) η is s.t. $\rho^{\psi_i}(\bar{\boldsymbol{x}}_i) > 0$ for some $\bar{\boldsymbol{x}}_i \in \mathbb{R}^{3P_i}$.

Assumption 2: The function $\boldsymbol{w}_i : \mathbb{R}_{\geq 0} \to \mathbb{R}^3$ is piecewise continuous, $f_i : \mathbb{R}^3 \to \mathbb{R}^3$ and $g_i : \mathbb{R}^3 \to \mathbb{R}^{3\times m_i}$ are locally Lipschitz continuous, and $g_i(\boldsymbol{x}_i)g_i(\boldsymbol{x}_i)^T$ is positive definite.

Remark 1: Part 3 of Assumption 1 implies that ψ_i is satisfiable, i.e., $\exists \bar{\boldsymbol{x}}_i \in \mathbb{R}^{3P_i}$ s.t. $(\bar{\boldsymbol{x}}_i, 0) \models \psi_i$, while Part 2 of Assumption 1 and Assumption 2 guarantee, in conjunction with $\boldsymbol{u}_i$ proposed later, existence of a solution $\boldsymbol{x}_i(t)$ to (1) defined for all $t \geq 0$. Part 1 of Assumption 1 implies that only concave predicate functions $h_i(\bar{\boldsymbol{x}}_i)$ are allowed which includes the class of linear functions; $g_i(\boldsymbol{x}_i)g_i(\boldsymbol{x}_i)^T$ being positive definite implies that $m_i \geq n_i$, see [31, Rem. 1-3].

The controller in [31] relies on two steps. First, a funnel-based control law is derived when all agents in $\mathcal{V}_i$ are subject to the same task. According to [31, Thm. 2], if for all agents $i, j \in \mathcal{V}_i$ we have $\phi_i = \phi_j$ and $(i, j) \in \mathcal{E}$, and all $i \in \mathcal{V}_i$ apply

$$\boldsymbol{u}_{\text{nom},i}(\bar{\boldsymbol{x}}_i, t) := -\epsilon_i(\bar{\boldsymbol{x}}_i, t)g_i(\boldsymbol{x}_i)^T \frac{\partial\rho^{\psi_i}(\bar{\boldsymbol{x}}_i)}{\partial\boldsymbol{x}_i} \tag{4}$$

where $\epsilon_i(\bar{\boldsymbol{x}}_i, t) := \ln\big(-\frac{\xi_i(\bar{\boldsymbol{x}}_i,t)+1}{\xi_i(\bar{\boldsymbol{x}}_i,t)}\big)$, $\xi_i(\bar{\boldsymbol{x}}_i, t) := \frac{\rho^{\psi_i}(\bar{\boldsymbol{x}}_i)-\rho_i^{\max}}{\gamma_i(t)}$ with $\rho_i^{\max}$ and $\gamma_i(t)$ as explained in [31, eq. (6)-(11)] and $\xi_i(\bar{\boldsymbol{x}}_i, t) = \xi_j(\bar{\boldsymbol{x}}_i, t)$ for all $i, j \in \mathcal{V}_i$, then $0 < r_i \leq \rho^{\phi_i}(\bar{\boldsymbol{x}}_i, 0) < \rho_i^{\max}$ so that $(\bar{\boldsymbol{x}}_i, 0) \models \phi_i$; $r_i > 0$ is a parameter that indicates the robustness by which ϕ_i is satisfied. The idea behind (4) is illustrated in Fig. 2 (left); (4) confines $\rho^{\psi_i}(\bar{\boldsymbol{x}}_i(t))$ within the funnel given by the red curves and achieves $-\gamma_i(t) + \rho_i^{\max} < \rho^{\psi_i}(\bar{\boldsymbol{x}}_i(t)) < \rho_i^{\max}$ for all $t \geq 0$. By the choice of γ_i it follows that $r_i \leq \rho^{\phi_i}(\bar{\boldsymbol{x}}_i, 0) < \rho_i^{\max}$. In the second step of [31], dealing with cases where agents

in $\mathcal{V}_i$ are subject to different tasks, the conditions in [31, Thm. 2] do not hold. If (4) is still applied by each agent i, (4) may become singular since $\epsilon_i(\bar{\boldsymbol{x}}_i(t), t) \to \infty$ as $\rho^{\psi_i}(\bar{\boldsymbol{x}}_i(t)) \to \{-\gamma_i(t)+\rho_i^{\max}, \rho_i^{\max}\}$. We use (4) in conjunction with an online detection & repair scheme that takes care of *critical events*, which are the events where $\rho^{\psi_i}(\bar{\boldsymbol{x}}_i(t)) \in \{-\gamma_i(t)+\rho_i^{\max}, \rho_i^{\max}\}$. Upon detection of a critical event, three repair stages are initiated. The *first repair stage* enlarges the funnel as illustrated in Fig. 2 (right) by relaxing r_i, $\rho_i^{\max}$, and γ_i (see [31, Sec. 3.2] for details); r_i remains positive, while γ_i is such that $0 < r_i \leq \rho^{\phi_i}(\bar{\boldsymbol{x}}_i, 0) < \rho_i^{\max}$ if no further critical event occurs. Let N_i indicate the maximum number of critical events in the first repair stage. After detection of N_i critical events in the first stage, the *second repair stage* is enabled if, in case that all agents in $\mathcal{V}_i$ first satisfy ϕ_i collaboratively, there is enough time left so that all agents $j \in \mathcal{V}_i \setminus \{i\}$ can satisfy ϕ_j afterwards. The funnel is enlarged the same way as in the first repair stage and all agents $j \in \mathcal{V}_i \setminus \{i\}$ replace, temporarily, $\epsilon_j(\bar{\boldsymbol{x}}_j, t)$, $\rho^{\psi_j}(\bar{\boldsymbol{x}}_j)$, and ψ_j in (4) by $\epsilon_i(\bar{\boldsymbol{x}}_i, t)$, $\rho^{\psi_i}(\bar{\boldsymbol{x}}_i)$, and ψ_i, respectively, to collaboratively satisfy ϕ_i by [31, Thm. 2], i.e., the agents $j \in \mathcal{V}_i \setminus \{i\}$ are subject to ϕ_i until ϕ_i is satisfied. If the second repair stage is not enabled, the *third repair stage* is enabled. At each further critical event the funnel is relaxed as in Fig. 2 (right), but now by allowing $r_i < 0$. In particular, we set $r_i = r_i - \delta_i$ where $\delta_i > 0$ is a design parameter. As a consequence, the control law in (4) changes since $\epsilon_i(\bar{\boldsymbol{x}}_i, t)$ and $\rho^{\psi_i}(\bar{\boldsymbol{x}}_i)$ change due to the change in r_i, $\rho_i^{\max}$, and γ_i. By [31, Thm. 3], it holds that $\rho^{\phi_i}(\bar{\boldsymbol{x}}_i, 0) \geq \bar{r}_i$ where $\bar{r}_i$ is maximized.

C. Problem Formulation

In the remainder of the paper, we use the term *robot* instead of agent due to the use of mobile robots in the experiments. In particular, we consider a group of mobile robots where the robots have different sensing and actuation capabilities. Each robot indexed by $i \in \{1, \ldots, M_{\text{LTL}}\}$ is subject to an LTL formula ϕ_i of the form (2) and each each robot indexed by $i \in \{M_{\text{LTL}}+1, \ldots, M\}$ is subject to an STL formula ϕ_i of the form (3b). The objectives are, based on Sections II-A and II-B, to implement a control algorithm so that:

- for each $i \in \{1, \ldots, M_{\text{LTL}}\}$, $(\sigma_i, 0) \models \phi_i$,
- for each $i \in \{M_{\text{LTL}}+1, \ldots, M\}$, $\rho^{\phi_i}(\bar{\boldsymbol{x}}_i, 0) \geq \bar{r}_i$ where $\bar{r}_i$ is maximized,
- and, for each $i, j \in \mathcal{V}$ with $i \neq j$, $\|\boldsymbol{x}_i(t) - \boldsymbol{x}_j(t)\| \geq R_{i,j}$ for all $t \geq 0$ where $R_{i,j}$ depends on the agents' geometry, i.e., collisions are mutually avoided.

For the STL approach, we in particular need to account for input limitations that are not considered in [31]. Furthermore, we want to show that Assumption 2 is not restrictive in practice when omnidirectional robots are considered. Finally, our goal is to show the computationally-efficient and robust nature of our control strategies and that considering LTL and STL mission specifications at the same time can be beneficial.

III. Multi-robot systems under LTL and STL

The implementation of our framework, which can be found in [33], is written in *Python* and *C++* and embedded in the *Robot Operating System* (ROS) [34]. In the experiments,

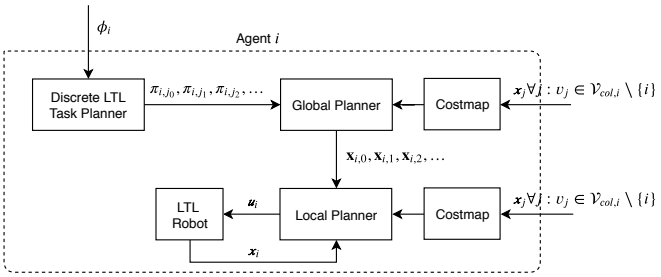


Fig. 3: Control architecture used for robots under LTL tasks.

Turtlebots and Nexus 4WD Mecanum Robotic Cars, omnidirectional robots, are considered (see Fig 1). The omnidirectional robots satisfy Assumption 2, while this is not the case for the Turtlebots. The omnidirectional robots are hence subject to STL tasks, while the TurtleBots are subject to LTL tasks. Due to different sensing and actuation capabilities, different communication and control strategies are derived, resulting in different degrees of decentralization.

LTL tasks: TurtleBots are modelled with $f_i(\boldsymbol{x}_i) := \mathbf{0}_3$ and $g_i(\boldsymbol{x}_i) := \begin{bmatrix}\cos(x_{i,3})u_{i,1} & \sin(x_{i,3})u_{i,1} & u_{i,2}\end{bmatrix}^T$ in (1). Sensing is performed locally by the use of onboard sensors with a directional field of view. Let $\mathcal{V}_{\text{col},i}(t) \subseteq \mathcal{V} \setminus \{i\}$ denote the set of agents that are in the field of view at time t, to be used for collision avoidance. Since LTL tasks are not coupled (as assumed in Section II-A) and due to local sensing, no communication between robots in $i \in \{1, \ldots, M_{\text{LTL}}\}$ is needed. The control architecture is shown in Fig. 3. The discrete LTL task planner, introduced in Section II-A, is further summarized in Algorithm 1 below.

Algorithm 1 Discrete LTL task planner for robot i

1: Form $\mathcal{A}_{\phi_i}$ corresponding to ϕ_i
2: Construct $\mathcal{T}_i$ considering the poses of interest in Π_i
3: Construct the product automaton $\mathcal{A}'_{\phi_i} := \mathcal{T}_i \times \mathcal{A}_{\phi_i}$
4: Apply the modified Dijkstra search algorithm [15, Alg. 3] to obtain $\tilde{q}_i := q'_{i,j_0} q'_{i,j_1} \cdots = \langle \pi_{i,j_0}, q_{i,j_0} \rangle \langle \pi_{i,j_1}, q_{i,j_1} \rangle \ldots$
5: Output $\tau_i := \pi_{i,j_0} \pi_{i,j_1} \ldots$

Navigation according to the obtained high-level LTL plan τ_i while adhering to collision avoidance is achieved by the combination of a global (ROS package *global_planner*) and a local (ROS package *dwa_local_planner*) planner within the *move_base* ROS package. The global planner finds, using a Dijkstra algorithm, a set of sampled waypoints $\mathbf{x}_{i,0}$, $\mathbf{x}_{i,1}$, $\mathbf{x}_{i,2}$, ... which sequentially connect the initial position $\boldsymbol{x}_i(0)$ and the poses of interest π_{i,j_0}, π_{i,j_1}, π_{i,j_2}, ... as closely as possible while taking into account obstacles that may result in a collision. The set of waypoints is then followed in the local planner as accurately as possible by avoiding these obstacles under consideration of the robot dynamics in (1) using the dynamic window approach [35], outputting $\boldsymbol{u}_i(t)$. Both global and local planner take obstacles into account by creating a dynamic cost map using the ROS package *costmap_2d* which is based on the local sensors of each turtlebot. This package

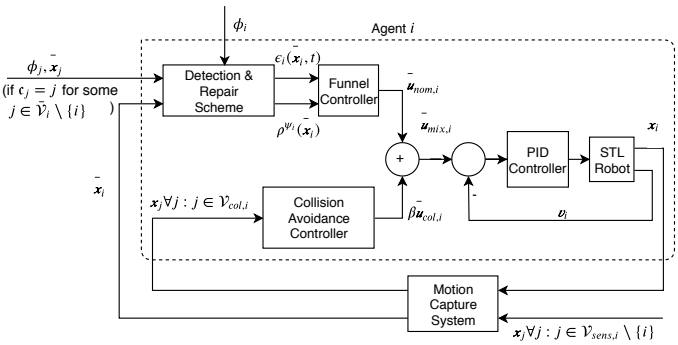


Fig. 4: Control architecture used for robots under STL tasks.

creates a grid of the workspace with different costs for each cell in the grid indicating how close the robot is in the vicinity to an obstacle and how close the robot is to the goal. Local and global cost maps are different since the goals for the global planner are poses of interest π_{i,j_0}, π_{i,j_1}, π_{i,j_2}, ..., while the local planner considers the waypoints $\mathbf{x}_{i,0}$, $\mathbf{x}_{i,1}$, $\mathbf{x}_{i,2}$, ...

STL tasks: Using the approach presented in Section II-B, we assume the dynamics in (1) with unkown $f_i(\boldsymbol{x}_i)$. For the omnidirectional robots, it holds that $m_i := 3$. Since $g_i(\boldsymbol{x}_i)$ is a square matrix, it is straightforward to show that

$$\boldsymbol{u}_{\text{nom},i}(\bar{\boldsymbol{x}}_i, t) := -\epsilon_i(\bar{\boldsymbol{x}}_i, t)\frac{\partial \rho^{\psi_i}(\bar{\boldsymbol{x}}_i)}{\partial \boldsymbol{x}_i} \tag{5}$$

gives the same guarantees as (4), but without knowing $f_i(\boldsymbol{x}_i)$ and $g_i(\boldsymbol{x}_i)$ illustrating the robustness of the controller. Hence, $\boldsymbol{u}_{\text{nom},i}(\bar{\boldsymbol{x}}_i, t)$ as in (5) will be used instead of (4). The control architecture is shown in Fig. 4. For collision avoidance, we again only consider robots within a neighborhood of robot i and within a range of R, where $R > R_{i,j}$ for all $j \in \mathcal{V}$, as $\mathcal{V}_{\text{col},i}(t) := \{j \in \mathcal{V} | \|\boldsymbol{x}_i(t) - \boldsymbol{x}_j(t)\| \leq R\}$. Sensing is done by means of a motion capture system that can be replaced, in experiments performed outside the lab, by using laser range finders [36] or visual odometry [37]. Let us define $\mathcal{V}_{\text{sens},i}(t) := \mathcal{V}_i \cup \mathcal{V}_{\text{col},i}(t)$ to denote the robot information that is needed at time t. In [31], we have shown that the control law $\beta_i \boldsymbol{u}_{\text{col},i}(\boldsymbol{x}) + \boldsymbol{u}_{\text{nom},i}(\bar{\boldsymbol{x}}_i, t)$ with $\beta_i \geq 0$ provides the same guarantees as [31, Thm. 2] when $\boldsymbol{u}_{\text{col},i}(\boldsymbol{x})$, used for collision avoidance, is locally Lipschitz continuous in $\boldsymbol{x}$. In real physical systems, however, the input is limited by $u_{\max,i} \in \mathbb{R}_{\geq 0}$ and hence we first define

$$\bar{\boldsymbol{u}}_{\text{nom},i}(\bar{\boldsymbol{x}}_i, t) := \begin{cases} \boldsymbol{u}_{\text{nom},i}(\bar{\boldsymbol{x}}_i, t) & \text{if } \|\boldsymbol{u}_{\text{nom},i}(\bar{\boldsymbol{x}}_i, t)\| \leq u_{\max,i} \\ u_{\max,i}\frac{\boldsymbol{u}_{\text{nom},i}(\bar{\boldsymbol{x}}_i, t)}{\|\boldsymbol{u}_{\text{nom},i}(\bar{\boldsymbol{x}}_i, t)\|} & \text{otherwise} \end{cases}$$

and $\bar{\boldsymbol{u}}_{\text{col},i}(\boldsymbol{x}) := \begin{bmatrix} \tilde{\boldsymbol{u}}_{\text{col},i}(\boldsymbol{x})^T & 0 \end{bmatrix}^T$ where, for $R_1 < R$,

$$\tilde{\boldsymbol{u}}_{\text{col},i}(\boldsymbol{x}) := \sum_{j \in \mathcal{V}: \|z_i - z_j\| < R_1} u_{\max,i} D(\boldsymbol{z}_i, \boldsymbol{z}_j) + \sum_{j \in \mathcal{V}: R_1 \leq \|\boldsymbol{z}_i - \boldsymbol{z}_j\| \leq R} k_i \left(\frac{R - \|\boldsymbol{z}_i - \boldsymbol{z}_j\|}{\|\boldsymbol{z}_i - \boldsymbol{z}_j\|^3 R}\right) D(\boldsymbol{z}_i, \boldsymbol{z}_j)$$

with $D(\boldsymbol{z}_i, \boldsymbol{z}_j) := \frac{\boldsymbol{z}_i - \boldsymbol{z}_j}{\|\boldsymbol{z}_i - \boldsymbol{z}_j\|}$ and where $k_i := u_{\max,i}\frac{RR_1^3}{R - R_1}$ ensures that $\bar{\boldsymbol{u}}_{\text{col},i}(\boldsymbol{x})$ is continuous. Let $\boldsymbol{u}_{\text{mix},i}(\boldsymbol{x}, t) := \beta_i \bar{\boldsymbol{u}}_{\text{col},i}(\boldsymbol{x}) + \bar{\boldsymbol{u}}_{\text{nom},i}(\bar{\boldsymbol{x}}_i, t)$ so that we define the control law

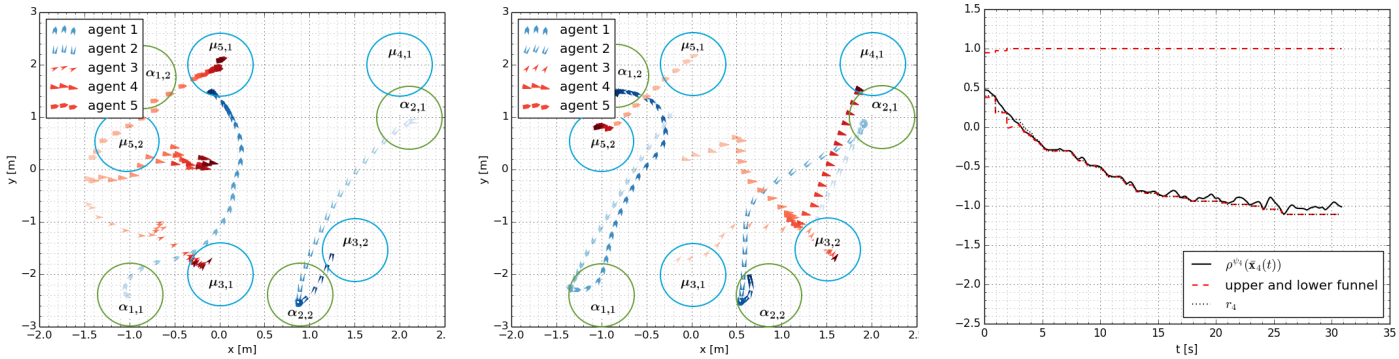
$$\bar{\boldsymbol{u}}_{\text{mix},i}(\boldsymbol{x}, t) := \begin{cases} \boldsymbol{u}_{\text{mix},i}(\boldsymbol{x}, t) & \text{if } \|\boldsymbol{u}_{\text{mix},i}(\boldsymbol{x}, t)\| \leq u_{\max,i} \\ u_{\max,i}\frac{\boldsymbol{u}_{\text{mix},i}(\boldsymbol{x}, t)}{\|\boldsymbol{u}_{\text{mix},i}(\boldsymbol{x}, t)\|} & \text{otherwise.} \end{cases} \tag{6}$$

The parameter β_i determines how much collision avoidance is taken into account. For only two robots, $\beta_i \geq 1$ ensures collision avoidance since $\bar{\boldsymbol{u}}_{\text{col},i}(\boldsymbol{x})$ dominates $\bar{\boldsymbol{u}}_{\text{nom},i}(\bar{\boldsymbol{x}}_i, t)$. The obtained control commands (6) are forwarded to a low-level PID controller that is integrated into each omnidirectional robot to track these velocity commands. Input limitations, the collision avoidance mechanism, and the digital implementation of (6) are the reason why the guarantees of [31, Thm. 2] do not hold anymore. As a result, more critical events may occur. As shown in Section IV, satisfactory behavior can still be achieved when deadlines of the STL task are not too tight due to the detection & repair scheme, summarized in Algorithm 2.

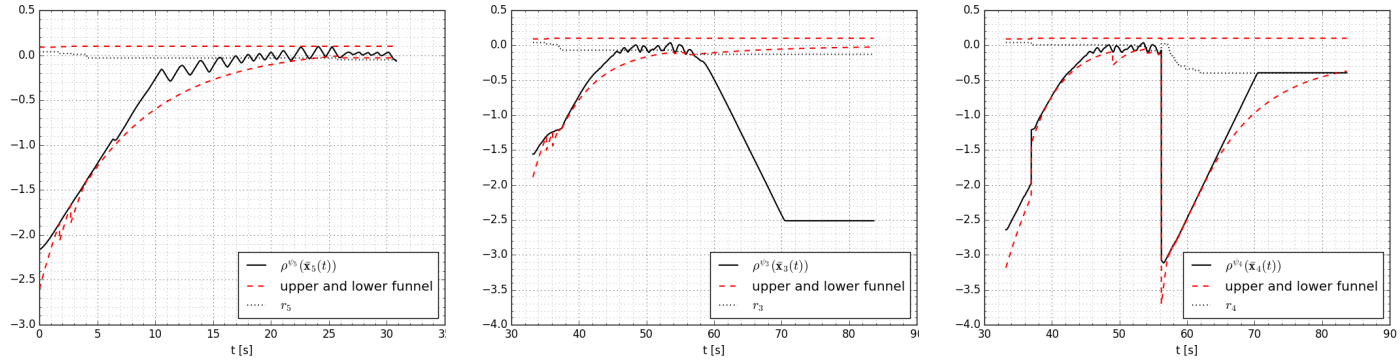
Algorithm 2 Detection & Repair Scheme for robot i

1: Calculate initial r_i, $\rho_i^{\max}$, γ_i [31, Sec. 3.2]
2: Set $ce := 0$ ▷ Counter for critical events
3: Set $\mathfrak{c}_i := 0$ ▷ Indicates 2nd repair stage
4: **repeat**
5: **if** Critical Event and $\mathfrak{c}_j := 0$ for all $i \in \mathcal{V}_j \setminus \{j\}$ **then**
6: Set $ce := ce + 1$
7: **if** $ce \leq N_i$ **then** ▷ 1st repair stage
8: Calculate new r_i, $\rho_i^{\max}$, γ_i [31, Sec. 3.2]
9: **else** ▷ 2nd and 3rd repair stages
10: **if** 2nd repair stage and $\mathfrak{c}_i = 0$ **then**
11: Calculate new r_i, $\rho_i^{\max}$, γ_i [31, Sec. 3.2]
12: Set $\mathfrak{c}_i := i$; send $\rho_i^{\max}$, γ_i, ψ_i to $j \in \mathcal{V}_i \setminus \{i\}$
13: **else if** 2nd repair stage and $\mathfrak{c}_i = i$ **then**
14: $r_i := r_i - \delta_i$; Calculate new $\rho_i^{\max}$, γ_i
15: Send $\rho_i^{\max}$, γ_i, ψ_i to $j \in \mathcal{V}_i \setminus \{i\}$
16: **else** 3rd repair stage
17: $r_i := r_i - \delta_i$; Calculate new $\rho_i^{\max}$, γ_i
18: **end if**
19: **end if**
20: **if** $\mathfrak{c}_j = j$ and $i \in \mathcal{V}_j \setminus \{j\}$ **then** ▷ 2nd repair stage
21: Set $\rho_i^{\max} := \rho_j^{\max}$, $\gamma_i := \gamma_j$, $\psi_i := \psi_j$
22: **end if**
23: **if** $\mathfrak{c}_j$ changed from j to -1 and $i \in \mathcal{V}_j \setminus \{j\}$ **then**
24: Reset ψ_i, Calculate new r_i, $\rho_i^{\max}$, γ_i
25: **end if**
26: **end if**
27: Calculate and output $\epsilon(\bar{\boldsymbol{x}}_i, t)$ and $\rho^{\psi_i}(\bar{\boldsymbol{x}}_i)$
28: **until** $\rho^{\phi_i}(\bar{\boldsymbol{x}}_i, 0) \geq \bar{r}_i$ where $\bar{r}_i$ is maximized.
29: Set $\mathfrak{c}_i := -1$

Algorithm 2 follows Section II-B and uses a parameter $\mathfrak{c}_i$ indicating, when $\mathfrak{c}_i \notin \{0, -1\}$, that the second repair stage has been initiated by agent i (lines 10-12). Agents $j \in \mathcal{V}_i \setminus \{i\}$ then collaborate to satisfy ϕ_i (lines 20-21). If ϕ_i is satisfied, collaboration is terminated and $\mathfrak{c}_i = -1$ (line 29) so that robots in $j \in \mathcal{V}_i \setminus \{i\}$ can continue with ϕ_j (line 24). We obtain $\rho^{\phi_i}(\bar{\boldsymbol{x}}_i, 0) \geq \bar{r}_i$ where $\bar{r}_i$ is maximized (line 28).



(a) Robot trajectories from 0-30 seconds. (b) Robot trajectories from 30-90 seconds. (c) Robust semantics $\rho^{\psi_4'}(\bar{\boldsymbol{x}}_4(t))$ for ϕ_4'.



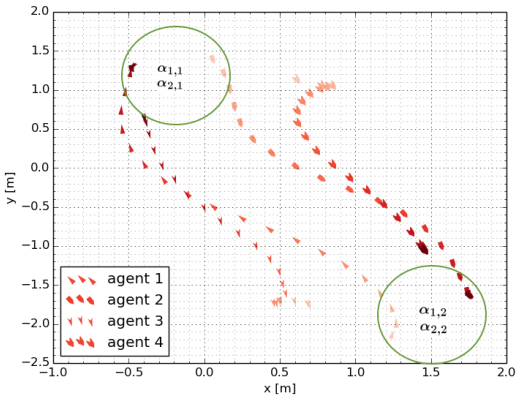
(d) Robust semantics $\rho^{\psi_5'}(\bar{\boldsymbol{x}}_5(t))$ for ϕ_5'. (e) Robust semantics $\rho^{\psi_3''}(\bar{\boldsymbol{x}}_3(t))$ for ϕ_3''. (f) Robust semantics $\rho^{\psi_4''}(\bar{\boldsymbol{x}}_4(t))$ for ϕ_4''.

Fig. 5: Robot trajectories and evolution of the robust semantics for selected robots of Scenario 1.

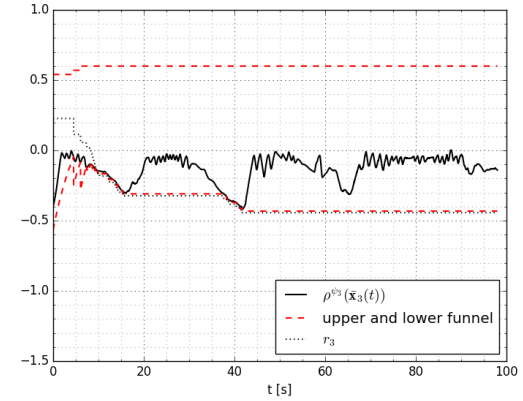
IV. Experimental Results

We now present the experimental results for three different scenarios that were performed on a rectangular workspace of size $[-3.5, -3.5] \times [3.5, 3.5]$ (measured in meters). Videos of the three scenarios can be found in [38]–[40], respectively.

Scenario 1: The first scenario demonstrates the full functionality of the presented framework. We employ $M_{\text{LTL}} = 2$ and $M_{\text{STL}} = 3$ robots. For robots 1 and 2, we define the propositions $\alpha_{1,1}$, $\alpha_{1,2}$, $\alpha_{2,1}$, and $\alpha_{2,2}$ as illustrated in Fig. 5. The LTL tasks are $\phi_1 := GF(\alpha_{1,1} \wedge \alpha_{1,2})$ and $\phi_2 := GF(\alpha_{2,1} \wedge \alpha_{2,2})$ or, in words, robot 1 (robot 2) should periodically visit $\alpha_{1,1}$ and $\alpha_{1,2}$ ($\alpha_{2,1}$ and $\alpha_{2,2}$). For robot 3, define $\mu_{3,1} := (\|\boldsymbol{z}_3 - \begin{bmatrix}0 & -2\end{bmatrix}^T\| \leq 0.1)$, $\mu_{3,2} := (\|\boldsymbol{z}_3 - \begin{bmatrix}1.5 & -1.5\end{bmatrix}^T\| \leq 0.1)$, $\mu_{3,3} := (\|\boldsymbol{z}_3 - \boldsymbol{z}_4\| \leq 0.7)$, $\phi_3' := G_{[21,30]}\mu_{3,1}$, and $\phi_3'' := F_{[57,58]}(\mu_{3,2} \wedge \mu_{3,3})$. Robot 3 is then subject to $\phi_3 := \phi_3' \wedge \phi_3''$ or, in words, always between 21-30 sec be in region $\mu_{3,1}$ and eventually between 57-58 sec be in region $\mu_{3,2}$ while being at least 0.7 m close to robot 4. For robot 4, define $\mu_{4,1} := (\|\boldsymbol{z}_4 - \begin{bmatrix}2 & 2\end{bmatrix}^T\| \leq 0.1)$, $\mu_{4,2} := (\|\boldsymbol{z}_4 - \boldsymbol{z}_3\| \leq 1)$, $\mu_{4,3} := (\|\boldsymbol{z}_4 - \boldsymbol{z}_5\| \leq 1)$, $\phi_4' := G_{[5,30]}(\mu_{4,2} \wedge \mu_{4,3})$, and $\phi_4'' := F_{[83,87]}\mu_{4,1}$. Robot 4 is subject to $\phi_4 := \phi_4' \wedge \phi_4''$ or, in words, always between 5-30 sec be 1 m close to agent 3 and 5 and eventually between 83-87 seconds be in region $\mu_{4,1}$. Robot 5 should always between 21-30 sec be in region $\mu_{5,1}$ and eventually between 44-47 sec be in region $\mu_{5,2}$ where $\mu_{5,1} := (\|\boldsymbol{z}_5 - \begin{bmatrix}0 & 2\end{bmatrix}^T\| \leq 0.1)$ and $\mu_{5,2} := (\|\boldsymbol{z}_5 - \begin{bmatrix}-1 & 0.5\end{bmatrix}^T\| \leq 0.1)$ so that $\phi_5 := \phi_5' \wedge \phi_5''$ where $\phi_5' := G_{[21,30]}\mu_{5,1}$ and $\phi_5'' := F_{[44,47]}\mu_{5,2}$. The trajectories of the robots for 0-30 seconds and 30-90 seconds are shown in Fig. 5a and 5b, respectively. The evolution of a robot over time is indicated by increasing color intensity and it can hence be seen that collisions are avoided. Note that robots 3 and 4 are coupled to other robots. In Fig. 5a, showing ϕ_3', ϕ_4', and ϕ_5', robot 4 needs to stay close to robot 3 and 5, while robots 3 and 5 are supposed to move to $\mu_{3,1}$ and $\mu_{5,1}$, respectively, so that agent 4 can not satisfy ϕ_4'. Robot 4 finds a least violating solution by successively reducing the robustnes r_4 (third repair stage, see Fig. 5c) and consequently staying as close as possible to robots 3 and 5. Robots 3 and 5 satisfy their tasks ϕ_3' and ϕ_5', respectively, as illustrated for robot 5 in Fig. 5d. More formally, we have $\rho^{\phi_3'}(\bar{\boldsymbol{x}}_3, 0) \geq -0.18$, $\rho^{\phi_4'}(\bar{\boldsymbol{x}}_4, 0) \geq -1.11$, and $\rho^{\phi_5'}(\bar{\boldsymbol{x}}_5, 0) \geq -0.12$. In Fig. 5b, showing ϕ_3'', ϕ_4'', and ϕ_5'', robot 3 needs to move to $\mu_{3,2}$ while staying close to robot 4. Robot 4, however, is supposed to move to $\mu_{4,1}$. Therefore, at some point, robot 3 establishes communication with robot 4 to collaboratively satisfy ϕ_3'' (second repair stage, see Fig. 5e). Afterwards, robot 4 continuous with ϕ_4'' (see Fig. 5f). It holds that $\rho^{\phi_3''}(\bar{\boldsymbol{x}}_3, 0) \geq 0.02$, $\rho^{\phi_4''}(\bar{\boldsymbol{x}}_4, 0) \geq -0.4$, and $\rho^{\phi_5''}(\bar{\boldsymbol{x}}_5, 0) \geq 0.5$. This scenario illustrates how the online detection & repair scheme handles critical events so that, even when obstacles need to be avoided or when local tasks are not satisfiable, $\rho^{\phi_i}(\bar{\boldsymbol{x}}_i, 0) \geq \bar{r}_i$ is achieved where $\bar{r}_i$ is maximized.

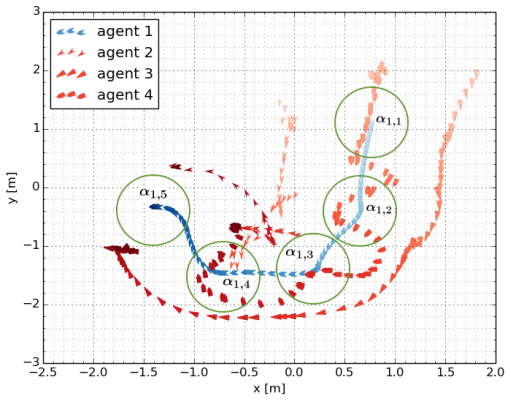


(a) Robot trajectories from 23-45 seconds.

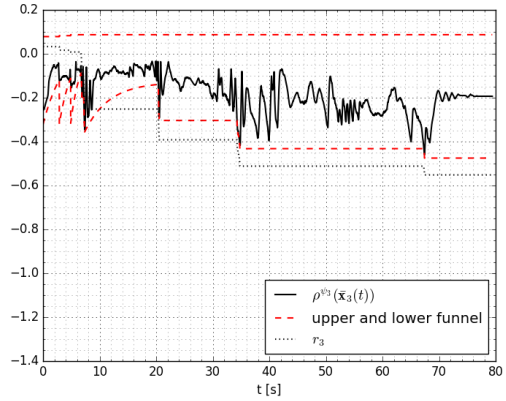


(b) Robust semantics $\rho^{\psi_3}(\bar{\boldsymbol{x}}_3(t))$ for ϕ_3.

Fig. 6: Robot trajectories and robust semantics for Scenario 2.



(a) Robot trajectories.



(b) Robust semantics $\rho^{\psi_3}(\bar{\boldsymbol{x}}_3(t))$ for ϕ_3.

Fig. 7: Robot trajectories and robust semantics for Scenario 3.

As can be seen in the video in [38], the workspace is densely filled with robots so that collision avoidance is the main reason why the robustness $\bar{r}_i$ is decreased.

Scenario 2: In this scenario, we couple robots under STL tasks to robots under LTL tasks to induce periodic motion. Consider $M_{\text{LTL}} = 2$ and $M_{\text{STL}} = 2$ robots. For robots 1 and 2, $\phi_1 := GF(\alpha_{1,1} \wedge \alpha_{1,2})$ and $\phi_2 := GF(\alpha_{2,1} \wedge \alpha_{2,2})$ again encode to periodically visit the regions $\alpha_{1,1}$, $\alpha_{1,2}$ and $\alpha_{2,1}$, $\alpha_{2,2}$, respectively (see Fig. 6a). For robots 3 and 4, define $\mu_3 := (\|\boldsymbol{z}_3 - \boldsymbol{z}_1\| \leq 0.6)$, $\mu_4 := (\|\boldsymbol{z}_4 - \boldsymbol{z}_2\| \leq 0.6)$, $\phi_3 := G_{[0,90]}\mu_3$, and $\phi_4 := G_{[0,90]}\mu_4$, i.e., robots 3 and 4 track robots 1 and 2, respectively. The robots under STL tasks congest the path so that collision are expected. The trajectories of the robots from 23-45 sec are shown in Fig. 6a. The LTL tasks ϕ_1 and ϕ_2 are satisfied while the robots under STL tasks track the robots under LTL tasks closely (see Fig. 6b); ϕ_3 and ϕ_4 are not satisfied due to collision avoidance and the induced repair stages; however, we obtain $\rho^{\phi_3}(\bar{\boldsymbol{x}}_3, 0) \geq -0.45$ and $\rho^{\phi_4}(\bar{\boldsymbol{x}}_4, 0) \geq -0.47$.

Scenario 3: We again couple robots to other robots, but in a more complex way. Consider $M_{\text{LTL}} = 1$ and $M_{\text{STL}} = 3$ robots. We have $\phi_1 := F(\alpha_{1,1} \wedge \alpha_{1,2} \wedge \alpha_{1,3} \wedge \alpha_{1,4} \wedge \alpha_{1,5})$ where the propositions can be seen in Fig. 7a. The method proposed in Section III will find the shortest path satisfying ϕ_1, which is to sequentially move to $\alpha_{1,1}$, $\alpha_{1,2}$, $\alpha_{1,3}$, $\alpha_{1,4}$, and $\alpha_{1,5}$. The robots under STL tasks are supposed to form a formation with respect to the LTL robot and to also track its orientation. Let $\mu_{2,1} := (\|\boldsymbol{z}_2 - \boldsymbol{z}_1 - \begin{bmatrix}\sin(x_{1,3}) & -\cos(x_{1,3})\end{bmatrix}^T) \leq 0.1$ and $\mu_{2,2} := (|x_{2,3} - x_{1,3}|) \leq 0.09$ so that $\phi_2 := G_{[10,100]}(\mu_{2,1} \wedge \mu_{2,2})$ which means, in words, that robot 2 should always be to the right of robot 1 and track its orientation. Let also $\mu_{3,1} := (\|\boldsymbol{z}_3 - \boldsymbol{z}_1 - \begin{bmatrix}-\sin(x_{1,3}) & \cos(x_{1,3})\end{bmatrix}^T) \leq 0.1$ and $\mu_{3,2} := (|x_{3,3} - x_{1,3}|) \leq 0.09$ so that $\phi_3 := G_{[10,100]}(\mu_{3,1} \wedge \mu_{3,2})$, i.e., robot 3 should always be to the left of robot 1 and track its orientation. Similarly, ϕ_4, omitted here, means that robot 4 should always be behind robot 1 and track its orientation. The robustness function for robot 3 is shown in Fig. 7b.

It has been shown that the proposed method is robust in the sense that STL tasks can be satisfied with a given robustness. Input limitations, collision avoidance, the digital implementation of (6), or cases not covered by [31, Thm. 2], however, may induce critical events that are handled by the detection & repair scheme. The computation times of Algorithm 1 are, on average, 0.5 seconds. The control loops for robots under LTL and STL tasks are run with 5 Hz and 100 Hz, respectively.

V. Conclusion

We presented the implementation and experimental results of [14] and [31], which present theoretical frameworks for the control of multi-agent systems under linear temporal logic and signal temporal logic tasks. In particular, each agent is represented by a mobile robot that is either subject to a local linear temporal logic or a local signal temporal logic task. Our implementation deals with practical issues such as collision avoidance, input saturations, the digital implementation of continuous-time feedback control laws, and a controllability assumption that was made in the original works. We also argued that using linear temporal logic and signal temporal logic at the same time can be beneficial. A particular strength of combining these temporal logics is that signal temporal logic tasks can depend on the agents under periodic linear temporal logic tasks. We provided three experiments and have shown that our method can be used as a multi-purpose tool. For the future, we plan to introduce a robustness recovery mechanism that can be integrated into the online detection & repair scheme of the signal temporal logic framework. Thereby, the robustness, by which a signal temporal logic task is satisfied, can again be increased after the occasion of unforeseen events that potentially decreased this robustness.

References

[1] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Trans. Autom. Control*, vol. 50, no. 5, pp. 655–661, 2005.

[2] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Stable flocking of mobile agents, part i: Fixed topology," in *Proc. Conf. Decis. Control.*, Maui, HI, December 2003, pp. 2010–2015.

[3] M. M. Zavlanos and G. J. Pappas, "Distributed connectivity control of mobile networks," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1416–1428, 2008.

[4] D. V. Dimarogonas, S. G. Loizou, K. J. Kyriakopoulos, and M. M. Zavlanos, "A feedback stabilization and collision avoidance scheme for multiple independent non-point agents," *Automatica*, vol. 42, no. 2, pp. 229–243, 2006.

[5] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*, 1st ed. Princeton University Press, 2010.

[6] C. Baier and J.-P. Katoen, *Principles of Model Checking*, 1st ed. Cambridge, MA: The MIT Press, 2008.

[7] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. Autom. Control*, vol. 53, no. 1, pp. 287–297, Feb 2008.

[8] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *Proc. Conf. Robot. Autom.*, Barcelona, Spain, April 2005, pp. 2020–2025.

[9] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive (1) designs," in *Proc. Int. Workshop on VMCAI*, Charleston, SC, 2006, pp. 364–380.

[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, Dec 2009.

[11] D. Maity and J. S. Baras, "Event-triggered controller synthesis for dynamical systems with temporal logic constraints," in *Proc. American Control Conf.*, Milwaukee, WI, June 2018, pp. 1184–1189.

[12] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *Proc. Int. Conf. Hybrid Systems: Comp. Control*, New York, NY, April 2010, pp. 101–110.

[13] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multi-agent motion tasks based on LTL specifications," in *Proc. Conf. Decis. Control*, vol. 1, Nassau, Bahamas, December 2004, pp. 153–158.

[14] M. Guo, J. Tumova, and D. V. Dimarogonas, "Communication-free multi-agent control under local temporal tasks and relative-distance constraints," *IEEE Trans. Autom. Control*, vol. 61, no. 12, pp. 3948–3962, 2016.

[15] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The Int. Journal Robot. Res.*, vol. 34, no. 2, pp. 218–235, 2015.

[16] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic motion specifications," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 48–61, Feb 2010.

[17] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local LTL specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239 – 248, Aug 2016.

[18] Y. E. Sahin, P. Nilsson, and N. Ozay, "Synchronous and asynchronous multi-agent coordination with cLTL+ constraints," in *Proc. Conf. Decis. Control*. Melbourne, Australia: IEEE, December 2017, pp. 335–342.

[19] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proc. FORMATS-FTRTFT*, Grenoble, France, September 2004, pp. 152–166.

[20] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Proc. Int. Conf. FORMATS*, Klosterneuburg, Austria, September 2010, pp. 92–106.

[21] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoret. Comp. Science*, vol. 410, no. 42, pp. 4262 – 4291, Sept 2009.

[22] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Proc. Conf. Decis. Control*, Los Angeles, CA, 2014, pp. 81–87.

[23] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *Proc. Conf. Commun. Control Comp.*, Monticello, IL, Sept 2015, pp. 772–779.

[24] L. Lindemann and D. V. Dimarogonas, "Robust control for signal temporal logic specifications using discrete average space robustness," *Automatica*, vol. 101, pp. 377–387, 2019.

[25] ——, "Control barrier functions for signal temporal logic tasks," *IEEE Control Syst. Lett.*, vol. 3, no. 1, pp. 96–101, 2019.

[26] S. S. Farahani, R. Majumdar, V. S. Prabhu, and S. E. Z. Soudjani, "Shrinking horizon model predictive control with chance-constrained signal temporal logic specifications," in *Proc. American Control Conf.*, Seattle, WA, May 2017, pp. 1740–1746.

[27] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," in *Proc. Robot. Science Systems*, Ann Arbor, MI, June 2016.

[28] S. Sadraddini and C. Belta, "Formal guarantees in data-driven model identification and control synthesis," in *Proc. Int. Conf. Hybrid Systems: Comp. Control*, Porto, Portugal, April 2018, pp. 147–156.

[29] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam, "Fly-by-logic: Control of multi-drone fleets with temporal logic objectives," in *Proc. Int. Conf. Cyber-Physical Syst.*, Porto, Portugal, April 2018, pp. 186–197.

[30] N. Mehdipour, C.-I. Vasile, and C. Belta, "Arithmetic-geometric mean robustness for control from signal temporal logic specifications," in *Proc. American Control Conf.*, Philadelphia, PA, July 2019, pp. 1690–1695.

[31] L. Lindemann and D. V. Dimarogonas, "Feedback control strategies for multi-agent systems under a fragment of signal temporal logic tasks," *Automatica*, vol. 106, pp. 284–293, 2019.

[32] Z. Liu, B. Wu, J. Dai, and H. Lin, "Distributed communication-aware motion planning for multi-agent systems from stl and spatel specifications," in *Proc. Conf. Decis. Control*, Melbourne, Australia, December 2017, pp. 4452–4457.

[33] https://github.com/KTH-SML/CodeRepositoryTCST.

[34] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, May 2009, p. 5.

[35] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, 1997.

[36] H. Surmann, A. Nüchter, and J. Hertzberg, "An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments," *Robot. Autonomous Syst.*, vol. 45, no. 3-4, pp. 181–198, 2003.

[37] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. Conf. Comp. Vision Pattern Recog.*, vol. 1, Washington, DC, June 2004.

[38] https://youtu.be/pUq4r48p6Sg.

[39] https://youtu.be/M8x9WTgaDU4.

[40] https://youtu.be/4FmDKrHC9rc.