Planning of Heterogeneous Multi-Agent Systems Under Signal Temporal Logic Specifications With Integral Predicates

Ali Tevfik Buyukkocak, Member, IEEE, Derya Aksaray, Member, IEEE, and Yasin Yazıcıoğlu, Member, IEEE

Abstract—We address the problem of coordinating the trajectories of heterogeneous multi-agent systems under spatio-temporal specifications. In particular, we consider global Signal Temporal Logic (STL) constraints to express the number and type of agents that should be present at specific locations within the desired time windows. We also introduce an integral predicate to specify cumulative progress that can be achieved asynchronously by multiple agents. In order to generate optimal trajectories, we formulate a mixed-integer linear program whose objective is minimizing the total distance traveled subject to the heterogeneous abstracted dynamics of the agents and a global STL specification including the novel integral predicate. We demonstrate the performance of the proposed method via simulations and experiments with drones.

Index Terms—Formal methods in robotics and automation, multi-robot systems, optimization and optimal control.

I. INTRODUCTION

The availability of inexpensive robotic platforms with advanced sensing, communication, and computation capabilities has significantly expanded the application areas of multi-agent systems. Accordingly, a great deal of research focuses on the design of scalable and provably correct algorithms for achieving cooperative tasks such as formation (e.g., [1]), consensus (e.g., [2], [3]), task allocation (e.g., [4], [5]), or coverage control (e.g., [6], [7]).

In many applications, multi-robot systems with heterogeneous dynamics and capabilities need to achieve tasks with complex specifications. For example, consider the following specification for a multi-robot surveillance mission: “at least one robot with high resolution camera needs to monitor region A in every 5 minutes, and two ground robots need to be at region B simultaneously within a time window [t₁, t₂], and region C should be visited by a ground or aerial robot 10 times in total during the mission.” As the complexity of the mission increases due to such spatial and temporal specifications, it becomes harder to express the requirements as algebraic equations. Alternatively, temporal logics [8] compactly represent such complex requirements. For example, the constraint or objective of each agent in a multi-agent system can be defined by temporal logic, and the multi-agent planning problem solves for each agent trajectory that satisfies the corresponding local specification [9]–[13].

In this study, we use Signal Temporal Logic (STL) [14] which differs from other temporal logics by accommodating explicit time, including predicates in the form of inequalities, and endowing a robustness degree metric quantifying the degree of satisfaction [15], [16]. Such a metric also enables STL to be encoded into some equality and inequality constraints which can be used in formulating optimization problems [17], [18]. The use of STL in multi-agent planning mostly include individual specifications, or partitioning a global specification into each agent [9], [18], [19]. We, on the other hand, aim to satisfy team objectives instead of local specifications.

In this paper, we introduce a novel integral predicate that can be used in an STL formula by allowing the definition of properties such as average or cumulative progress at desired time intervals. For example, some applications may require the agents to service a region multiple times within a given time window without requiring an instantaneous and synchronous satisfaction. Multiple agents of different types can contribute to the progress over time in such preemptable tasks. This kind of specifications cannot be defined via conventional STL predicates whose satisfactions depend only on a single time step. Unlike the recent studies of [20]–[23], which propose modified quantitative semantics for STL, our new predicate definition preserves standard STL features and enables specifying cumulative properties.

This work is closely related to [24], [25], and [26]. The main differences of this paper from those works are as follows.

Firstly, none of those works are capable of specifying a cumulative progress as captured by the novel integral predicate in this paper. Secondly, [25] and [26] consider only identical agents and do not account for any heterogeneity. While [24] considers heterogeneity in capabilities, it does not account for heterogeneous dynamics, which is a major challenge when planning the trajectories of different agent types. To reflect heterogeneity among agents, we use an abstract model of the
agent dynamics defining different velocities for different agent types.

The paper is organized as follows: Section II introduces preliminaries on STL and graph theory. Section III formulates the heterogeneous multi-agent coordination problem under a global temporal logic constraint. Section IV presents the solution approach with the extension of STL by the integral predicates to define cumulative properties together with mixed-integer linear program (MILP) encoding of the problem. Section V discusses the simulation and experiment results. Finally, Section VI concludes the paper and provides some future directions.

II. PRELIMINARIES

A. Notation

In the following sections, $\mathbb{R}$ and $\mathbb{Z}$ represent the set of real and integer numbers, respectively. More specifically, $\mathbb{R}_{\geq 0}$ ($\mathbb{Z}_{\geq 0}$) is the set of nonnegative real (integer) numbers, and $\mathbb{R}^+$ ($\mathbb{Z}^+$) is the set of positive real (integer) numbers. While $\mathbb{R}^n$ is the set of all $n$-dimensional vectors, $\mathbb{R}^{m \times n}$ ($\mathbb{Z}^{m \times n}$) denotes the set of all real (integer) valued $m \times n$ matrices. The expression of $\mathbf{1}_{m \times 1}$ is an $m$ element-column vector of ones. For any $X \in \mathbb{R}^{m \times n}$, we use $\|X\|$ to denote the element-wise $l_1$-norm.

$$\|X\| = \sum_{i=1}^{m} \sum_{j=1}^{n} |X_{i,j}|,$$

where $|X_{i,j}|$ is the absolute value of $X_{i,j}$. Operator $|\cdot|$, is also used to designate the cardinality of a set.

For any $Y, Y' \in \mathbb{R}^n$, we use $Y \leq Y'$ (or $Y < Y'$) to denote the element-wise inequalities, i.e., $Y_i \leq Y'_i$ (or $Y_i < Y'_i$) for all $i = 1, 2, \ldots, n$.

B. Signal Temporal Logic

Signal Temporal Logic [14] can compactly express rich time series. In this paper, we use the following STL fragment:

$$\phi := \mu \mid \neg \phi \mid \phi_1 \& \phi_2 \mid F_{[t_1,t_2]} \phi,$$

where $t_1, t_2 \in \mathbb{R}_{\geq 0}$ are time bounds with $t_1 \leq t_2 < \infty$; $F_{[t_1,t_2]}$, $\neg$, $\&$ are finally (i.e., eventually), negation, and conjunction operators, respectively; $\phi$ is an STL formula, and $\mu$ is a predicate in the inequality form such as $\mu = g(s) \geq e$ with a constant $e \in \mathbb{R}$, a signal $s : \mathbb{Z}_{\geq 0} \rightarrow \mathbb{R}$ mapping each time instant $k = t/\Delta t \in \mathbb{Z}_{\geq 0}$ to a signal value, and a signal function $g : \mathbb{R}^n \rightarrow \mathbb{R}$. We assume that all the time bounds $t_1, t_2$ that appear in the specifications are exactly divisible by some time step $\Delta t$.

Other useful operators are generated from the others as follows:

$$G_{[t_1,t_2]} \phi = \neg F_{[t_1,t_2]} \neg \phi$$

is globally (i.e., always) operator, and $\phi_1 \& \phi_2 = \neg (\neg \phi_1 \& \neg \phi_2)$ is disjunction operator. We can also define an implication operator as $\phi_1 \Rightarrow \phi_2 = \neg \phi_1 \& \phi_2$.

For the signal $s$ representing the run of the system, let $s_t$ denote the value of $s$ at time $t$ and $(s, t)$ be the part of the signal that is a sequence of $s_t$ for $t' \in [t, \infty)$. Satisfaction by the signal $(s, t)$ is then determined as

$$(s, t) \models \mu \iff g(s_t) \geq e,$$

$$(s, t) \models \neg \mu \iff \neg((s, t) \models \mu),$$

$(s, t) \models \phi_1 \land \phi_2 \iff (s, t) \models \phi_1$ and $(s, t) \models \phi_2,$

$(s, t) \models \phi_1 \lor \phi_2 \iff (s, t) \models \phi_1$ or $(s, t) \models \phi_2,$

$(s, t) \models G_{[t_1,t_2]} \phi \iff \forall t' \in [t+t_1, t+t_2], (s, t') \models \phi,$

$(s, t) \models F_{[t_1,t_2]} \phi \iff \exists t' \in [t+t_1, t+t_2], (s, t') \models \phi.$

The expressions of $(s, t) \models F_{[t_1,t_2]} \phi$ and $(s, t) \models G_{[t_1,t_2]} \phi$ imply that $\phi$ holds at some time instant and all time instants between $[t + t_1, t + t_2]$, respectively. Horizon of an STL formula, $hrz(\phi)$, is informally defined as the minimum amount of time required to decide whether the formula is satisfied or not [27]. Formally, $hrz(\phi)$ is found recursively as follows:

$$\mu = g(s) \geq e \implies hrz(\mu) = 0,$$

$$\phi = \neg \psi \implies hrz(\phi) = hrz(\psi),$$

$$\phi = \bigwedge_{i=1}^{m} \phi_i \text{ or } \bigvee_{i=1}^{m} \phi_i \implies hrz(\phi) = \max_{i \in \{1, \ldots, m\}} hrz(\phi_i),$$

$$\phi = G_{[t_1,t_2]} \mu \text{ or } \phi = F_{[t_1,t_2]} \mu \implies hrz(\phi) = t_2,$$

$$\phi = G_{[t_1,t_2]} \psi \text{ or } \phi = F_{[t_1,t_2]} \psi \implies hrz(\phi) = t_2 + hrz(\phi).$$

Apart from other temporal logics most of which denote the satisfaction of a specification as either True or False, STL is endowed with a metric called “robustness degree” that quantifies how well (or how bad) an STL specification is satisfied (or violated) by measuring the distance to violation (or satisfaction). More formally, robustness degree, $r(s, \phi, t) \in \mathbb{R}$, is a real-valued function that is used to quantify the satisfaction of an STL formula $\phi$ with respect to a signal $(s, t)$. While positive robustness degree indicates the satisfaction of $\phi$, negative one represents violation, and it can be recursively defined as follows [16]:

$$r(s, g(s) \geq e, t) = g(s_t) - e,$$

$$r(s, \neg(g(s) \geq e), t) = -r(s, g(s) \geq e, t),$$

$$r(s, \phi_1 \land \phi_2, t) = \min(r(s, \phi_1, t), r(s, \phi_2, t)),$$

$$r(s, \phi_1 \lor \phi_2, t) = \max(r(s, \phi_1, t), r(s, \phi_2, t)),$$

$$r(s, \mu, t) = \max_{t' \in [t+t_1, t+t_2]} r(s, \phi, t'),$$

$$r(s, G_{[t_1,t_2]} \phi, t) = \min_{t' \in [t+t_1, t+t_2]} r(s, \phi, t').$$

In the rest of the paper, the time of the signals starting from $t = 0$ is omitted, i.e., $(s, 0)$ will be denoted by $s$.

C. Graph Theory

A weighted graph is a tuple $G = (\mathcal{V}, \mathcal{E}, W)$ where $\mathcal{V}$ is a set of nodes, and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is a set of edges between the nodes with weight $W : \mathcal{E} \rightarrow \mathbb{R}^+$. A node $v_i \in \mathcal{V}$ is said to be adjacent to another node $v_j \in \mathcal{V}$ if $(v_i, v_j) \in \mathcal{E}$. The adjacency matrix between $v_i$ and $v_j$ is defined as:

$$A_{i,j} = \begin{cases} 1 & \text{if } (v_i, v_j) \in \mathcal{E}, \\ 0 & \text{otherwise}. \end{cases}$$
III. PROBLEM STATEMENT

This paper aims to generate coordinated trajectories for robots with heterogeneous capabilities and dynamics to achieve cooperative tasks with complex spatio-temporal specifications. For example, the agents can be asked to perform several missions in more than one region within different time windows. Also, some parts of the mission may require collaboration among agents with different types (such as drones with high-resolution cameras and ground robots with heat sensors).

Another type of mission that requires coordination is the achievement of preemptable tasks, i.e., tasks that require an accumulation of certain service over time. For example, an agent may start collecting data in a region and then leave before sampling the whole region to serve a different task. In that case, some other agent may later go to the same region and finish the data collection task by sampling the remaining parts.

Let us start with the agent-type-specific environment definition based on the respective regions of interests.

A. Environment Model

In this paper, we assume that the pairwise paths between the regions of interest for each agent type are generated via sampling based algorithms. These regions include the base and the service areas that need to be visited by the agents of the respective type. We model the partitioned environment of paths for each type \( w \) as a graph \( G_w = (\Sigma_w, A_w, d_w) \) where \( \Sigma_w \) is the set of nodes containing the regions of interest and the intermediate nodes, \( A_w \subseteq \Sigma_w \times \Sigma_w \) denotes the set of transitions between the nodes that can be traversed in one time step with cost of \( d_w : A_w \rightarrow \mathbb{R}^+ \) denoting the Euclidean distance between two nodes. With this approach, we can define heterogeneous dynamics for single integrator agents by setting different transition costs for agent types moving with different speeds, such that \( d_w/\Delta t \leq v_{\text{max}}^{-w} \). For example, if type 1 agents are twice faster than type 2 agents, then we build \( G_1 \) and \( G_2 \) with \( 2d_1 = d_2 \). A sampling based algorithm (e.g., RRT* [28]) can be used to generate such type-specific environment models with transition costs in accordance with the maximum speed agents from different types can attain.

B. Swarm Dynamics

Consider \( l \) agent types each of which is formed by \( L_w \) identical agents where \( w \in W \) for \( W = \{1, \ldots, l\} \). We can construct discrete-time vectors as the signals to be used in STL specifications to track the global behavior of the swarm as \( N_w(k) = [n_w(1,k), n_w(2,k), \ldots, n_w(|\Sigma_w|, k)]^T \in \mathbb{Z}_{\geq 0}^{|\Sigma_w|} \), where \( n_w(\sigma,k) : W \times \Sigma_w \times \mathbb{Z}_{\geq 0} \rightarrow \mathbb{Z}_{\geq 0} \) refers to the number of type \( w \) agents at node \( \sigma \in \Sigma_w \), and \( k \) is the step number such that \( k = \Delta t \). Each agent is represented as a point mass with single integrator dynamics and can move to one of its adjacent nodes within a time step. Furthermore, if the nodes of different agent types intersect, they are assumed to be large enough to accommodate all of the agents which are required to be over that node at the same time, and collision avoidance is kept out of the scope of this paper along with the low-level trajectory plans.

Swarm dynamics can now be constructed to formulate the evolution of \( N_w(k) \) for each type \( w \in W \). The number of type \( w \) agents moving from \( i^{th} \) node to \( j^{th} \) node at time instant \( k \) is denoted as \( u_{wi,j}(k) \in \mathbb{Z}_{\geq 0} \). For any agent type \( w \), the flow over the type-specific environment graph can be represented as:

\[
\begin{bmatrix}
u_{w1,1}(k) & \cdots & u_{w1,|\Sigma_w|}(k) \\
\vdots & \ddots & \vdots \\
u_{w|\Sigma_w|,1}(k) & \cdots & u_{w|\Sigma_w|,|\Sigma_w|}(k)
\end{bmatrix}
\tag{7}
\]

For each agent type \( w \), the number of agents leaving specific nodes is represented as the column-vector below,

\[
u_{w}^{\text{out}}(k) = u_{w}^{T}(k) \mathbf{1}_{|\Sigma_w|} \tag{8}
\]

where each element of the \( u_{w}^{\text{out}}(k) \in \mathbb{Z}_{\geq 0}^{|\Sigma_w|} \) displays the number of type \( w \) agents leaving from the corresponding node at \( k^{th} \) step. Note that the size of \( u_{w}^{\text{out}}(k) \) is the same with \( N_w(k) \). Similarly, for each agent type \( w \), the number of agents moving into specific nodes at time instant \( k \) can be represented with a column-vector as follows:

\[
u_{w}^{\text{in}}(k) = u_{w}^{T}(k) \mathbf{1}_{|\Sigma_w|} \tag{9}
\]

where each element of the \( u_{w}^{\text{in}}(k) \in \mathbb{Z}_{\geq 0}^{|\Sigma_w|} \) shows the number of type \( w \) agents entering into the corresponding node. Again the size of \( u_{w}^{\text{in}}(k) \) is the same with \( N_w(k) \). The swarm dynamics that is used to determine the evolution of \( N_w(k) \) can then be stated as:

\[
N_w(k+1) = N_{w}(k) + u_{w}^{\text{in}}(k) - u_{w}^{\text{out}}(k). \tag{10}
\]

Furthermore, the system should also satisfy

\[
N_w(k) \geq u_{w}^{\text{out}}(k), \tag{11a}
\]

\[
A_{wi,j} = 0 \implies u_{wi,j} = 0, \tag{11b}
\]

where \( A_{wi,j} \) is the indicator of adjacency between \( i^{th} \) and \( j^{th} \) nodes of \( \Sigma_w \). While (11a) ensures that the number of agents leaving a node cannot be larger than the number of agents present at that node, (11b) prevents the flow of agents to the nodes which are infeasible to reach within a single time step.

Objectives in a multi-agent mission can include requirements defined over space and time, i.e., spatio-temporal specifications. These can be expressed as temporal logic formulas using the regions/nodes of interest combined with the desired type/capability and number of agents that are needed to satisfy the mission specifications. After defining the agent-type-specific environment and swarm dynamics, we can now state the problem that solves for the team trajectories satisfying mission specifications.

C. Coordination Problem Under STL Constraint

Suppose that a heterogeneous multi-agent system is initially distributed over an environment. Let \( \Phi \) be a global STL specification having a horizon of \( h r g(\Phi) \). The optimal control problem we address is optimizing an objective function (e.g., minimizing
the total agent movement) subject to the swarm dynamics and the satisfaction of $\Phi$.

**Problem 1:** Let a multi-agent system comprising $l$ agent types evolve over the environment graphs $G_w = (\Sigma_w, A_w, d_w)$ according to the dynamics of (10), (11a), (11b) for $w \in \{1, \ldots, l\}$. Given a global STL specification $\Phi$, find an optimal flow of agents $u^*_w = [u^*_w(0), u^*_w(1), \ldots, u^*_w(H - 1)] \in \mathbb{Z}_{\geq 0}^{(|\Sigma_w| +|\Sigma_w|H)}$ subject to the dynamics and STL constraints:

$$
\begin{align*}
\mathbf{u}^* &= \arg \min \sum_{k=0}^{H-1} \mathcal{J}(u_1(k), \ldots, u_l(k)) \\
\text{s.t.} &\quad (10), (11a), (11b), \\
&\quad N_w(0) = N_w, \quad \forall w \in \{1, \ldots, l\}, \\
&\quad r(N, \Phi, 0) \geq 0,
\end{align*}
$$

where $N = [N_1 \ldots N_l]$ and $\mathbf{u} = [\mathbf{u}_1 \ldots \mathbf{u}_l]$ are the aggregated agent distribution and flow policy, respectively, $H > h_{\text{hrz}}(\Phi)/\Delta t$ is the overall mission horizon, $\mathcal{J}(u_1(k), \ldots, u_l(k))$ is the running cost as a function of agent flow, $N_w$ is the initial agent distribution over the environment $G_w = (\Sigma_w, A_w, d_w)$, and $r(N, \Phi, 0) \geq 0$ enforces the satisfaction of STL specification by requiring the robustness degree to be nonnegative.

Note that using conventional STL, $\Phi$ cannot define tasks including cumulative properties such as reward collection or any other preemptable task. Hence, we define new predicates to specify cumulative progress within a desired time interval in the next section.

### IV. Solution Approach

#### A. Integral Predicates

Although conventional STL can express rich time series, together with its existing extensions, it is limited by the predicates whose satisfactions are determined considering the signal value at a single time instant. In other words, these predicates are memoryless, and not all the signal values at multiple different time instants affect the satisfaction of a conventional predicate.

In the case of progressive events, on the other hand, evaluation of the contributions of the past, future, or both signal values can be useful to determine a satisfaction at the current time. Generally, the success may depend on the accumulation of the signal values at different time instants. We can define such success criteria inside the same predicate. The integral of a signal over a given bounded time interval can be used to assess such a satisfaction.

We introduce an integral predicate to express specific amount of progress in preemptable and cumulative properties via STL specifications.

**Definition 1:** (Integral predicate) An integral predicate over a bounded time interval $[a, b] \subset \mathbb{R}$ with $b \geq a$ is defined as:

$$
\mu_{[a,b]}^i = \sum_{\tau=a/\Delta t}^{b/\Delta t} f(s_{\tau+\Delta t}) \Delta t \geq e,
$$

where $e \in \mathbb{R}$ is a constant, $s : \mathbb{Z}_{\geq 0} \rightarrow \mathbb{R}^n$ is the signal, $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a function to evaluate the progress within $[a, b]$, and $\Delta t$ is the time step.

Notice that the integral predicate requires the input of the time bounds as the temporal operators like eventually and globally. The bounds of the integral can be explicitly defined, and a certain threshold on the progress over a certain time interval can be specified with the proposed integral predicate. The satisfaction of $\mu_{[a,b]}^i$ is determined at $t = k\Delta t$ as:

$$
(s, t) \models \mu_{[a,b]}^i \iff \sum_{\tau=k+a/\Delta t}^{k+b/\Delta t} f(s_{\tau+\Delta t}) \Delta t \geq e,
$$

$$
(s, t) \models \neg \mu_{[a,b]}^i \iff \neg \left((s, t) \models \mu_{[a,b]}^i\right).
$$

Note that the signal $s$ is undefined for $t < 0$; therefore, we assume $t + a \geq 0$ throughout the paper. By preserving the quantitative semantics for the common operators defined in (5), we can quantify the satisfaction of the integral predicate similarly as:

$$
\begin{align*}
&\quad r(s, \mu_{[a,b]}^i, t) = \sum_{\tau=k+a/\Delta t}^{k+b/\Delta t} f(s_{\tau+\Delta t}) \Delta t - e, \\
&\quad r(s, \neg \mu_{[a,b]}^i, t) = -r(s, \mu_{[a,b]}^i, t).
\end{align*}
$$

While positive robustness degree indicates the satisfaction of the predicates, e.g., $r(s, \mu_{[a,b]}^i, t) > 0 \Rightarrow (s, t) \models \mu_{[a,b]}^i$, negative one represents a violation $r(s, \mu_{[a,b]}^i, t) < 0 \Rightarrow (s, t) \not\models \mu_{[a,b]}^i$. Although zero robustness degree, e.g., $r(s, \mu_{[a,b]}^i, t) = 0$, is inconclusive, in this paper, we consider this case as a satisfaction as well.

**Remark 1:** An integral predicate $\mu_{[a,b]}^i$ for $[a, b] \subset \mathbb{R}$ can depend both on the past and future signal values by enabling negative time bounds.

We can define negative bounds on the integral predicate; however, it should be noticed that these are defined with respect to the time bounds of the outer temporal operators such as eventually and globally. We can still use the integral predicate alone by using nonnegative time bounds or defining it with the negative time bounds at future time steps. For a better understanding of the nested use of the integral predicate with the temporal operators, we can extend the horizon definition for the STL specifications given in (4) by introducing integral predicate time bounds as follows:

$$
\begin{align*}
\mu_{[a,b]}^i &= \sum_{\tau=a/\Delta t}^{b/\Delta t} f(s_{\tau+\Delta t}) \Delta t \geq e, \\
\Rightarrow hrz(\mu_{[a,b]}^i) &= \max (|a|, b - b) , \\
\phi &= G_{[t_1, t_2]} \mu_{[a,b]}^i \text{ or } \phi = F_{[t_1, t_2]} \mu_{[a,b]}^i \\
\Rightarrow hrz(\phi) &= \max(t_2 - t_1 + b),
\end{align*}
$$

where $t_1, t_2 \in \mathbb{R} \geq 0$ and $a, b \in \mathbb{R}$ are time bounds with $t_2 \geq t_1$, $b \geq a$, and a constraint of $t_1 + a \geq 0$ for the sake of nonnegative global time.
B. Comparison With the Existing Metrics

Conventional robustness degree of STL evaluates a signal with respect to the critical time instants and neglects the remaining parts of the signal. To overcome this issues, the authors of [29] and [30] defined new metrics to capture the duration of predicate satisfaction compared to the best/worst value of the satisfaction. Similarly, the authors in [31] proposed a temporal operator to specify how long a predicate must be satisfied within a bounded time interval. Yet the success definition in such studies depend on the multiple instantaneous satisfactions of the predicates and misses a notion of cumulative success.

The authors of [20]–[23] introduced new robustness degree metrics that quantify average and cumulative properties. These metrics may be used as predicates to specify thresholds over cumulative properties. However, they are defined by modifying the quantitative semantics of existing temporal operators such as “globally” and “eventually”. In this paper, we introduce the integral predicate as a new operator with its own qualitative and quantitative semantics that can easily be used with the standard STL syntax and semantics. For example, consider a signal \( x = \{1, 1, 1, 1, 1, 2\} \). Suppose that the sum of any two consecutive signal values need to eventually be at least 3. The proposed integral predicate facilitates to define this specification as \( \varphi = F_{[0,4]}[\mu_{[1,3]}^i x] \geq 3 \), for which the robustness degree of \( x \) with respect to \( \varphi \) is \( r(x, \varphi) = 0 \). Thus, \( x \) barely satisfies the specification \( \varphi \). Now, consider the cumulative robustness degree [20] that is the closest measure to our proposed idea. One can express the task in \( \varphi \) by using the cumulative robustness degree as \( \varphi' = F_{[0,4]}[\rho^+(x, F_{[1,3]} x \geq 0) \geq 3] \), for which \( \rho^+(x, \varphi') = -4 \) implying a violation while \( x \) satisfies \( \varphi \).

Furthermore, the difference is more dramatic when compared with [21]–[23] which also modify the quantitative semantics of STL, e.g., using “geometric mean”. This yields a failure to decide the satisfaction of cumulative properties even with the thresholds applied on the robustness metrics. Moreover, depending on the sign of the signal values, modifications are required such as combining different metrics or using different temporal operators to represent the same cumulative properties for different signals. Overall, there is no specific syntax in STL that can capture the cumulative properties, and the proposed integral predicate is introduced to facilitate the definition of cumulative signal properties via STL.

C. Mission Requirements Including Cumulative Progress

Let’s introduce a service proposition, \( S = \{p, e, c\} \), which can be interpreted as: “within given time interval, service \( c \) must be given in region \( p \) for at least \( e \) many times in total.” Here \( p \in P \) is the region to be serviced with \( P \) being the set of all regions of interest, i.e., \( P = \bigcup_{\bar{a}, \bar{b}} P_{\bar{a}, \bar{b}} \), \( c \in C \) is the type of service to be given with \( C \) being the set of all service types, and \( e \) is the total number of service instances. Note that the service proposition \( S \) does not require a synchronous and instantaneous satisfaction.

Now, we will use the integral predicate combined with a service proposition as \( \mu_{[a,b]}^i S \), i.e., \( S \) should be satisfied within \([a, b]\). We can determine the satisfaction of the integral predicate with the service proposition as follows:

\[
(s, t) \models \mu_{[a,b]}^i S \iff \sum_{\tau = k + a / \Delta t}^{k + b / \Delta t} f(s_{\tau \Delta t}) \geq 1, \quad (17)
\]  

where \( f(s_{\tau \Delta t}) : W \times P \times Z_{\geq 0} \rightarrow \mathbb{R}_{\geq 0} \) is a progress function which maps each signal value at time instant \( \tau \) to a progress rate that needs to sum to at least 1 for satisfaction. The selection of \( f(s) \) depends on the requirements of the mission. For a heterogeneous multi-agent mission in which multiple agent types can give the same service with a different efficiency, we define it as:

\[
f(s_{k\Delta t}) = \sum_{w=1}^{t} \frac{\text{cap}_{w} \cdot n_{w}(p, k)}{e}, \quad (18)
\]

where \( \text{cap}_{w} \in \{0, 1, 2\} \) indicates the capability of agents with type \( w \) to implement task \( c \). Since the threshold \( e \) denotes the number of instances, and we define the capability of agents as the number of service instances that can be given in a single time step, we do not need \( \Delta t \) multiplication in (17) anymore. Our capability metric definition is similar to [24]; however, we can 1) assign how fast a task can be performed by different agent types, i.e., introduced weighted capabilities, 2) request the presence of certain agent types in addition to asking for random agent types with a specified capability. While \( \text{cap}_{w} = 0 \) implies that the task \( c \) cannot be completed by type \( w \) agents, \( \text{cap}_{w} = 1 \) and \( \text{cap}_{w} = 2 \) denote the ability to service in regular and more efficient (twice faster) ways, respectively. Again, the capability values are user-defined and subject to change as per task and agent type. For instance, let a specification be given as \( \mu_{[0,10]}[A, \text{Take Picture}, 20] \) implying that region \( A \) must be imaged by agents equipped with a camera at least 20 times between \([0, 10]\). Note that this cannot be satisfied by a single agent due to tight time constraints unless it can take two pictures at a time step. However, two agents would be able to satisfy it each by taking a picture per each time step in \( A \) for 10 time steps; or 20 agents could satisfy the specification by servicing \( A \) only for one time step. Such flexible specifications can be represented by the conventional STL syntax only in an impractical manner with the disjunction of all possible combinations as below:

\[
\mu_{[0,10]} f(s_{t}) \geq 1 \equiv F_{[0,10]} f(s_{t}) \geq 1 \\
\lor (F_{[0,9]} f(s_{t}) \geq 0.9 \land F_{[9,10]} f(s_{t}) \geq 0.1) \\
\lor (F_{[0,9]} f(s_{t}) \geq 0.8 \land F_{[9,10]} f(s_{t}) \geq 0.2) \\
\vdots \\
\lor (F_{[0,8]} f(s_{t}) \geq 0.9 \land F_{[8,10]} f(s_{t}) \geq 0.1) \\
\lor (F_{[0,8]} f(s_{t}) \geq 0.8 \land F_{[8,10]} f(s_{t}) \geq 0.2) \\
\vdots \quad (19)
\]

Remark 2: Use of integral predicate \( \mu_{[a,b]}^i \) provides the flexibility of asynchronous achievement of the tasks. For the cases that require synchronicity, we can simply use \( \mu_{[a,b]}^i \) which would imply the synchronous satisfaction of the task in the specified time \( t = a \). In other words, if the synchronous presence of a
certain number of agents with specific types or instantaneous achievement of a given task are needed in the time step \( a \), we can use \( \mu^t_{[a,b]} \) together with other timed operators instead of the conventional STL predicates.

By combining agent types with previously given examples of tasks such as \( C = \{ \text{Drone}, \text{Ground Vehicle}, \text{Take picture}, \text{Measure temperature}, \text{Collect sample} \} \), and designate \( \operatorname{cap}_{w}^{\mu} = 1 \) for all \( w \in W \), we can define type-specific missions as well. For instance, assume that we want 5 drones to be in region B together at the current time step, then the specification could be written as

\[
F = \bigcap_{l=1}^{5} \{ A, \text{Take Picture}, 10 \}.
\]

D. MILP Encoding

We can now solve Problem 1 subject to cumulative success requirements which is defined in the form of STL constraints via the integral predicates. Although a positive robustness degree (Eqs. (5) and (15)) enforces the satisfaction, it is computed via recursive definitions of computationally expensive \( \min \) and \( \max \) functions. Therefore, using it as a constraint in the optimization or feasibility problems makes the problem non-convex and non-trivial. An alternative way to represent the constraint of satisfying a specification, \( \Phi \), is encoding it as a set of constraints with binary variables in the form of \( z^\phi(k) \in \{0, 1\} \) [32], [17]. For each predicate in the form of inequality, there exists a couple of big M constraints depending on the binary variables as follows:

\[
\mu = g(s) \geq e \left\{ \begin{array}{ll}
g(s) - e \geq M(z^\mu - 1), & \\
g(s) - e \leq Mz^\mu, & 
\end{array} \right.
\]

where \( M \in \mathbb{R}^+ \) is a sufficiently large number. Similar to conventional predicates (20), the satisfaction of an integral predicate is also encoded as:

\[
\phi = \mu^t_{[a,b]} \left\{ \begin{array}{l}
\sum_{r=0}^{t} \frac{b}{\Delta t} f(s_{r+1}) - e \geq M(z^\phi(k) - 1), \\
\sum_{r=0}^{t} \frac{b}{\Delta t} f(s_{r+1}) - e \leq Mz^\phi(k).
\end{array} \right.
\]

The binary constraints corresponding to an STL formula can be built into each other starting from the predicates. The remaining connections of the Boolean operators with other temporal operators and predicates can then be constructed with the following rules and encoded as integer constraints considering two sample formulas \( \phi \) and \( \varphi \) [17].

Negation: \( \phi = \lnot \varphi \)

\[
z^\phi(k) = 1 - z^\varphi(k).
\]

Conjunction: \( \phi = \bigwedge_{i=1}^{m} \varphi_i \)

\[
z^\phi(k) \leq z^\varphi(i), \quad i = 1, \ldots, m,
\]

\[
z^\phi(k) \geq 1 - m + \sum_{i=1}^{m} z^\varphi(i).
\]

Disjunction: \( \phi = \bigvee_{i=1}^{m} \varphi_i \)

\[
z^\phi(k) \leq z^\varphi(i), \quad i = 1, \ldots, m,
\]

\[
z^\phi(k) \geq \sum_{i=1}^{m} z^\varphi(i).
\]

Globally: \( \phi = G[t_1, t_2] \varphi \)

\[
z^\phi(k) = \bigwedge_{\tau=t_1}^{t_2} z^\varphi(k).
\]

Eventually: \( \phi = F[t_1, t_2] \varphi \)

\[
z^\phi(k) = \bigvee_{\tau=t_1}^{t_2} z^\varphi(k).
\]

The satisfaction of an STL formula which is previously implied by \( r(N, \Phi, t) \geq 0 \), now reduces to \( z^\phi(k) = 1 \).

The different agent types with the properties represented in Table I, which will be referred to by their shapes. The pairwise paths between regions of interest for each agent type are found via RRT* sampling-based algorithm avoiding
TABLE I

<table>
<thead>
<tr>
<th>Agent Type (Agent Number)</th>
<th>Capability (per time step)</th>
<th>Regions of interest</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diamond (2)</td>
<td>Take picture (x1)</td>
<td>5</td>
<td>0.25 m/s</td>
</tr>
<tr>
<td>Triangle (2)</td>
<td>Take picture (x1) Measure temperature (x1)</td>
<td>4</td>
<td>0.25 m/s</td>
</tr>
<tr>
<td>Circle (2)</td>
<td>Measure temperature (x2)</td>
<td>4</td>
<td>0.125 m/s</td>
</tr>
</tbody>
</table>

Table 1: Properties of the Three Agent Types

Fig. 1. Team trajectories of diamond, triangle, and circle type agents at selected time instants (dark gray: obstacle for all agents, light gray: obstacles for circle agents only). While the values attached to markers indicate the number of agents from the respective type present at that node, larger circles represent the all regions of interest.

For instance, imaging service in the bottom corner area is started for temperature reading service. Moreover, the implication we specify caused the diamond agents not to service the different corner regions at the same time.

Specifications for the multi-agent system can be summarized as follows: i) One diamond agent and one circle agent continuously and synchronously present at each bridge for any 8 seconds within [10, 38]. ii) Total of 50 temperature readings are made on the service area in the middle within [40, 60]. iii) Top corner area should be imaged 15 times between t = 40 and t = 60. iv) Bottom corner area should be imaged 15 times within [40, 60]. v) Whenever the diamond agents take pictures on both corners at the same time, all the diamond agents need to return to base within the next 10 seconds. Recall that $L_w$ is the number of type w agents. All of these specifications can be combined formally as,

$$\Phi = F_{[10,30]}[G_{[0,8]}(\mu_{[0,0]}^{1}\{Bridge_{top}, Circle, 1\} \land \mu_{[0,0]}^{1}\{Bridge_{top}, Diamond, 1\} \land \{\mu_{[0,0]}^{1}Bridge_{bot.}, Circle, 1\})$$

Simulations are implemented for 70 seconds with $\Delta t = 2$, assuming the presence of two agents of each type, i.e., $L_w = 2$, $\forall w$. Distribution of agents moving over the environment is represented in Fig. 1 for selected time instants. Cumulative specifications defined by integral predicates are successfully accomplished by the contribution of agents with different types. For instance, imaging service in the bottom corner area is started by a diamond agent, then completed by the triangle agents, one of which also joined to the circle agents in the middle for temperature reading service. Moreover, the implication we specify caused the diamond agents not to service the different corner regions at the same time.

To illustrate the scalability of our approach, the same mission is executed by teams of different sizes, and the results are given in Table II. We observe that the increasing number of agents has no significant adverse effect on the size of the problem and the complexity for the same STL specification and number of agent types. The number of total constraints in the optimization problem is 2537 for the original specification and 1876 for the case without implication. These values remain the same with the changing number of agents. In the case of larger environments and more complex specifications with long time horizons, the size of the problem would increase as expected though [17]. It is also clear from the Table II that the implication in (28) has a high impact on the solution time. This is mostly due to the negation operator we used to define the implication.

Although the results imply an inconvenience for real-time application, we consider a high-level, offline planning, and specification of rich tasks with cumulative properties is our main

TABLE II

<table>
<thead>
<tr>
<th>Agent Number</th>
<th>2 + 2 + 2</th>
<th>2 + 5 + 10</th>
<th>10 + 10 + 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Construction Time [s]</td>
<td>0.93</td>
<td>0.94</td>
<td>1.01</td>
</tr>
<tr>
<td>Solver Time [s]</td>
<td>117.72</td>
<td>114.70</td>
<td>154.39</td>
</tr>
<tr>
<td>Problem Construction Time [s] (w/o implication)</td>
<td>0.84</td>
<td>0.83</td>
<td>0.80</td>
</tr>
<tr>
<td>Solver Time [s] (w/o implication)</td>
<td>52.59</td>
<td>64.95</td>
<td>64.76</td>
</tr>
</tbody>
</table>

Table 2: Computation times for different number of agents with three agent types designated as $L_{diamond} + L_{triangle} + L_{circle}$

$$\land \mu_{[0,0]}^{1}\{Bridge_{bot.}, Diamond, 1\})$$

$$\land \mu_{[0,0]}^{1}\{Target_{middle}, Measure temperature, 50\}$$

$$\land \mu_{[0,0]}^{1}\{Target_{top corner}, Take picture, 15\}$$

$$\land \mu_{[0,0]}^{1}\{Target_{bot. corner}, Take picture, 15\}$$

$$\land G_{[0,60]}((\mu_{[0,0]}^{1}\{Target_{top corner}, Diamond, 1\}) \land \mu_{[0,0]}^{1}\{Target_{bot. corner}, Diamond, 1\})$$

$$\implies F_{[0,10]}\{Base, Diamond, L_{diamond}\}. \quad (28)$$
goal. Furthermore, with a limited number of integral predicates and without any implication or negation, we can define similar missions that are much easier to solve.

An experiment is also conducted for the same scenario satisfying (28) for a similar environment via Crazyflie 2.0 platforms. Results of the experiment can be found in the accompanying video.

VI. CONCLUSION

In this study, we introduce integral predicates for STL specifications that can be used in multi-agent systems to encode the cumulative progress in tasks. In the case of heterogeneous agents, such a predicate can be used to define the progress within a specific time interval that can be asynchronously achieved by multiple agents during different time steps. We also show that the integral predicates can be encoded as mixed-integer linear constraints. Optimal trajectories are then obtained by solving a MILP formulation. We show how the proposed predicate can be used in the planning of the multi-agent trajectories in a richer and more expressive way compared to the conventional STL. As a future direction, we plan to focus on the agent assignment and low-level trajectory tracking with collision-avoidance.

REFERENCES


