Synchronous and Asynchronous Multi-Agent Coordination With cLTL+ Constraints

Yunus Emre Sahin, Petter Nilsson, Necmiye Ozay

***Abstract*— Planning trajectories for multiple agents in a way to guarantee that their collective behavior satisfies a certain high-level specification is crucial in many application domains. Motivated by this problem, we introduce a new logic called counting linear temporal logic plus (cLTL+). This logic enables specifying multi-agent tasks over possibly infinite horizons in a compact manner. We then propose an optimization-based method that generates trajectories for individual agents that, when implemented together, guarantee the satisfaction of a given cLTL+ formula. In the second part of the paper, we discuss how these results can be extended to generate trajectories that can be asynchronously implemented by the agents while preserving the satisfaction of the desired cLTL+ specification. In particular, we show that when the asynchrony between agent trajectories is bounded, it is possible to generate trajectories robust against such asynchrony with an appropriate modification of the optimization problem. Finally, we demonstrate these ideas on selected examples.**

I. Introduction

Traditional motion planning algorithms focus on finding a feasible and collision free path that takes a robot from a given initial state to a goal region. More recently, usage of temporal logic to describe more complex high-level specifications such as coverage, formation control and emergency-response has attracted considerable attention [1], [2]. Scalable design methodologies with formal guarantees are needed, in particular in multi-agent settings, as the tasks get more complex with the increasing number of agents.

One approach for synthesizing controllers that satisfy given temporal logic specifications is to recast the problem as a Mixed-Integer Linear Program (MILP) [3], [4]. Using a similar approach, in [5] we introduced a new logic called counting linear temporal logic (cLTL) that is convenient for defining collective behavior for a group of agents and to synthesize controllers that can coordinate large groups. In cLTL synthesis, the agents are assumed to be identical and task completion does not depend on the identity of individual agents, which enables scalability in synthesis algorithms. However, homogeneity and synchrony assumptions in cLTL limit the expressiveness: it is for instance not possible to require that "*the property A must be satisfied by at least 5 agents*" without requiring that A must be satisfied by at least 5 agents *at the same time instant*. For example, in a search and rescue mission one might require each agent to charge its batteries every now and then, but such a requirement can not be expressed in cLTL.

This work is supported in part by NSF grants CNS-1239037, CNS-1446298 and ECCS-1553873, and DARPA grant N66001-14-1-4045. The authors are with the EECS Department, University of Michigan, Ann Arbor, MI 48109, email: {ysahin,pettni,necmiye}@umich.edu.

To overcome this problem, in this paper we extend the syntax of cLTL and propose a new logic, namely cLTL+. In addition to introducing this new logic, our main contributions include an optimization based framework to synthesize controllers from cLTL+ specifications to coordinate multiple agents (i) with possibly *heterogenous* dynamics and (ii) that move potentially *asynchronously*. One way of dealing with asynchrony is to prioritize agents and to generate plans sequentially [6]. However, such an approach depends highly on the order of agents and feasible solutions can be missed if priorities are not assigned correctly. In this paper, we take a different approach and formally define robust satisfaction of a cLTL+ formula. We then present necessary modifications to our synthesis algorithm to generate solutions that are robust against bounded asynchrony.

Notation: The set of nonnegative integers is denoted by $\mathbb{N}$ and $[N] = \{1, 2, \ldots, N\}$ denotes the set of positive integers up to N. The vector of all 1's is denoted by $\mathbf{1}$. For a sequence $\nu = \nu_0\nu_1 \ldots$ of N-tuples, $\nu_t(k)$, $k \in [N]$, denotes the k^{th} element of the t^{th} tuple. We define an indicator function for set membership such that given a set A, $\mathbb{1}_A(a) = 1$ if $a \in A$ and $\mathbb{1}_A(a) = 0$ otherwise. The cardinality of a set A is denoted by $\#A$.

II. System and behavior descriptions

We model each agent's dynamics with a transition system.

Definition 1: A *transition system* is a tuple $T = (S, \to, AP, L)$ where S is a finite set of states, $\to \subseteq S \times S$ is a transition relation, AP is a finite set of atomic propositions and $L : S \to 2^{AP}$ is a labeling function.

Transition systems used in this paper are assumed to be non-blocking, i.e., for all $s \in S$, there exists at least one state $s' \in S$ such that $(s, s') \in \to$. The labeling function L maps a state into a set of atomic propositions that hold in that state. The labeling function is defined such that $a \in AP$ holds in state $s \in S$ if and only if $a \in L(s)$, and we say s satisfies a in this case.

Definition 2: Given $T = (S, \to, AP, L)$, a transition system for an individual agent, an infinite sequence $\pi : s_0 s_1 s_2 \ldots \in S^\omega$ of states such that $(s_k, s_{k+1}) \in \to$ is called a *path (trajectory)*. For a given path π, the corresponding *trace* is defined as $trace(\pi) = L(s_0)L(s_1)L(s_2)\ldots \in (2^{AP})^\omega$. We use σ to denote a trace when π is clear from the context.

In the multi-agent setting, we denote the transition system and a path of the n^{th} agent by $T^n = (S^n, \to^n, AP, L^n)$ and π^n, respectively. Note that dynamics and transitions of the agents can be different but they share the same atomic propositions. This is without loss of generality as one can define a global atomic proposition set by taking the union of atomic propositions of individual agents. For simplicity of notation, for collections of agents $\{T^n\}_{n\in[N]}$, paths $\{\pi^n\}_{n\in[N]}$, etc., we will drop $n \in [N]$ and write $\{T^n\}$, $\{\pi^n\}$, when the range of n is clear from the context.

Individual paths alone are not sufficient to reason about the collective behavior of a multi-agent system. The timing of when agents transition from one state to another relative to the other agents is also important. For this reason, we define the following:

Definition 3: Given N agents with dynamics $\{T^n\}$, let π^n denote the path agent n follows. For a collection of paths $\{\pi^n\}$, a sequence $\Pi = S_0S_1S_2 \cdots \in (\prod_n S^n)^\omega$ of N-dimensional state vectors, where $S_t = [s^1_{k^1_t} \dots s^N_{k^N_t}]^T$, is a *collective execution* consistent with $\{\pi^n\}$ if the following inequalities hold for all $n \in [N]$ and for all t:

$$k^n_0 = 0,\ k^n_t \le k^n_{t+1} \le k^n_t + 1 \text{ and } \lim_{t\to\infty} k^n_t = \infty. \quad (1)$$

The corresponding *collective trace* is defined as

$$ctrace(\Pi) = \Sigma_0\Sigma_1\Sigma_2 \cdots \in (2^{AP^N})^\omega,$$

where $\Sigma_t = L(S_t) \doteq [L^1(s^1_{k^1_t}) \dots L^N(s^N_{k^N_t})]^T$.

The first equality in (1) guarantees that agents start moving synchronously while the second inequality guarantees that no state which appears in an individual path is skipped and the order of states in an individual path is preserved. Note that the second inequality allows stuttering, i.e., agents can choose not to move but the third limit guarantees that the agents eventually make progress with respect to the global time t. Since there are no further restrictions, agents can move asynchronously and there are infinitely many collective executions corresponding to a collection of individual paths. The set of all collective executions consistent with $\{\pi^n\}$ is denoted by $cexecs(\{\pi^n\})$. Similarly, the set of all collective traces consistent with $\{\pi^n\}$ is denoted by $ctraces(\{\pi^n\}) \doteq \{ctrace(\Pi) \mid \Pi \in cexecs(\{\pi^n\})\}$. To exemplify, the following is a valid collective execution for $\{\pi^n : s^n_0 s^n_1 s^n_2 \dots\}$:

$$\Pi = \begin{bmatrix} s^1_0 \\ s^2_0 \\ \vdots \\ s^N_0 \end{bmatrix} \begin{bmatrix} s^1_1 \\ s^2_1 \\ \vdots \\ s^N_1 \end{bmatrix} \begin{bmatrix} s^1_2 \\ s^2_2 \\ \vdots \\ s^N_1 \end{bmatrix} \begin{bmatrix} s^1_2 \\ s^2_3 \\ \vdots \\ s^N_1 \end{bmatrix} \begin{bmatrix} s^1_2 \\ s^2_3 \\ \vdots \\ s^N_1 \end{bmatrix} \cdots \in cexecs(\{\pi^n\}).$$

As previously mentioned, Definition 3 allows agents to execute their paths asynchronously. However, unbounded asynchrony would limit the use of a synthesis approach when reasoning about collective behavior. Without run-time communication or interaction, it is impossible to achieve a task where more than one agents are needed to simultaneously satisfy a proposition if there is no bound on asynchrony. Hence we introduce the following definition.

Definition 4: Let $\{\pi^n : s^n_0 s^n_1 s^n_2 \dots\}$ be a collection of agent trajectories and let $\Pi = S_0S_1S_2\dots$ be a member of $cexecs(\{\pi^n\})$, where $S_t = [s^1_{k^1_t} \dots s^N_{k^N_t}]^T$. Then Π is called a *τ-bounded execution* of $\{\pi^n\}$ if

$$\max_{t\in\mathbb{N}, n,m\in[N]} (|k^n_t - k^m_t|) \le \tau.$$

The set of all τ-bounded executions of $\{\pi^n\}$ is denoted by $cexecs_\tau(\{\pi^n\})$.

For the special case when $\tau = 0$, a collective execution $\Pi \in cexecs_0(\{\pi^n\})$ is called a *synchronous* execution of $\{\pi^n\}$. That means, all agents move from their current state to the next state simultaneously when Π is synchronous. An execution $\Pi^* = S^*_0 S^*_1 S^*_2 \dots$ is the *globally synchronous* execution of $\{\pi^n\}$ if $\Pi^* \in cexecs_0(\{\pi^n\})$, and for all $n, m \in [N]$ and for all t, we have $k^n_t = k^m_t = t$, where k^n_t's are indices in $S^*_t = [s^1_{k^1_t} \dots s^N_{k^N_t}]^T$ as before. A collection of agents is called globally synchronous if all of their executions are globally synchronous. Note that in this case there is a one to one mapping between the collection $\{\pi^n\}$ of trajectories, the corresponding collective execution and the corresponding collective trace. (Globally) synchronous traces are defined similarly.

Given a collective execution Π, the local execution $\tilde{\pi}^n$ of agent n in the global time frame is defined using an *extraction function loc_n* such that $\tilde{\pi}^n = loc_n(\Pi) = \tilde{s}^n_0 \tilde{s}^n_1 \tilde{s}^n_2 \dots$, where $\tilde{s}^n_t = \Pi_t(n)$ for all t. We overload the extraction function and use it also with collective traces. That is, given a collective trace $\Sigma = \Sigma_0\Sigma_1 \dots \in (2^{AP^N})^\omega$, the local trace $\tilde{\sigma}^n$ of agent n in the global time frame is defined by $\tilde{\sigma}^n = loc_n(\Sigma) = \tilde{\sigma}^n_0 \tilde{\sigma}^n_1 \tilde{\sigma}^n_2 \dots$, where $\tilde{\sigma}^n_t = \Sigma_t(n)$ for all t.

III. cLTL+: SYNTAX AND SEMANTICS

In this section we introduce a logic called *counting linear temporal logic plus* (cLTL+) that is convenient for specifying desired collective behaviors of multi-agent systems. The logic cLTL+ is a two layer logic similar to censusSTL [7]. The *inner logic* is identical to LTL and is used to describe tasks that can be satisfied by a single agent. For example, tasks such as *"avoid collisions with obstacles"* or *"eventually visit a charging station"* can be described by the inner logic. cLTL+ is then built on top and is used to specify the evolution of the number of agents that needs to satisfy an inner logic formula. Using the earlier examples, we can specify tasks such as *"All agents must avoid collisions with obstacles"* or *"At least five agents should eventually visit a charging station"* using cLTL+. We now provide formal definitions both for the inner and the outer logic.

An inner logic formula over a set AP of atomic propositions can be defined recursively as follows:

$$\phi ::= True \mid ap \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \bigcirc\phi \mid \phi_1\, \mathcal{U}\, \phi_2, \quad (2)$$

where $ap \in AP$ is an atomic proposition and ϕ, ϕ_1 and ϕ_2 are inner logic formulas. The symbols $\neg, \wedge, \bigcirc$ and $\mathcal{U}$ correspond to the logical operators *negation, conjunction* and the temporal operators *next* and *until*, respectively. Other commonly used operators can be derived from these operators, such as *disjunction* ($\phi_1 \vee \phi_2 \doteq \neg(\neg\phi_1 \wedge \neg\phi_2)$), *release* ($\phi_1\, \mathcal{R}\, \phi_2 \doteq \neg(\neg\phi_1\, \mathcal{U}\, \neg\phi_2)$), *eventually* ($\Diamond\phi \doteq True\, \mathcal{U}\, \phi$), *always* ($\Box\phi \doteq \neg(\Diamond\neg\phi)$), etc. We use Φ to denote the set of all inner logic formulas defined according to (2). Note that the inner logic is identical to LTL but for the sake of completeness we present the semantics here.

Let $\sigma \in (2^{AP})^{\omega}$ and let ϕ be an inner logic formula. Satisfaction of ϕ by σ at step t is denoted by $\sigma, t \models \phi$ and is defined as follows:

- $\sigma, t \models True$,
- for any atomic proposition $ap \in AP$, we say $\sigma, t \models ap$ if and only if $ap \in \sigma_t$,
- $\sigma, t \models \varphi_1 \wedge \varphi_2$ if and only if $\sigma, t \models \varphi_1$ and $\sigma, t \models \varphi_2$,
- $\sigma, t \models \neg\varphi$ if and only if $\sigma, t \not\models \varphi$,
- $\sigma, t \models \bigcirc\varphi$ if and only if $\sigma, t+1 \models \varphi$, and
- $\sigma, t \models \varphi_1 \,\mathcal{U}\, \varphi_2$ if and only if there exists $l \geq 0$ such that $\sigma, t+l \models \varphi_2$ and $\sigma, t+l' \models \varphi_1$ for all $0 \leq l' < l$.

If $\sigma, 0 \models \varphi$, then we say that σ satisfies φ and write $\sigma \models \varphi$ for short. We say that a path π satisfies φ if $trace(\pi) \models \varphi$, and write $\pi \models \varphi$.

Having defined the inner logic we can now present the syntax for cLTL+ which is based on a new proposition type: a *temporal counting proposition* tcp is an inner logic formula paired with a nonnegative integer, i.e., $tcp = [\phi, m] \in \Phi \times \mathbb{N}$ where $\phi \in \Phi$ and $m \in \mathbb{N}$.

The following grammar can now be used to recursively define cLTL+ formulas.

$$\mu ::= True \mid tcp \mid \neg\mu \mid \mu_1 \wedge \mu_2 \mid \bigcirc\mu \mid \mu_1 \,\mathcal{U}\, \mu_2, \qquad (3)$$

where $tcp \in \Phi \times \mathbb{N}$ is a temporal counting proposition and μ, μ_1 and μ_2 are cLTL+ formulas. Identical to inner logic, other commonly used operators can be derived from (3).

Formulas in cLTL+ are evaluated over collective traces of a multi-agent system. In order to define the semantics, we first define a counting function.

Definition 5: Given an inner logic formula $\phi \in \Phi$ and a collective trace Σ, the *counting function* $\xi_{\phi,t} : ((2^{AP})^{\omega})^N \to \mathbb{N}$ maps Σ to the number of local traces that satisfy the inner formula ϕ at step t in the global time frame, that is, $\xi_{\phi,t} : \Sigma \mapsto \#\{n \mid loc_n(\Sigma), t \models \phi\}$.

Next we define the semantics for cLTL+, which is almost identical to inner logic. Satisfaction of a cLTL+ formula μ by a collective trace Σ at step t, denoted as $\Sigma, t \models \mu$, is defined as follows:

- $\Sigma, t \models True$,
- for any temporal counting proposition $tcp = [\phi, m] \in \Phi \times \mathbb{N}$, we say $\Sigma, t \models tcp$ if and only if $\xi_{\phi,t}(\Sigma) \geq m$,
- $\Sigma, t \models \mu_1 \wedge \mu_2$ if and only if $\Sigma, t \models \mu_1$ and $\Sigma, t \models \mu_2$,
- $\Sigma, t \models \neg\mu$ if and only if $\Sigma, t \not\models \mu$,
- $\Sigma, t \models \bigcirc\mu$ if and only if $\Sigma, t+1 \models \mu$, and
- $\Sigma, t \models \mu_1 \,\mathcal{U}\, \mu_2$ if and only if there exists $l \geq 0$ such that $\Sigma, t+l \models \mu_2$ and $\Sigma, t+l' \models \mu_1$ for all $0 \leq l' < l$.

If $\Sigma, 0 \models \mu$, then we say that the collective trace Σ satisfies μ and write $\Sigma \models \mu$ for short. We say that a collective execution Π satisfies μ if $ctrace(\Pi) \models \mu$, and write $\Pi \models \mu$.

Example 1: Consider the following set of similar cLTL+ formulas: $\mu_1 \doteq \Box\Diamond[C_s, m]$, $\mu_2 \doteq [\Box\Diamond C_s, m]$, and $\mu_3 \doteq \Box[\Diamond C_s, m]$ for $C_s \in AP$.

In the first formula μ_1—which is also a cLTL formula—the inner formula $[C_s, m]$ requires at least m agents to simultaneously satisfy C_s. Combined with the temporal operators; in order for the cLTL+ formula μ_1 to be satisfied there should be an infinite number of instances where C_s is simultaneously satisfied by more than m agents.

On the other hand, μ_2 relaxes the simultaneity requirement. The inner formula $\Box\Diamond C_s$ requires an agent to satisfy C_s infinitely many times, hence the cLTL+ formula μ_2 is satisfied if the number of agents that satisfy C_s infinitely many times is greater than or equal to m. Note that the agents can satisfy C_s at different times—the number of agents that satisfy C_s might never exceed 1 for any given time and μ_2 can still be satisfied. This implies that any collective path that satisfies μ_1 also satisfies μ_2 but the converse is not true.

The difference between μ_3 and μ_2 is more subtle: when the number of agents are finite they are equivalent, meaning that any collective path that satisfies one will also satisfy the other. However, in the hypothetical case where the number of agents is infinite [1], μ_3 can be satisfied even if no agent satisfies C_s more than once. This implies that solutions to μ_2 will also satisfy μ_3 but the converse is not true. $\Box$

Remark 1: As a comparison, cLTL corresponds to the fragment of cLTL+ where the inner logic is constrained to the grammar $\phi ::= ap$.

IV. Synchronous coordination problem and its solution

Now we are in a position to formally pose the first multi-agent coordination problem we are interested in; which is in the synchronous setting. After the problem statement, we propose a solution.

Problem 1: Given N agents with dynamics $\{T^n = (S^n, \to^n, AP, L^n)\}$, initial conditions $\{s_0^n\}$, and a cLTL+ formula μ over AP, synthesize individual paths π^n starting at s_0^n for each agent n such that the globally synchronous collective execution Π^* corresponding to the collection $\{\pi^n\}$ satisfies μ, i.e., $\Pi^* \models \mu$.

In order to solve Problem 1, we encode the agent dynamics and the cLTL+ constraints using integer linear constraints and pose the synthesis problem as an integer linear program (ILP). This is similar to the approach adopted in [5] for cLTL, and inspired by the bounded model-checking literature [8]. In particular, we focus the search to individual paths in prefix-suffix form. That is, for a given integer h, we aim to construct individual paths of the form $\pi^n = s_0^n s_1^n \ldots s_h^n \ldots$ and find an integer $l \in \{0, \ldots, h-1\}$ such that for all $k \geq h$, $s_k^n = s_{k+l-h}^n$.

A. Globally synchronous agent dynamics

Given the transition system $T^n = (S^n, \to^n, AP, L^n)$ that represents the dynamics of agent n, consider the adjacency matrix A^n corresponding to the transition relation $\to^n$. We use a Boolean vector $w_t^n \in \{0,1\}^{\#S^n}$ with a single nonzero component to denote the state of agent n at time t. For example, assume $S^n = \{v_1^n, v_2^n, v_3^n\}$ and that agent n is at v_2^n at time t. Then, $w_t^n = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$. With slight abuse of notation, we equivalently write, $w_t^n(v_2^n) = 1$ and $w_t^n(v_i^n) = 0$ for $i \neq 2$.

[1]The rest of the paper assumes that the number of agents is finite.

Given s_0^n, the following constraints encode the transitions of individual agents when they execute globally synchronously:

$$\begin{aligned} w_{t+1}^n &\le A^n w_t^n, \\ w_0^n(s_0^n) &= 1, \mathbf{1}^T w_t^n = 1, \end{aligned} \tag{4}$$

for all $n \in [N]$ and for all $t \in \{0, \dots, h-1\}$. The path π^n corresponding to a sequence $w^n = w_0^n w_1^n \dots$ can then be extracted by locating the nonzero component in each w_t^n. When paths of all agents are combined, there is a one to one correspondence between $\{w^n\}$ and Π^*—the globally synchronous collective execution corresponding to $\{\pi^n\}$.

B. Loop constraints

In order to impose prefix-suffix form on the generated paths, we introduce h binary variables $\mathbf{z^{loop}} = \{z_0^{loop}, \dots z_{h-1}^{loop}\}$ and the following constraints:

$$w_t^n - \mathbf{1}(1 - z_t^{loop}) \le w_h^n \le w_t^n + \mathbf{1}(1 - z_t^{loop}), \tag{5}$$

for all $n \in [N]$ and for all $t \in \{0, \dots, h-1\}$, together with $\sum_{t=0}^{h-1} z_t^{loop} = 1$.

C. Inner logic constraints

Let $\phi \in \Phi$ be an inner logic formula given according to (2) and h be the horizon length. For each agent n, we generate h binary decision variables $z_t^{\phi,n} \in \{0, 1\}$ for $t \in \{0, 1, \dots, h-1\}$ and ILP constraints such that $z_t^{\phi,n} = 1$ if and only if $trace(loc_n(\Pi^*)), t \models \phi$. Hence, satisfaction of an inner formula ϕ by the agent n is equivalent to $z_0^{\phi,n}$ being 1. We use the following encodings in a recursive manner to create the corresponding ILP constraints.

ap (atomic proposition): Let $\phi = ap \in AP$ be an atomic proposition and the states of T^n be given by the set $S^n = \{v_1^n, v_2^n, \dots, v_{\#S^n}^n\}$. We define the vector $\mathbf{v}_\phi^n \in \{0, 1\}^{\#S^n}$ such that $\mathbf{v}_\phi^n(i) = 1$ if and only if $\phi \in L(v_i^n)$. That is, $\mathbf{v}^n$ encodes the labeling function L^n. Then we introduce the following constraints for all $n \in [N]$:

$$z_t^{\phi,n} < (\mathbf{v}_\phi^n)^T w_t^n < z_t^{\phi,n} + 1. \tag{6}$$

$\neg$ *(negation):* Let $\phi = \neg\varphi$. Then for all $n \in [N]$,

$$z_t^{\phi,n} = 1 - z_t^{\varphi,n}, \qquad t = 0, \dots, h-1. \tag{7}$$

$\wedge$ *(conjunction):* Let $\phi = \bigwedge_{i=1}^{I} \varphi_i$. Then for all $t = 0, \dots, h-1$ and for all $n \in [N]$,

$$\begin{aligned} z_t^{\phi,n} &\le z_t^{\varphi_i,n}, \qquad \text{for } i = 1, \dots, I \quad \text{and,} \\ z_t^{\phi,n} &\ge 1 - I + \sum_{i=1}^{I} z_t^{\varphi_i,n}. \end{aligned} \tag{8}$$

With slight abuse of notation, we also use Boolean operators on these optimization variables. For example, for $\phi = \bigwedge_{i=1}^{I} \varphi_i$, we say $z^{\phi,n} = \bigwedge_{i=1}^{I} z^{\varphi_i,n}$ instead of stating the inequalities in (8). Encoding of the temporal operators is as follows: $\bigcirc$ *(next):* Let $\phi = \bigcirc\varphi$, then for all $n \in [N]$

$$\begin{aligned} z_t^{\phi,n} &= z_{t+1}^{\varphi,n}, \qquad t = 0, \dots, h-2 \text{ and,} \\ z_{h-1}^{\phi,n} &= \bigvee_{t=0}^{h-1} (z_t^{\varphi,n} \wedge z_t^{loop}). \end{aligned} \tag{9}$$

$\mathcal{U}$ *(until):* if $\phi = \varphi_1 \, \mathcal{U} \, \varphi_2$, then for all $n \in [N]$

$$\begin{aligned} z_t^{\phi,n} &= z_t^{\varphi_2,n} \vee \left(z_t^{\varphi_1,n} \wedge z_{t+1}^{\phi,n} \right), \quad t \le h-2, \\ z_{h-1}^{\phi,n} &= z_{h-1}^{\varphi_2,n} \vee \left(z_{h-1}^{\varphi_1,n} \wedge \left(\bigvee_{t=0}^{h-1} \left(z_t^{loop} \wedge \tilde{z}_t^{\phi,n} \right) \right) \right), \\ \tilde{z}_t^{\phi,n} &= z_t^{\varphi_2,n} \vee \left(z_t^{\varphi_1,n} \wedge \tilde{z}_{t+1}^{\phi,n} \right), \quad t \le h-2, \\ \tilde{z}_{h-1}^{\phi,n} &= z_{h-1}^{\varphi_2,n}, \end{aligned} \tag{10}$$

where $\tilde{z}_t^{\phi,n}$ are auxiliary binary variables. As shown in [8], not introducing auxiliary variables results in trivial satisfaction of the until formula ϕ.

D. Outer logic constraints

Similar to the inner logic we proceed by transforming a cLTL+ formula into ILP constraints. Given a cLTL+ formula μ and a time horizon h, we create h binary decision variables $\mathbf{y^{cLTL+}} = \{y_t^\mu\}$ where $t \in \{0, 1, \dots, h-1\}$ and ILP constraints $ILP(\mu)$ such that $y_t^\mu = 1$ if and only if $\Sigma, t \models \mu$ where Σ is a synchronous collective trace. We remind the reader that since ILP constraints are created recursively, creating the constraints for the formula μ will create the constraints for all the inner logic formulas appearing in μ. We denote by $ILP(\mu)$ the set of all resulting constraints that encode satisfaction of μ, and by $(\mathbf{z}, \mathbf{y})^{\mathbf{cLTL+}}$ the set of all variables created in this process.

We only provide encodings for counting propositions since the rest of the semantics are identical. Let $\mu = [\phi, m] \in AP \times \mathbb{N}$ be a counting proposition. Then

$$m - M \le \sum_{n=1}^{N} z_t^{\phi,n} - M y_t^\mu < m, \tag{11}$$

where M is a sufficiently large positive number, in particular, $M > N+1$. Note that when $y_t^\mu = 1$; the inequality on the left reduces to $m \le \sum_{n=1}^{N} z_t^{\phi,n}$, and the inequality on the right is trivially satisfied. Conversely, when $y_t^\mu = 0$ the inequality on the left is trivially satisfied and the right inequality reduces to $\sum_{n=1}^{N} z_t^{\phi,n} < m$. This implies that $y_t^\mu = 1$ ($y_t^\mu = 0$) if and only if the number of agents that satisfy ϕ at time t is greater than or equal to (less than) m. Therefore the ILP constraints in (11) are correct and consistent with the semantics of cLTL+.

E. Overall optimization problem and its analysis

A solution to an instance of Problem 1 for a horizon length h is generated via the following optimization problem:

$$\begin{aligned} \text{Find} \quad & w_{1:h}^{1:N}, \mathbf{z^{loop}}, (\mathbf{z}, \mathbf{y})^{\mathbf{cLTL+}} \\ \text{s.t.} \quad & (4), (5), ILP(\mu) \text{ and } y_0^\mu = 1. \end{aligned} \tag{12}$$

Next we analyze this solution approach.

Proposition 1: Given an instance of Problem 1, if (12) is feasible, paths generated from $w_{1:h}^{1:N}$ solves Problem 1.

Proof: The constraint (4) guarantees that individual trajectories $\{\pi^n\}$ generated from $\{w_{0:h}^n\}$ are feasible. Furthermore, (5) makes sure that these solutions can be extended to infinite trajectories of the form $\pi^n =$

$s_0^n s_1^n \dots s_{l-1}^n \left(s_l^n \dots s_{h-1}^n\right)^\omega$. The ILP encodings (6)-(10) of LTL formulas are sound [8] and the same encodings are also used for cLTL+ formulas by replacing $z_t^{\phi,n}$ with z_t^μ where μ is any cLTL+ formula. The only exception is that (6) is replaced with (3) which we showed to be correct. This implies that, enforcing the constraint $z_0^\mu = 1$ together with $ILP(\mu)$ guarantees that $\Sigma \models \mu$ where $\Sigma \in ctraces(\{\pi^n\})$ is globally synchronous. Hence, if (12) is feasible, the globally synchronous execution of $\{\pi^n\}$ solves Problem 1. ■

As a corollary, it is easy to show that stutter invariance of formulas (theorem 7.92 in [9]) allows to generalize the soundness result from globally synchronous executions to synchronous executions:

Corollary 1: If μ does not contain any next operator $\bigcirc$, neither in the inner nor in the outer logic, then $\Sigma \models \mu$ for all synchronous $\Sigma \in ctraces(\{\pi^n\})$.

Proposition 2: If there is a solution to Problem 1, then there exists a finite h such that (12) is feasible.

Proof: In order to show that prefix-suffix form solutions are complete, we reduce Problem 1 to a regular LTL control synthesis problem, for which prefix-suffix solutions have been shown to be complete [9].

Let Φ be the set of all inner logic formulas defined according to (2) over AP. Given any cLTL+ formula μ, one can define an equivalent LTL formula over a new set of atomic propositions $AP' = \bigcup_{a \in AP}\{a_1, a_2, \dots a_N\}$. For each temporal counting proposition $tcp = [\phi, m]$ in μ, we define a new set $\{\phi_1, \phi_2, \dots \phi_N\}$ of LTL formulas over AP' where ϕ_n is obtained by replacing every atomic proposition $a \in AP$ with the corresponding $a_n \in AP'$. Then we can define $tcp' \doteq \bigvee_{i=1}^{I}(\bigwedge_{j \in J_i} \phi_j)$ that is equivalent to tcp such that $J_i \in J$ where J is the set of all m-element subsets of $[N]$ and $I = \#J$. Even though this method increases the number of atomic propositions linearly and the length of the formula combinatorially with the number of agents, it could transform cLTL+ formula into a regular LTL formula over a finite set of atomic propositions.

Next we create a product transition system $T' \doteq \Pi_n T_n$ with the set AP' as its atomic propositions. Now that Problem 1 is reduced to a standard LTL synthesis problem, it can be solved using a model-checker to generate a prefix-suffix solution or to declare non-existence of solutions (see e.g., [10]). ■

Remark 2: The proof of Proposition 2 highlights the advantages of using cLTL+ in scenarios where agent identity is not critical to accomplish the collective task. Although the problem can be reduced to a standard LTL synthesis problem as the proof suggests, the reduction results in a synthesis problem on a product transition system with size exponential in the number of agents, and with an LTL formula that is combinatorially longer than the cLTL+ formula. Indeed, without a convenient logic, even writing that LTL formula would be a very tedious and error-prone task.

V. Robustness to asynchrony

We continue with incorporating the concept of time-robustness into our algorithm. Small modification to the original algorithm allow us to synthesize trajectories that are robust to bounded asynchrony.

The notion of robustness is useful since it is difficult to perfectly synchronize the motion of agents in real life applications. To exemplify, consider a task that requires more than a single agent to simultaneously satisfy a certain proposition. Let $\mu = \Diamond tcp$ for $tcp = [ap, m] \in AP \times \mathbb{N}$ and assume that $\Pi \in cexecs_0(\{\pi^n\})$ is a synchronous collective trajectory with trace Σ in which tcp fails to hold for all time steps except for a single step k, i.e., $\Sigma, t \not\models tcp$ for all $t \neq k$ and $\Sigma, k \models tcp$. Note that Π is a solution to Problem 1 but not very desirable: if for some reason one of the agents gets delayed and Π becomes asynchronous, correctness guarantees will no longer be valid.

In the globally synchronous case the time scales of individual agent trajectories and the collective execution coincide (up to stuttering), but with asynchrony this is no longer the case. In the asynchronous case we will interpret formulas over the *anchor time*, defined as follows:

Definition 6: Let $\Pi \in cexecs(\{\pi^n\})$ be a collective execution. The associated *anchor time mapping* is $b_\Pi(t) \doteq \min_n k_t^n$, where k_t^n is the local time of agent n at the t'th step of the collective execution.

For $\Pi \in cexecs_\tau(\{\pi^n\})$ and a given global time t, at least one agent has local time $b_\Pi(t)$ and all other agents have local times in the interval $[b_\Pi(t), b_\Pi(t) + \tau]$, for $b_\Pi(t) + \tau \leq t$. For the globally synchronous collective execution Π^*, the anchor time mapping is the identity mapping on $\mathbb{N}$. To further illustrate the concept, $b_\Pi^{-1}(0) = \{0, \dots, t_1 - 1\}$, where $t_1 = \min\{t : k_t^n \geq 1 \forall n \in [N]\}$ is the first time in the time scale of Π when all individual agents have made at least one move. Below is an illustration of the anchor times and the "anchoring agents" (in bold) for a collective 2-bounded execution Π constructed from four agent trajectories:

$$\Pi = \begin{array}{cccccc} \begin{bmatrix} \mathbf{s_0^1} \\ \mathbf{s_0^2} \\ \mathbf{s_0^3} \\ \mathbf{s_0^4} \end{bmatrix} & \begin{bmatrix} \mathbf{s_0^1} \\ s_1^2 \\ s_1^3 \\ s_1^4 \end{bmatrix} & \begin{bmatrix} \mathbf{s_0^1} \\ s_1^2 \\ s_2^3 \\ s_2^4 \end{bmatrix} & \begin{bmatrix} \mathbf{s_1^1} \\ s_2^2 \\ s_3^3 \\ s_3^4 \end{bmatrix} & \begin{bmatrix} \mathbf{s_2^1} \\ \mathbf{s_2^2} \\ s_4^3 \\ s_4^4 \end{bmatrix} & \begin{bmatrix} s_3^1 \\ \mathbf{s_2^2} \\ s_4^3 \\ s_4^4 \end{bmatrix} \\ {\scriptstyle b_\Pi(0)=0} & {\scriptstyle b_\Pi(1)=0} & {\scriptstyle b_\Pi(2)=0} & {\scriptstyle b_\Pi(3)=1} & {\scriptstyle b_\Pi(4)=2} & {\scriptstyle b_\Pi(5)=2} \end{array} \dots \tag{13}$$

We can now define the concept of robust satisfaction for a collection of agent trajectories.

Definition 7: A collection of agent trajectories $\{\pi^n\}$ τ-*robustly satisfies* μ at time T, denoted

$$\{\pi^n\}, T \models_\tau \mu, \tag{14}$$

if and only if for all $\Pi \in cexecs_\tau(\{\pi^n\})$,

$$\forall\, t \in b_\Pi^{-1}(T);\ \Pi, t \models \mu. \tag{15}$$

In other words, a specification μ is τ-robustly satisfied at time T by $\{\pi^n\}$ if every collective execution $\Pi \in cexecs_\tau(\{\pi^n\})$ satisfies μ at all time instances t for which the anchor time is T. Consider for example a specification $\{\pi^1, \dots, \pi^4\}, 0 \models_2 \mu$; then for Π in (13) above we must have $\Pi, 0 \models \mu$, $\Pi, 1 \models \mu$, and $\Pi, 2 \models \mu$ since $b_\Pi^{-1}(0) = \{0, 1, 2\}$. In addition the same must hold for every possible $\Pi \in cexecs_\tau(\{\pi^1, \dots, \pi^4\})$.

We propose slight modifications to the encodings presented in Section IV in order to generate agent trajectories that are τ-robust. First, we require the cLTL+ formula to be given according to the following syntax:

$$\mu ::= True \mid tcp \mid \bigcirc\mu \mid \mu_1 \wedge \mu_2 \mid \mu_1 \vee \mu_2 \\ \mid \mu_1 \,\mathcal{U}\, \mu_2 \mid \mu_1 \,\mathcal{R}\, \mu_2. \quad (16)$$

Remark 3: The negation operator can be omitted without loss of generality because of two reasons. First, we remind the reader that any LTL formula can be transformed into positive normal form (PNF) [9] where the negation operator appears only before atomic propositions. Since the syntax of cLTL+ is identical to LTL, any cLTL+ formula can also be written in PNF, where negation only appears before *tcp*'s. An arbitrary $\mu = \neg[\phi, m]$ for $\phi \in \Phi$ and $m \in \mathbb{N}$ can equivalently be written as $\mu' = [\neg\phi, N+1-m]$. Clearly, if there are at least $N+1-m$ robots satisfying $\neg\phi$, then ϕ is satisfied by less than m robots, hence $\mu \equiv \mu'$.

We further limit the inner logic to $\text{LTL}_{\setminus\{\bigcirc\}}$ to remove ambiguity resulting from asynchrony. In the (outer) cLTL+ logic '$\bigcirc$' is by Definition 7 interpreted over *anchor time*: $\bigcirc\phi$ holds at anchor time T if ϕ holds when the slowest agent(s) has made $T+1$ time steps.

To help us define the robust encodings, for each n we define additional state vectors w^n_{h+k} for $k \in [\tau]$ and $t \geq k$:

$$w^n_t - \mathbf{1}(1 - z^{loop}_{t-k}) \leq w^n_{h+k} \leq w^n_t + \mathbf{1}(1 - z^{loop}_{t-k}), \quad (17)$$

and use these to define the $z^{\phi,n}_{h+k}$s for $k \in [\tau]$ via (6). Equipped with these additional variables we introduce robust versions $r^{\phi,n}_t$ of the inner binary variables $z^{\phi,n}_t$. Let $\mu = [\phi, m]$ for $\phi \in \Phi$, then

$$r^{\phi,n}_t = \bigwedge_{k=0}^{\tau} z^{\phi,n}_{t+k}, \qquad \text{for } 0 \leq t \leq h. \quad (18)$$

In order to generate robust solutions, we modify the encoding of an outer temporal counting proposition $\mu = [\phi, m]$ as a binary variable y^μ_t given in (11) as follows:

$$m - M \leq \sum_{n=1}^{N} r^{\phi,n}_t - M y^\mu_t < m. \quad (19)$$

Robustification ensures that a temporal counting proposition which would normally be satisfied only at a synchronous instant, is now also satisfied for all possible asynchronous states that are τ-bounded. This way, even in the worst case scenario where the maximum bound on asynchrony is achieved, the corresponding proposition will be satisfied.

Higher level outer binary variables y^μ_t for $\wedge$, $\vee$, $\bigcirc$ and $\mathcal{U}$ are defined with the standard encodings from before, and "release" is encoded with the following standard encoding: if $\phi = \mu_1 \,\mathcal{R}\, \mu_2$, then

$$y^\mu_t = y^{\mu_2}_t \wedge \left(y^{\mu_1}_t \vee y^\mu_{t+1}\right), \quad t \leq h-2,$$
$$y^\mu_{h-1} = y^{\mu_2}_{h-1} \wedge \left(y^{\mu_1}_{h-1} \vee \left(\bigvee_{t=0}^{h-1} \left(y^{loop}_t \wedge \tilde{y}^\mu_t\right)\right)\right),$$
$$\tilde{y}^\mu_t = y^{\mu_2}_t \wedge \left(y^{\mu_1}_t \vee \tilde{y}^\mu_{t+1}\right), \quad t \leq h-2,$$
$$\tilde{y}^\mu_{h-1} = y^{\phi_2}_{h-1}.$$

Proposition 3: The robustification (18) - (19) is sound with respect to the robust synthesis problem for a cLTL+ specification μ given on positive normal form. That is, if the resulting ILP is feasible a collection of agent trajectories $\{\pi^n\}$ can be extracted such that $\{\pi^n\}, 0 \models_\tau \mu$.

The proof of Proposition 3 relies on reasoning over quantifiers; for convenience we introduce some additional notation. There is an infinite number of τ-bounded executions but each execution is uniquely defined by the local time series $\{k^n_t\}$, $n \in [N]$, $t \geq 0$. Given an anchor time T, the set of possible local times is

$$\mathcal{T}_\tau(T) = \left\{ (T+\delta_1, \ldots, T+\delta_N) \;\middle|\; \begin{array}{l} 0 \leq \delta_n \leq \tau \;\forall n; \\ \exists n \; s.t. \; \delta_n = 0 \end{array} \right\}.$$

We can then define the set of all collective execution suffixes beyond an anchor time T as

$$\mathfrak{C}_\tau(T, \{\pi^n\}) = \left\{ S_0 S_1, \ldots \;\middle|\; \begin{array}{l} S_0 = [s^1_{k^1_0}, \ldots, s^N_{k^N_0}]^T, \\ (k^1_0, \ldots, k^N_0) \in \mathcal{T}_\tau(T) \end{array} \right\}.$$

The following lemma is now merely a restatement of Definition 7 which is convenient for conversion between robust satisfaction and a quantified traditional satisfaction relation.

Lemma 1: For any $T' \leq T$, $\{\pi^n\}, T \models_\tau \mu$ if and only if

$$\forall\, \tilde{\Pi} \in \mathfrak{C}_\tau(T - T', \{\pi^n\}); \;\; \forall t' \in b^{-1}_{\tilde{\Pi}}(T'); \;\; \tilde{\Pi}, t' \models \mu.$$

We point out the special case

$$\{\pi^n\}, T \models_\tau \mu \iff \\ \forall \Pi \in \mathfrak{C}_\tau(T, \{\pi^n\}); \; \forall t' \in b^{-1}_{\Pi}(0); \; \Pi, t' \models \mu,$$

which essentially means that ensuring robust satisfaction at (anchor) time T is equivalent to ensuring satisfaction at times with anchor time 0 for all suffixes of paths $\Pi \in ctrace_\tau(\{\pi^n\})$ that start at an anchor time T.

Proof: [Proposition 3] We first note that (18) results in binary variables $r^{\phi,n}_t$ that are equal to 1 if and only if $\pi^n, t+\delta \models \phi$ for all $\delta \in [0, \tau]$.

By the restriction to PNF it is enough to prove soundness for the operators in (16), we do so recursively starting with a temporal counting proposition $\mu = [\phi, m] \in \Phi \times \mathbb{N}$. By Lemma 1, $\{\pi^n\}, T \models_\tau \mu$ if and only if $\sum_{n=1}^N \mathbb{1}_\phi(L(s^n_{k^n_0})) \geq m$, for all $\{k^n_0\} \in \mathcal{T}_\tau(T)$. From (19), we have that $y^\mu_T = 1$ if and only if

$$m \leq \sum_{n=1}^{N} \min_{0 \leq \Delta_n \leq \tau} z^{\phi,n}_{T+\Delta_n} = \sum_{n=1}^{N} \min_{0 \leq \Delta_n \leq \tau} \mathbb{1}_\phi(L(s^n_{T+\Delta_n})) \\ = \min_{\{k^n_0\} \in \mathcal{T}_\tau(T)} \sum_{n=1}^{N} \mathbb{1}_\phi(L(s^n_{k^n_0})),$$

where the last equality follows from an argument comparing minimizers. Thus $y^\mu_T = 1$ if and only if $\{\pi^n\}, T \models_\tau \mu$.

The rest of the proof consists in applying Lemma 1 and manipulating quantifiers to show that robust satisfaction is propagated soundly for the LTL operators $\wedge, \bigcirc, \vee, \mathcal{U}, \mathcal{R}$:

1) $\{\pi^n\}, T \models_\tau \mu_1$ and $\{\pi^n\}, T \models_\tau \mu_2$ if and only if $\{\pi^n\}, T \models_\tau \mu_1 \wedge \mu_2$,
2) $\{\pi^n\}, T+1 \models_\tau \mu$ if and only if $\{\pi^n\}, T \models_\tau \bigcirc\mu$,

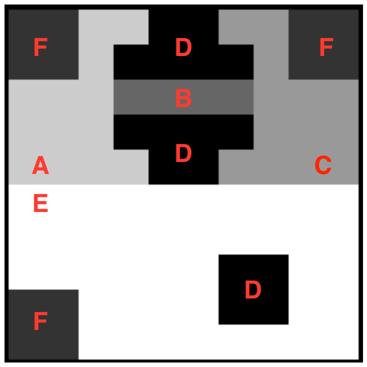

Fig. 1. Workspace: A, C, and E represent different neighborhoods, B represents a fragile bridge, F represents charging stations and D represents inaccessible zones.

3) If $(\{\pi^n\}, T \models_\tau \mu_1) \vee (\{\pi^n\}, T \models_\tau \mu_2)$, then it follows that $\{\pi^n\}, T \models_\tau \mu_1 \vee \mu_2$,
4) There exists $l \geq 0$ such that $\{\pi^n\}, T + l \models_\tau \mu_2$ and $\{\pi^n\}, T + l' \models_\tau \mu_1$ for $0 \leq l' < l$, implies that $\{\pi^n\}, T \models_\tau \mu_1 \,\mathcal{U}\, \mu_2$,
5) If there exists $l \geq 0$ such that $(\{\pi^n\}, T + l \models_\tau \mu_1) \wedge (\forall 0 \leq l' \leq l;\ \{\pi^n\}, T + l' \models_\tau \mu_2)$ OR for all $l \geq 0$ $T + l \models_\tau \mu_2$, then it holds that $\{\pi^n\}, T \models_\tau \mu_1 \,\mathcal{R}\, \mu_2$.

Note that $\wedge$ and $\bigcirc$ are sound and complete, whereas $\vee$, $\mathcal{U}$, and $\mathcal{R}$ are merely sound.

For space reasons we omit the last four and only show that encoding of $\wedge$ in (8) is sound and complete:

$$\begin{aligned}
&(\{\pi^n\}, T \models_\tau \mu_1) \wedge (\{\pi^n\}, T \models_\tau \mu_2)\,,\\
\iff &\big(\forall \Pi \in \mathfrak{C}_\tau(0, \{\pi^n\});\ \forall t \in b_\Pi^{-1}(T);\ \ \Pi, t \models \mu_1\big)\\
&\wedge \big(\forall \Pi \in \mathfrak{C}_\tau(0, \{\pi^n\});\ \forall t \in b_\Pi^{-1}(T);\ \ \Pi, t \models \mu_2\big)\,,\\
\iff &\forall \Pi \in \mathfrak{C}_\tau(0, \{\pi^n\});\ \forall t \in b_\Pi^{-1}(T);\\
&\qquad (\Pi, t \models \mu_1) \wedge (\Pi, t \models \mu_2)\,,\\
\iff &\forall \Pi \in \mathfrak{C}_\tau(0, \{\pi^n\});\ \forall t \in b_\Pi^{-1}(T);\ (\Pi, t \models \mu_1 \wedge \mu_2)\,,\\
\iff &(\{\pi^n\}, T \models_\tau \mu_1 \wedge \mu_2)\,.
\end{aligned}$$

Due to these relations soundness is propagated through the PNF LTL formula from the level of temporal counting proposition which we showed was sound, thus the soundness of the whole encoding procedure follows as before from soundness of ILP encodings of LTL. ■

VI. Results

We revisit the emergency response example in [5], where 10 robots are to be deployed in an environment depicted by Fig.1. The area marked A represents the only region where the atomic proposition a holds, and similarly for the regions B, C, D, E, and F. We discretize the workspace into 10×10 identical cells and model each robot with a transition system with 100 states corresponding to these cells. Robots can either choose to travel to any of the four neighboring cells without leaving the workspace, or stay put. Note that a monolithic LTL solution for this problem will require constructing a transition system with 100^{10} states.

The specification is of the form $\mu = \bigwedge_{i=1}^{7} \mu_i$, including:

- collision with obstacles, which are marked with D, should be avoided ($\mu_1 = \Box(\neg[d, 1])$).
- the bridge, marked by B, must never be occupied by more than 2 robots ($\mu_2 = \Box(\neg[b, 3])$).
- each robot should visit charging stations, marked by F, infinitely many times ($\mu_3 = [\Box\Diamond f, 10]$).
- regions A and C must be populated with more than 5 robots and should be left empty, infinitely many times ($\mu_4 = \Box\Diamond[a, 5]$, $\mu_5 = \Box\Diamond[c, 5]$, $\mu_6 = \Box\Diamond(\neg[a, 1])$ and $\mu_7 = \Box\Diamond(\neg[c, 1])$).

We posit a time horizon $h = 30$ and the resulting optimization problem is solved in 362 seconds on a laptop with Intel i7 processor and 16 GB RAM, and our implementation can be found at `https://github.com/sahiny/cLTL-synth`. Important frames from the obtained solution that illustrate that the specifications are met are shown in Fig. 2. Note that region C is populated with more than five robots only for a single time step at $t = 14$. Furthermore, even though the bridge is never occupied by more than two robots, at time $t = 13$, one new robot (olive) enters the bridge as one of the two robots (navy blue) present on the bridge exits. Even though this solution is valid for a synchronous execution, it can easily break with the introduction of asynchrony. A single step delay of one of the agents could result in violation of the aforementioned specifications.

To overcome this problem, we solve the robustified optimization problem for a robustification $\tau = 2$ with the same specifications. The solution is found in 9.5 hours and relevant solution frames are shown in Fig. 3. Note that this time number of robots in A and C are greater than five starting from $t = 14$ until $t = 16$ and from $t = 30$ until $t = 32$ (corresponds to frame $t = 3$ since the loop starts at $t = 2$), respectively. This implies that even in the worst case of asynchrony, there is going to be at least a single time step where the number of robots in A is greater than five. Additionally, robots are more careful when crossing the bridge. The number of agents robustly staying outside the bridge is always greater than or equal to 8, meaning that the bridge would never be populated by more than 2 robots even in the worst case asynchrony.

Also note that, tasks that do not require synchrony, as in the case of μ_3, are not robustified. No matter what the asynchrony is, as long as each agent visits one of the charging stations for a single time step, μ_3 will be satisfied. The robust solution does not enforce agents to spend unnecessary time in charging station as can be seen from Fig. 3.

VII. Conclusions

In this paper, we proposed a new logic cLTL+ for multi-agent coordination and presented an optimization-based trajectory generation method that guarantees that the collective behavior of the agents satisfy the given cLTL+ formula. We discussed an extension of the trajectory generation method for the case agents move asynchronously.

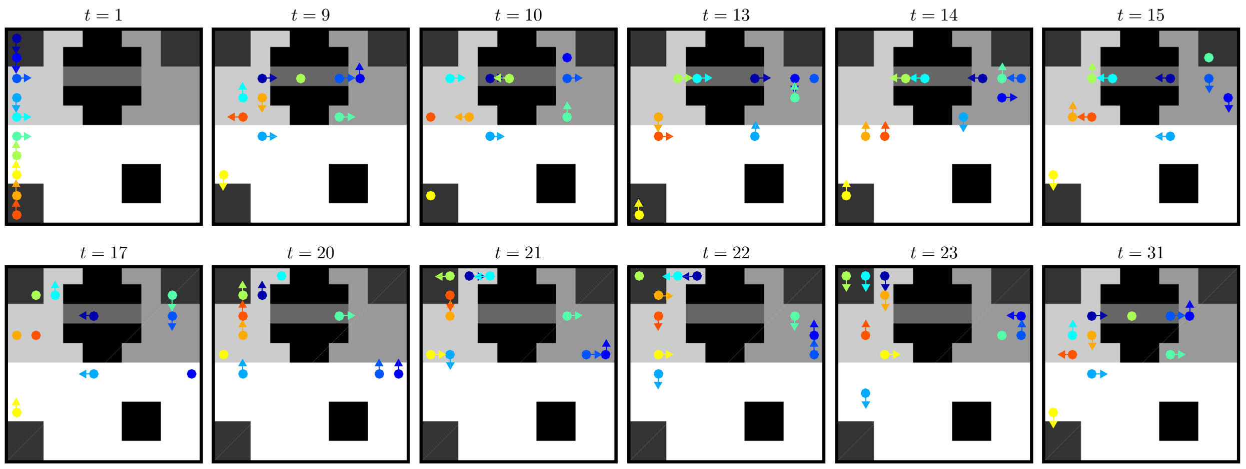

Fig. 2. Important frames from the synthesized non-robust agent trajectories, where arrows indicate direction of movement. The loop starts at frame $t = 9$, thus the state at $t = 9$ is identical to the state at $t = 31$. Time $t = 14$ is the only time step where region C is populated with more than 5 robots. Every agent visits the charging station, and, at time $t = 13$, one robot enters the bridge while another one exits.

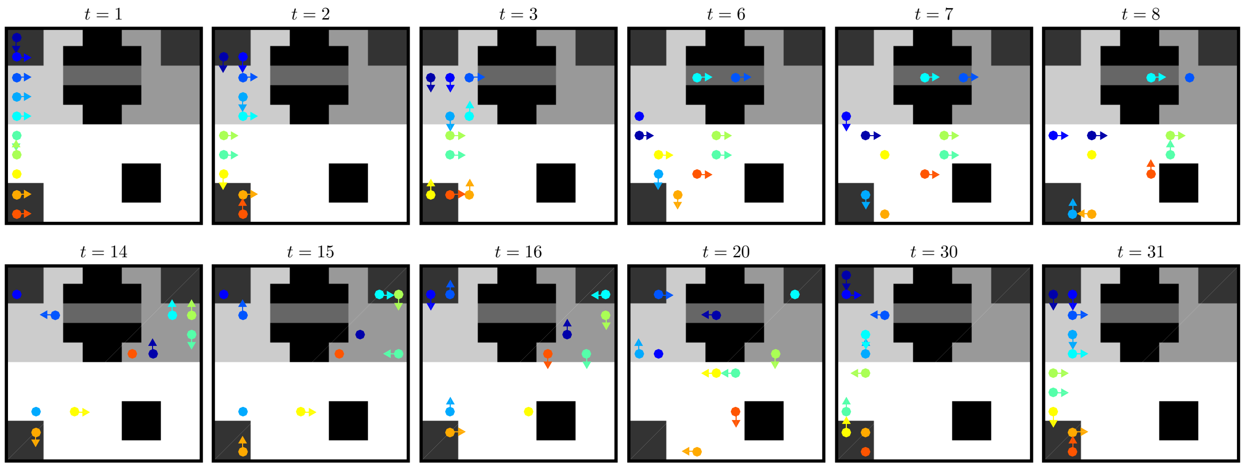

Fig. 3. Important frames from the synthesized robust agent trajectories, where arrows indicate direction of movement. The loop starts at frame $t = 2$, which is identical to frame $t = 31$. Every agent visits the charging station and other specifications, such as populating (emptying) region A and C with greater than or equal to 5 agents, are performed robustly.

There are several possible extensions. A trivial extension would be to allow discrete-time linear or piecewise-linear systems instead of finite transition systems for each agent as done for single agent LTL synthesis in [3], [4]. Another straight-forward extension is to expand the temporal counting constraints to contain a subset of agents, which are capable or responsible for specific tasks, and require the count m to be selected from this subset, similar to censusSTL syntax [7]. A less trivial extension is on the analysis of asynchronous behaviors. We conjecture that completeness of the asynchronous case can be achieved with slightly more complicated encodings. This is the subject of current research.

References

[1] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.
[2] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *Proc. IEEE ICRA*, 2010, pp. 2689–2696.
[3] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *Proc. IEEE ICRA*, 2014, pp. 5319–5325.
[4] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. Seshia, "Model predictive control with signal temporal logic specifications," in *Proc. IEEE CDC*, 2014, pp. 81–87.
[5] Y. E. Sahin, P. Nilsson, and N. Ozay, "Provably-correct coordination of large collections of agents with counting temporal logic constraints," in *Proc. ICCPS*, 2017, pp. 249–258.
[6] A. Desai, I. Saha, J. Yang, S. Qadeer, and S. A. Seshia, "Drona: a framework for safe distributed mobile robotics," in *Proc. ICCPS*. ACM, 2017, pp. 239–248.
[7] Z. Xu and A. A. Julius, "Census signal temporal logic inference for multiagent group behavior analysis," *IEEE Trans. Autom. Sci. Eng.*, vol. PP, no. 99, pp. 1–14, 2016.
[8] A. Biere, K. Heljanko, T. Junttila, T. Latvala, and V. Schuppan, "Linear encodings of bounded LTL model checking," *Logical Methods in Computer Science*, vol. 2, pp. 1–64, 2006.
[9] C. Baier and J. Katoen, *Principles of Model Checking*. MIT Press, 1999.
[10] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017.