CatlNet: Learning Communication and Coordination Policies from CaTL+ Specifications

Wenliang Liu WLIU97@BU.EDU
Boston University, Massachusetts, USA

Kevin Leahy KEVIN.LEAHY@LL.MIT.EDU
MIT Lincoln Laboratory, Lexington, MA, USA

Zachary Serlin ZACHARY.SERLIN@LL.MIT.EDU
MIT Lincoln Laboratory, Lexington, MA, USA

Calin Belta CBELTA@BU.EDU
Boston University, Massachusetts, USA

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

In this paper, we propose a learning-based framework to simultaneously learn the communication and distributed control policies for a heterogeneous multi-agent system (MAS) under complex mission requirements from Capability Temporal Logic plus (CaTL+) specifications. Both policies are trained, implemented, and deployed using a novel neural network model called CatlNet. Taking advantage of the robustness measure of CaTL+, we train CatlNet centrally to maximize it where network parameters are shared among all agents, allowing CatlNet to scale to large teams easily. CatlNet can then be deployed distributedly. A plan repair algorithm is also introduced to guide CatlNet's training and improve both training efficiency and the overall performance of CatlNet. The CatlNet approach is tested in simulation and results show that, after training, CatlNet can steer the decentralized MAS system online to satisfy a CaTL+ specification with a high success rate.

Keywords: multi-agent systems, temporal logic, model-based reinforcement learning, distributed control, communication

1. Introduction

Many real-world missions require the coordination of a heterogeneous Multi-Agent System (MAS). As tasks become increasingly complex, the need for an efficient way to define these tasks for a MAS becomes more and more stringent. This is especially true in cases where agents are controlled using learning-based methods (the focus of this paper), which may be advantageous for large MAS where coordination solutions are difficult to compute in real-time. Due to their expressivity and similarity to natural languages, temporal logics (such as Linear Temporal Logic (LTL) Pnueli (1977) and Signal Temporal Logic (STL) Maler and Nickovic (2004)) have been widely used as specification languages for control systems. More recently, some work has focused on specifically tailoring temporal logics for MAS Xu and Julius (2016); Sahin et al. (2017, 2019); Leahy et al. (2021).

In this paper, we focus on Capability Temporal Logic plus (CaTL+) Liu et al. (2022), which specifies rich task requirements with concrete temporal constraints for heterogeneous MAS. The agents can have different *capabilities* of servicing *tasks*. Besides qualitative semantics (whether requirements are satisfied), CaTL+ is also equipped with quantitative semantics, also called *robust-*

ness, which is a continuous real number that measures how strongly the requirements are satisfied. Taking advantage of this, controlling a MAS to satisfy a CaTL+ specification can be formulated as an optimization problem with the robustness as the objective function. In Liu et al. (2022), this problem was solved in one shot and results in an open-loop controller. However, computing this controller is time-consuming, and the open-loop controller is vulnerable under disturbances.

In this paper we propose a learning-based framework to train a distributed control policy for each agent that collectively attempts to satisfy a given CaTL+ specification. By training the policy off-line, each agent can compute a feedback control in real-time. We assume that each agent can only observe its own state directly. However, satisfying a CaTL+ specification requires coordination of multiple agents, so communication is necessary. In practice, communication resources are usually limited, and how to utilize these resources is a challenging problem in itself. The framework in this paper jointly learns a communication strategy (i.e., when and what each agent needs to communicate given limited bandwidth) together with a control policy. Under this communication strategy, agents only communicate when necessary and transmit the most useful information. Both the control policy and communication strategy are implemented in a model that we call CatlNet, which consists of several neural networks (NNs). We train CatlNet with the centralized training and decentralized execution (CTDE) paradigm. The CatlNet parameters are shared for all agents in the training phase, so no additional parameters are needed when adding more agents, which makes the algorithm scalable for very large teams.

Training the policy from scratch can be difficult especially when the task is complex. It has been shown in the literature (e.g., Leung and Pavone (2022)) that expert demonstrations can help the optimizer converge. However, a dataset of expert demonstrations is not always available. Hence, we design a repair scheme to fix the team trajectory generated by CatlNet such that it satisfies the CaTL+ specification. We use the repair algorithm to generate a dataset of satisfying trajectories to guide training, which is shown to improve the performance of the learned policies.

The main contributions of this paper are twofold: (1) We propose a learning-based framework, called CatlNet, which can learn both the distributed control policy and the communication strategy given limited bandwidth to steer a MAS to satisfy a CaTL+ specification. (2) We designed a repair scheme to guide the training, which improves the performance of CatlNet. We show that the control and communication policies generated by this framework are reliable and computationally efficient.

Due to space limitations, all proofs are omitted in this paper, but can be found on Github[1].

2. Related Work

Controller synthesis from temporal logic specifications has gained significant attention in recent years. Roughly, existing approaches can be divided into two schools of thought: (1) Synthesis for LTL and fragments of LTL, which employ automata-based methods (see, e.g., Belta et al. (2017)); (2) Synthesis for temporal logics defined over real-valued signals, such as STL, which can be formulated as optimization problems solved via Mixed Integer Programming (MIP) Raman et al. (2014), Sadraddini and Belta (2015) or gradient-based methods Pant et al. (2017), Gilpin et al. (2020). Both solution classes have also been extended to MAS. The authors in Chen et al. (2011); Schillinger et al. (2018); Kantaros and Zavlanos (2020); Luo et al. (2021) applied automata-based methods to synthesize distributed control policies from a global LTL specification. Logics specifically designed for MAS including counting LTL (cLTL) Sahin et al. (2017), cLTL+ Sahin et al. (2019), Capability Temporal Logic (CaTL) Leahy et al. (2021) and STL with integral predicates Buyukkocak et al.

1. https://github.com/WenliangLiu1997/CatlNet

(2021) have also been proposed; MIP is used for control synthesis. An extension of CaTL, called CaTL+, was proposed in Liu et al. (2022). Taking advantage of differentiable robustness, control synthesis from CaTL+ is solved using gradient-based methods. All the methods mentioned above either synthesize the control in one shot or compute the control online. Hence, they are computationally very expensive for large MAS, which prohibits their use for real-time control.

Learning-based methods can be used to move online computation offline, which enables real-time executions for the above methods. Reinforcement Learning (RL) was combined with automata-based methods Li et al. (2019); Cai et al. (2021) and optimization-based methods Aksaray et al. (2016); Liu and Belta (2021) to synthesize control policies for a single agent systems under temporal logic specifications. Model-free RL has also been applied to MAS under LTL Sun et al. (2020); Hammond et al. (2021); Zhang et al. (2022) and STL Muniraj et al. (2018) specifications. However, model-free RL requires a large number of trials to learn the policy, which might be infeasible in practice. In this paper, we apply model-based RL and assume the system model is known. Though not included in this work, the model can also be learned from data. This paper can be seen as an extension of Liu and Belta (2021), moving from a single agent system and STL to MAS and CaTL+.

Improving the performance of NNs is investigated in Ma et al. (2020) where a method called STLnet is proposed to project a sequence of NN outputs to satisfy an STL formula. However, STLnet is not designed for control systems and it cannot fix the controller given STL over states. Inspired by STLnet though, we repair the controls given by CatlNet to guide the training.

Finally, this work can also be viewed in the context of multi-agent RL and distributed networks in which communication is learned. Due to the importance of communication in cooperative tasks, many RL frameworks that can simultaneously learn control and communication policies were proposed recently, such as DIAL Foerster et al. (2016), CommNet Sukhbaatar et al. (2016), BicNet Peng et al. (2017) and ATOC Jiang and Lu (2018). The most related framework to ours is ATOC, which is designed for homogeneous MAS with a given reward function. The communication architecture of our CatlNet is inspired from ATOC, and extends it for heterogeneous teams under CaTL+ specifications. Using CaTLNet, rewards are generated automatically from the CaTL+ formula.

3. Preliminaries

3.1. System Model

We use bold and calligraphic symbols to represent trajectories and sets, respectively. $|\mathcal{X}|$ is the cardinality of a set $\mathcal{X}$. Consider a team of agents labelled from a finite set $\mathcal{J}$, where $j \in \mathcal{J}$ denotes an agent's index. We assume that all agents share the same state space $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ and discrete time dynamics (a relaxation to this will be discussed in Remark 1):

$$x_j(t+1) = x_j(t) + u_j(t), \quad t = 0, 1, \ldots, H-1, \tag{1}$$

where $x_j(t) \in \mathcal{X}$ and $u_j(t) \in \mathcal{U}_j \subset \mathbb{R}^{n_u}$ are the state and control at time t, $\mathcal{U}_j$ is the control space of agent j, and H is a finite time horizon determined by the mission specification. Each agent j is assumed to have a random initial state in $\mathcal{X}_{j,0} \subset \mathcal{X}$. Let $P_j : \mathcal{X}_{j,0} \to \mathbb{R}$ be the probability density function of the initial state $x_j(0)$. Consider a finite set of capabilities Cap for team $\mathcal{J}$. Each agent has its own set of capabilities $Cap_j \subseteq Cap$. We assume that $\cup_{j\in\mathcal{J}} Cap_j = Cap$.

The trajectory of an agent j, called an *individual trajectory*, is a sequence $\mathbf{x}_j = x_j(0) \ldots x_j(H)$. Then *team trajectory* is defined as a set of pairs $\mathbf{X} = \{(\mathbf{x}_j, Cap_j)\}_{j\in\mathcal{J}}$, which captures all the *individual trajectories* with their corresponding capabilities. Here we include capabilities in a team

trajectory so that we can define the semantics of CaTL+ on it, as it will be shown later. Let $\mathcal{J}_c = \{j \mid c \in Cap_j\}$ be the set of agent indices with capability c. Let $\mathbf{u}_j = u_j(0)\dots u_j(H-1)$ be the sequence of controls for agent j, $\bar{x}(t) = [x_j(t)]_{j=1}^{|\mathcal{J}|}$ and $\bar{u}(t) = [u_j(t)]_{j=1}^{|\mathcal{J}|}$ be the joint state and control of the MAS at time t. Denote $\mathbf{x}_j^{0:t} = x_j(0)\dots x_j(t)$ and $\bar{\mathbf{x}}^{0:t} = \bar{x}(0), \dots, \bar{x}(t)$.

Remark 1 *We simplify each agent's dynamics to a single integrator as in (1) such that all agents can be controlled by CatlNet (described in Sec. 5) with same parameters. In fact, these dynamics can be seen as high level nominal dynamics used to generate a sequence of waypoints. The true dense-time dynamics of the agents can be heterogeneous, as long as all agents share a common workspace $\mathcal{X}$ (not necessarily the state space). By properly selecting the control constraint $\mathcal{U}_j$, we can find a local controller for each agent that tracks the individual trajectory (waypoints) within given time window and avoids inter-agent collision, using the techniques in Sun et al. (2022).*

3.2. CaTL+ Syntax and Semantics

Capability Temporal Logic plus (CaTL+) Liu et al. (2022) is a two-layer logic that includes an inner logic and outer logic. The *inner* logic defined over individual trajectories $\mathbf{x}$ (subscript j omitted for simplicity) is identical to STL and has the following syntax:

$$\varphi := True \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \mathbf{U}_{[a,b]} \varphi_2, \tag{2}$$

where φ, φ_1 and φ_2 are inner logic formulas, μ is a *predicate* in the form of $f(x(t)) \geq 0$. We assume $f : \mathcal{X} \to \mathbb{R}$ is a differentiable function. $\neg$, $\wedge$, $\vee$ are the Boolean *not*, *conjunction* and *disjunction* respectively. $\mathbf{U}_{[a,b]}$ is the temporal operator *until*, where $\varphi_1 \mathbf{U}_{[a,b]} \varphi_2$ means "φ_2 must become true at some time point in $[a, b]$ and φ_1 must stay true before that", $[a, b]$ are all integer time points between a and b. Other temporal operators like *eventually* $\mathbf{F}_{[a,b]}\varphi$ and *always* $\mathbf{G}_{[a,b]}\varphi$ are defined as $\mathbf{F}_{[a,b]}\varphi = True \mathbf{U}_{[a,b]}\varphi$ and $\mathbf{G}_{[a,b]}\varphi = \neg\mathbf{F}_{[a,b]}\neg\varphi$, where $\mathbf{F}_{[a,b]}\varphi$ states that "φ becomes true at some time point in $[a, b]$" and $\mathbf{G}_{[a,b]}\varphi$ states that "φ stays true at all time points in $[a, b]$". An individual trajectory $\mathbf{x}$ satisfies a inner logic (STL) φ at time t is denoted as $(\mathbf{x}, t) \models \varphi$.

The outer logic (with a slight abuse of terminology we refer it as CaTL+), which is defined over team trajectories, has similar syntax with STL, except for predicates μ are replaced by *tasks* T:

$$\Phi := True \mid T \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \Phi_1 \mathbf{U}_{[a,b]} \Phi_2, \tag{3}$$

where Φ, Φ_1 and Φ_2 are CaTL+ formulas, $T = \langle \varphi, c, m \rangle$ is a *task*, φ is an inner logic formula, $c \in Cap$ is a capability, and m is a positive integer. The other operators are the same as the ones in STL. A task is satisfied at time t if and only if at least m *individual trajectories* of agents with capability c satisfy φ at time t. Formally, we define a counting function $n(\mathbf{X}, c, \varphi, t)$ to capture this:

$$n(\mathbf{X}, c, \varphi, t) = \sum_{j \in \mathcal{J}_c} I\big((\mathbf{x}_j, t) \models \varphi\big), \tag{4}$$

where I is an indicator function, i.e., $I = 1$ if $(\mathbf{x}_j, t) \models \varphi$ and $I = 0$ otherwise. Then the team trajectory $\mathbf{X}$ satisfies T at time t, denoted by $(\mathbf{X}, t) \models T$, if and only if $n(\mathbf{X}, c, \varphi, t) \geq m$.

CaTL+ not only has qualitative semantics, i.e., *whether* $\mathbf{X}$ satisfies Φ, but also has quantitative semantics (also called robustness), i.e., *how much* the specification is satisfied or violated. Denote the (exponential) robustness of a CaTL+ formula Φ with respect to a team trajectory $\mathbf{X}$ at time t as $\eta(\mathbf{X}, \Phi, t)$, which is differentiable almost everywhere. The detailed definition of η can be found in Liu et al. (2022). The robustness of CaTL+ is sound, i.e., $\eta(\mathbf{X}, \Phi, t) \geq 0$ if and only if $(\mathbf{X}, t) \models \Phi$.

The time horizon of a CaTL+ formula Φ, denoted by $hrz(\Phi)$, is defined as the closest future time point that is needed to decide the satisfaction of Φ.

Example 1 *To provide a comparison, we use the earthquake emergency response scenario defined in Liu et al. (2022). The workspace $\mathcal{X} \subset \mathbb{R}^2$ is shown in Fig. 3(a). There are 4 ground vehicles $j \in \{1,2,3,4\}$ and 2 aerial vehicles $j \in \{5,6\}$, totaling 6 robots indexed from $\mathcal{J} = \{1,2,3,4,5,6\}$. A bridge B goes across a river R in the area. All ground vehicles start from initial state $x_j(0)$ uniformly sampled in region $Init_g$ and have capabilities $Cap_j = \{\text{“}Delivery\text{”}, \text{“}Ground\text{”}\}$, $j \in \{1,2,3,4\}$. All the aerial vehicles have initial state $x_j(0)$ uniformly sampled in the region $Init_a$ and have capabilities $Cap_j = \{\text{“}Delivery\text{”}, \text{“}Inspection\text{”}\}$, $j \in \{5,6\}$.*

Consider the following CaTL+ specifications: (1) $\Phi_1 = \langle \mathbf{F}_{[0,8]} x \in C,\ \text{“}Delivery\text{”},\ 6\rangle$: 6 agents with capability “$Delivery$” should pick up supplies from region C within 8 time units; (2) $\Phi_2 = \langle \mathbf{F}_{[0,25]} x \in V_1,\ \text{“}Delivery\text{”},\ 3\rangle \wedge \langle \mathbf{F}_{[0,25]} x \in V_2,\ \text{“}Delivery\text{”},\ 3\rangle$: 3 agents with capability “$Delivery$” should deliver supplies to the affected village V_1 and V_2 within 25 time units, respectively; (3) $\Phi_3 = \neg\langle x \in B,\ \text{“}Ground\text{”},\ 1\rangle \mathbf{U}_{[0,5]} \langle x \in B,\ \text{“}Inspection\text{”},\ 2\rangle$: any agent with capability “$Ground$” cannot go over the bridge until 2 agents with capability “$Inspection$” inspect it within 5 time units; (4) $\Phi_4 = \mathbf{G}_{[0,25]} \langle \neg(x \in R),\ \text{“}Ground\text{”},\ 4\rangle$: agents with capability “$Ground$” should always avoid entering the river R; (5) $\Phi_5 = \mathbf{G}_{[0,25]} \neg\langle x \in B,\ \text{“}Ground\text{”},\ 2\rangle$: Since the load of the bridge is limited, at all times no more than 1 agent with capability “$Ground$” can be on B; (6) $\Phi_6 = \mathbf{G}_{[0,25]} \langle x \in M,\ \text{“}Delivery\text{”},\ 6\rangle$: 6 agents with capability “$Delivery$” should always stay in region M. The overall specification for the system is $\Phi = \bigwedge_{i=1}^{6} \Phi_i$, with $hrz(\Phi) = 25$. An example team trajectory is shown in Fig. 3(a)subfigure, where Φ_1 is satisfied because all 6 agents enter C while Φ_4 is violated since a ground vehicle falls into R.

4. Problem Formulation and Approach

Consider a team of agents $\mathcal{J}$ that needs to collaboratively satisfy a CaTL+ specification Φ. We assume that: (1) each agent can only observe its own state $x_j(t)$ at each time t; (2) all agents have access to a communication channel for all times. At each time t, each agent can broadcast a vector $h_j^t \in \mathbb{R}^{n_c}$ to the channel and receive a vector $\tilde{h}_j^t \in \mathbb{R}^{n_c}$ from the channel. The dimension of the communication vectors are fixed because of the limited bandwidth of the channel. We formulate the joint control and communication synthesis problem as:

Problem 1 *Given a multi-agent system $\mathcal{J}$ with initial states $\{x_j(0) \in \mathcal{X}_{0,j} \mid j \in \mathcal{J}\}$ distributed as P_j and a CaTL+ specification Φ defined over the team trajectory $\mathbf{X}$, find the control policy $u_j(t) = \pi_j(\mathbf{x}_j^{0:t}, h_j^t, \tilde{h}_j^t)$ and the communication vectors h_j^t and $\tilde{h}_j^t$ that maximize the objective:*

$$\begin{aligned} \max_{\pi_j, h_j^t, \tilde{h}_j^t,\ j\in\mathcal{J}} \quad & \eta(\mathbf{X}, \Phi, 0) - \gamma \cdot \max\big(\eta(\mathbf{X}, \Phi, 0), 0\big) \cdot \sum_{j\in\mathcal{J}} \sum_{t=0}^{H-1} C\big(\pi_j(\mathbf{x}_j^{0:t}, h_j^t, \tilde{h}_j^t)\big) \\ \text{s.t.} \quad & x_j(t+1) = f_j(x_j(t), \pi_j(\mathbf{x}_j^{0:t}, h_j^t, \tilde{h}_j^t)), \\ & \pi_j(\mathbf{x}_j^{0:t}, h_j^t, \tilde{h}_j^t) \in \mathcal{U}_j,\ t = 0, \ldots, H-1, \end{aligned} \tag{5}$$

where $\eta(\mathbf{X}, \Phi, 0)$ is the CaTL+ robustness, $C(\cdot)$ is a cost function, $H \geq hrz(\Phi)$ is the planning horizon, and γ is a parameter satisfying $\gamma^{-1} \geq \sup_{u_j(t)\in\mathcal{U}_j} \sum_{j\in\mathcal{J}} \sum_{t=0}^{H-1} C(u_j(t))$.

In addition, our secondary objective is to minimize the total number of times that agents access the communication channel to save the energy cost on communication.

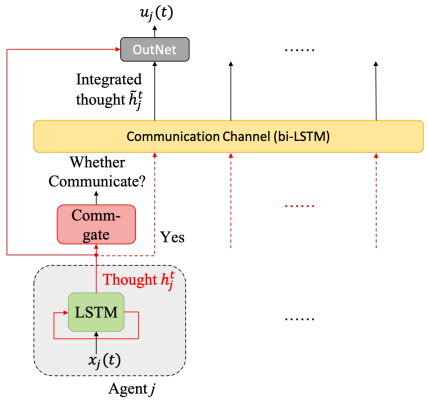


Figure 1: The overall architecture of CatlNet.

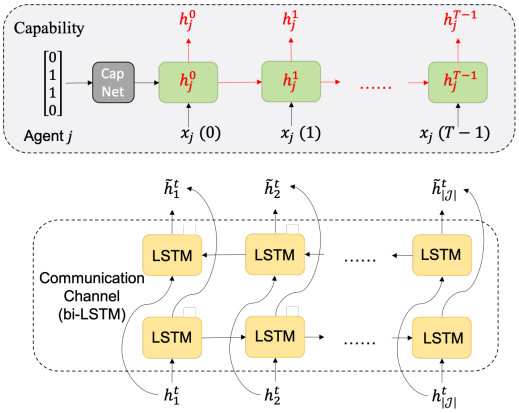


Figure 2: CapNet and unfolded LSTM (above), bi-directional LSTM (bottom).

Since the satisfaction of Φ is always a priority, the constraint on γ ensures that the objective function has the same sign as $\eta(\mathbf{X}, \Phi, 0)$, so minimizing the cost never overrides maximizing the robustness. Note that to determine an agent's control at time t, history states of the agent are needed due to the temporal requirements (as described in Liu et al. (2021)).

A straightforward method to get π_j in Pb. 1 is to apply Model Predictive Control (MPC), i.e., compute a sequence of controls within a planning horizon and apply the first one to the system at each time. Since the objective function is defined on the entire team, the history information of all agents is required by the MPC controller at all time. This information needs to be either stored at a central node (taking up a large storage space) where $h_j^t = x_j(t)$, or sent from each agent to the communication channel (i.e., $h_j^t = \mathbf{x}_j^{0:t}$, which results in a large communication vector). Moreover, since the robustness of CaTL+ is nonconvex, solving the MPC problem at each time step can be time-consuming. Finally, MPC requires all agents to connect to the channel at all time steps, which is not going to accomplish our secondary objective. Hence, this MPC approach can be intractable in practice due to the limits on communication, storage and the real-time computation requirement.

In this paper, we propose a learning-based algorithm (detailed in Sec. 5) to solve Pb. 1. The algorithm finds the communication strategy, i.e., whether an agent needs to communicate at time t and what the communication vectors h_j^t and $\tilde{h}_j^t$ are, and a distributed control policy for each agent that can compute the control in real time. We propose a NN-based framework, called CatlNet, to implement both policies. CTDE paradigm is applied, i.e., we assume all state information is known during training. During execution, each agent only observe its own state but it is connected to a communication channel. After training, CatlNet can be generalized to random initial states $x_j(0)$.

5. CatlNet

5.1. Architecture of CatlNet

The overall architecture of CatlNet is shown in Fig. 1. We extract a vector h_j^t called *thought* from the state of agent j at time t using a Long Short Term Memory (LSTM) NN with parameters θ_L (Fig 2 above). Here the thought contains the information of history states. Let Cap_j^v be the vectorized representation of agent j's capability, where $Cap_j^v = [b_1\ b_2 \cdots b_{|Cap|}]^\top$, $b_i = 1$ if $c_i \in Cap_j$, $b_i = 0$ if $c_i \notin Cap_j$. We input Cap_j^v to a NN called CapNet with parameters θ_{cap} and use its output as the initial hidden state of the LSTM as shown in Fig. 2 (above). The output of the LSTM, i.e., the

thought h_j^t is then fed to a classifier, called a Comm-gate, to decide whether the agent communicates. The Comm-gate is also implemented as a NN with parameters θ_g. If the Comm-gate decides to communicate, then the thought h_j^t is passed to a communication channel implemented by a bi-directional LSTM with parameters θ_b. The communication channel can merge all agents' thoughts (who decide to communicate at that time) and output an integrated thought $\tilde{h}_j^t$ that guides agents to generate coordinated actions as shown in Fig. 2 (bottom). Then the integrated thought $\tilde{h}_j^t$ is sent back to agent j and concatenated with its original thought h_j^t. Finally, $[h_j^t, \tilde{h}_j^t]$ is fed to another NN called OutNet with parameters θ_o, and it outputs the control $u_j(t)$. A hyperbolic tangent function is applied at the last layer of OutNet to satisfy the constraint $u_j \in \mathcal{U}_j$ as in Yaghoubi and Fainekos (2019). We denote the joint control policy given by CatlNet as $\bar{u}(t) = \bar{\pi}(\bar{\mathbf{x}}_{0:t}, \theta_{cap}, \theta_L, \theta_b, \theta_o, \theta_g)$, where the (integrated) thoughts h_j^t ($\tilde{h}_j^t$) are internal variables for the joint policy, therefore omitted.

Note that in CatlNet all agents share the same parameters (θ_{cap}, θ_L, θ_b, θ_o, θ_g), so the number of trainable parameters does not increase as the number of agents increases. Capabilities fed to CapNet make different type of agents behave differently, while communication can make same type of agents behave differently. Hence, communication is necessary when such diversity is needed.

5.2. Training of CatlNet

Following the CTDE paradigm, CatlNet is trained as two parts: (1) the policy networks, including CapNet, LSTM, communication channel and OutNet; and (2) the Comm-gate.

5.2.1. TRAINING OF THE POLICY NETWORKS

We initially ignore the Comm-gate network training and let all agents communicate openly to the channel, denoted as $\theta_g = \theta_g^{full}$. Since the control policy has been parameterized by CatlNet and we want to generalize CatlNet to different initial states, the first objective in Pb. 1 becomes:

Problem 2 *Given a multi-agent system $\{A_j \mid j \in \mathcal{J}\}$ and a CaTL+ specification Φ defined over the team trajectory $\mathbf{X}$, find the optimal CatlNet parameters θ_{cap}, θ_L, θ_b, θ_o that maximizes the objective:*

$$\begin{aligned} \max_{\theta_{cap},\theta_L,\theta_b,\theta_o} \quad & E_{P(x_j(0))}\big[\eta(\mathbf{X},\Phi,0) - \gamma \cdot \max\big(\eta(\mathbf{X},\Phi,0),0\big) \cdot \sum_{j\in\mathcal{J}} C(\mathbf{u}_j)\big] \\ \textit{s.t.} \quad & \bar{u}(t) = \bar{\pi}(\bar{\mathbf{x}}_{0:t}, \theta_{cap}, \theta_L, \theta_b, \theta_o, \theta_g^{full}), \\ & x_j(t+1) = f_j(x_j(t), u_j(t)),\ t = 0, \ldots, H-1. \end{aligned} \tag{6}$$

In practice, we randomly sample M initial states of the MAS and use the average to approximate the expectation in (6). We substitute the constraints to the objective function to make (6) an unconstrained optimization problem and use the Adam stochastic optimizer Kingma and Ba (2014) to update θ_{cap}, θ_L, θ_b, θ_o. We resample M initial states at each optimization step. Note that all gradients can be computed automatically and analytically using the technique in Leung et al. (2020).

When the CaTL+ specification is complex, the policy may become stuck in a local optima that violates the specification. To improve training reliability, we consider how humans learn. (1) Given an objective, a human learner might achieve it with a suboptimal solution. On the other hand, if optimal demonstrations are provided, a learner who do not know the objective can imitate but might fail given unseen conditions. Hence, the best strategy is to provide the learner both the objective and the demonstrations. (2) Facing a bunch of unfamiliar demonstrations, it might be hard for a learner to discover the underlying rules. However, if a coach shows the learner a solution which is

an adaptation of the learner's own behavior, it will be easier for the learner to improve her solution. Inspired by these two insights, we designed a repair algorithm (Alg. 1 described in Sec. 5.3), which can fix the team trajectory generated by CatlNet to satisfy the CaTL+ specification. We use the repair algorithm to guide the training and further improve the policy.

We use the trained CatlNet to generate a set of N team trajectories starting from random initial states and collect the violating trajectories. Then we use Alg. 1 to repair them and collect all successfully repaired team trajectories to form a dataset $D = \{\mathbf{X}_d^{(i)} | i = 1, \dots, N\}$. Denote the objective in (6) as $L(\theta_{cap}, \theta_L, \theta_b, \theta_o, \theta_g)$. Then train CatlNet again to maximize the objective:

$$\begin{aligned} \max_{\theta_{cap},\theta_L,\theta_b,\theta_o} \quad & (1-\beta) L(\theta_{cap}, \theta_L, \theta_b, \theta_o, \theta_g^{full}) - \beta \sum_{i=1}^{N} \sum_{t=0}^{T-1} \|\bar{x}^{(i)}(t) - \bar{x}_d^{(i)}(t)\|^2 \\ \text{s.t.} \quad & x_j^{(i)}(t+1) = f_j(x_j^{(i)}(t), u_j^{(i)}(t)), \quad x_j^{(i)}(0) = x_{j,D}^{(i)}(0), \\ & \bar{u}^{(i)}(t) = \bar{\pi}(\bar{\mathbf{x}}_{0:t}^{(i)}, \theta_{cap}, \theta_L, \theta_b, \theta_o, \theta_g^{full}),\ t = 0, \dots, H-1,\ i = 1, \dots, N, \end{aligned} \tag{7}$$

where $\beta \in [0, 1]$ balances maximizing (6) and imitating the dataset. Note that we can rearrange the identical agents in the team to minimize $\|\bar{x}^{(i)}(t) - \bar{x}_d^{(i)}(t)\|^2$ in (7), which makes the dataset permutation-invariant. We keep repairing the violated trajectories, adding them to dataset D and retrain CatlNet until convergence. Now we have obtained a control policy with full communication.

5.2.2. TRAINING OF THE COMM-GATE

To train the Comm-gate that decides when an agent needs to communicate, we first use the objective (7) with the final dataset to train another CatlNet with no communication at all (likely it cannot satisfy the specification). Then we generate trajectories using the full communication CatlNet, but disable the communication channel connection for one agent j at one time point t at a time. The chosen agent j will use the no communication CatlNet instead at the chosen time t. Then we compare the robustness values with and without this deactivation. If the deactivation makes the robustness decrease over a threshold, we label the thought $h_j^{t,(i)}$ with $y^{(i)} = 1$ (the agent should communicate), otherwise we label $h_j^{t,(i)}$ with $y^{(i)} = 0$ (the agent does not need communication). Repeat this from sampled initial states to form a dataset D_g consist of data pairs $(h_j^{t,(i)}, y^{(i)})$ that covers all agents at all time points. Let $G(h_j^t, \theta_g) \in \mathbb{R}^2$ be the output of Comm-gate. We train the Comm-gate on D_g as a standard classifier to minimize the cross entropy loss:

$$\min_{\theta_g} \sum_{D_g} -y^{(i)} \log\left(\sigma\big(G(h_j^{t,(i)}, \theta_g)\big)_1\right) - (1 - y^{(i)}) \log\left(\sigma\big(G(h_j^{t,(i)}, \theta_g)\big)_2\right) \tag{8}$$

where $\sigma : \mathbb{R}^2 \to \mathbb{R}^2$ is the softmax function, $\sigma()_1$ and $\sigma()_2$ denote its 1st and 2nd elements. Finally, we train CatlNet with the Comm-gate using the objective (7), which gives us the final CatlNet.

5.3. Repair of CatlNet

Now we describe the repair algorithm. First, rewrite the outer logic of CaTL+ into a negation-free Disjunctive Normal Form (DNF, a disjunction of conjunctions). To do this, we extend the CaTL+ task to a *timed task*, denoted as $\bar{T} = \langle \varphi, c, m \rangle_t$, such that $(\mathbf{X}, t_0) \models \bar{T}$ iff $(\mathbf{X}, t_0 + t) \models \langle \varphi, c, m \rangle$, i.e., the task is required to be satisfied at t. The original task $\langle \varphi, c, m \rangle$ is equivalent to $\langle \varphi, c, m \rangle_{t=0}$.

Proposition 2 *Every CaTL+ formula can be represented in the negation-free DNF:* $\bigvee_{k=1}^{K} \bigwedge_{i=1}^{I_k} \bar{T}_i^k$ *where* $\bar{T}_i^k$ *is a timed task,* I_k *is the number of conjunctions in the* k^{th} *disjunction.*

Let $sort(\cdot)$ reorder a sequence of scalars from largest to smallest and return the reordered index:

$$sort([\eta_i]_{i=1}^N) = i_1, i_2, \ldots, i_N,\ s.t.\ \eta_{i_1} \geq \eta_{i_2} \geq \ldots \geq \eta_{i_N}. \tag{9}$$

Next, we follow Alg. 1 to repair the output of CatlNet. We assume that an STL control synthesis algorithm is available. That is, given an STL formula φ defined over an agent's individual trajectory, we can find the control sequence for the agent that steers it to satisfy the STL formula if a solution exists, denoted as $\mathbf{x}, \mathbf{u} \leftarrow syn(\varphi)$. We first rewrite the CaTL+ formula into its negation-free DNF $\Phi = \bigvee_{k=1}^{K} \bigwedge_{i=1}^{I_k} \bar{T}_i^k$. Then at the initial state, we predict the team trajectory using CatlNet and the system model. To satisfy Φ, at least one of the k clauses $\bigwedge_{i=1}^{I_k} \bar{T}_i^k$ needs to be satisfied. We calculate the robustness for all of them and consider these clauses from the highest robustness to the lowest (step 1). For the clause $\bigwedge_{i=1}^{I_k} \bar{T}_i^k$, if it is violated, we find all tasks $\bar{T}_i^k = \langle \varphi_i^k, c_i^k, m_i^k \rangle_{t_i^k}$ that are violated (or satisfied by exactly m_i^k agents) and assign a STL formula $\mathbf{F}_{[t_i^k, t_i^k]} \varphi_i^k$ to enough (m_i^k) agents with the required capabilities c_i^k (steps 2-6). Since we repair one agent at a time, those tasks satisfied by more than m_i^k agents would not be violated. Now we have assigned a set of STL formulas to each agent that needs repair. Then we apply the STL control synthesis algorithm to make an agent's trajectory satisfy the conjunction of these formulas (step 8). After updating the trajectory, we find tasks satisfied by exactly m_i^k agents and assign $\mathbf{F}_{[t_i^k, t_i^k]} \varphi_i^k$ to them again (steps 9-11). Repeat until all agents are repaired. If all agents get positive robustness, then the algorithm terminates and return *success* (step 12). Otherwise redo all these steps for the next clause $\bigwedge_{i=1}^{I_k} \bar{T}_i^k$. If all clauses cannot be satisfied, then the algorithm terminates and return *fail*.

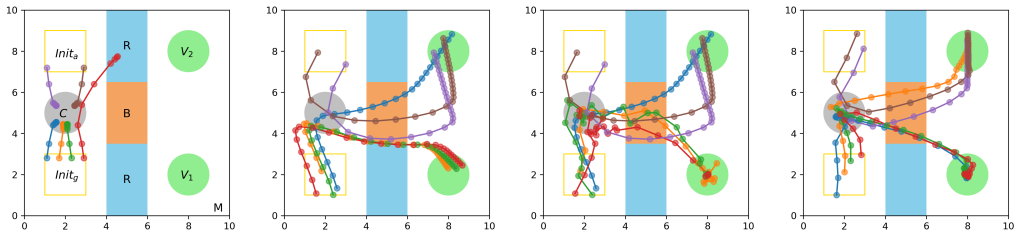
Algorithm 1: Trajectory Repair

Input: $\Phi = \bigvee_{k=1}^{K} \bigwedge_{i=1}^{I_k} \bar{T}_i^k$, $R_1 = \cdots = R_{|J|} = \emptyset$, $F_1 = \cdots = F_{|J|} = 0$, $Result = fail$

1 **for** k *in* $sort\big([\eta(\mathbf{X}, \bigwedge_{i=1}^{|I_k|} T_i^k, 0)]_{k=1}^K\big)$ **do** // all clauses
2 **for** $\{i \in [1, I_k] \mid n(\mathbf{X}, c_i^k, \varphi_i^k, t_i^k) \leq m_i^k)\}$ **do** // all not (just) satisfied $\bar{T}_i^k$
3 **for** j^* *in* $sort\{\rho(x_j, \varphi_i^k, t_i^k) | j \in \mathcal{J}_{c_i^k}\}$ **do** // all agents with c_i^k
4 $R_{j^*} \leftarrow R_{j^*} \cup \{i\}$; // assign task i to agent j^*
5 **if** $(\mathbf{x}_{j^*}, t_i^k) \not\models \varphi_i^k$ **then** $F_{j^*} \leftarrow 1$; // flag agents that need repair;
6 **if** *task* T_i^k *is assigned to* m_i^k *agents* **then** break;
7 **for** $\{j \in \mathcal{J} | F_j = 1\}$ **do** // for all flagged agents
8 $\mathbf{x}_j, \mathbf{u}_j \leftarrow syn(\bigwedge_{i \in R_j} \mathbf{F}_{[t_i^k, t_i^k]} \varphi_i^k)$; // get repaired trajectory
9 **for** $\{i \in [1, I_k] \mid n(\mathbf{X}, c_i^k, \varphi_i^k, t_i^k) = m_i^k)\}$ **do**
10 **for** $j \in \mathcal{J}_{c_i^k}$ **do**
11 **if** $(\mathbf{x}_j, t_i^k) \models \varphi_i$ **then** $R_j \leftarrow R_j \cup \{i\}$ // redo assignment;
12 **if** *all flagged agents get positive robustness* **then** $Result \leftarrow suc$, break;
13 **return** $\mathbf{X}, \mathbf{u}_1, \cdots, \mathbf{u}_{|\mathcal{J}|}, Result$.

Proposition 3 (soundness) *If Alg. 1 returns* $Result = suc$*, then the returned team trajectory* $\mathbf{X}$ *satisfies the CaTL+ specification:* $(\mathbf{X}, 0) \models \Phi$.

Remark 4 *Alg. 1 ensures soundness (Proposition 3) but it is not complete. Here completeness means that if a solution (a team trajectory that satisfies the CaTL+ specification) exists, then the algorithm can find it. In other words, it is possible that Alg. 1 returns* $fail$ *though a solution exists.*



(a) (b) (c) (d)

Figure 3: (a) Environment and example trajectories. (b) Team trajectory before repair. (c) Team trajectory after repair. (d) Team trajectory generated by the final CatlNet.

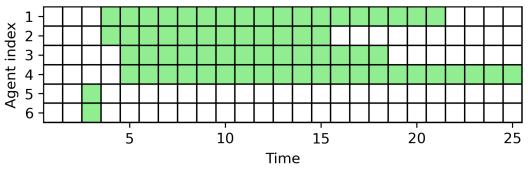


Figure 4: Communication at each time step. Green squares mean the agent communicates at that time, while white squares indicate that the agent does not communicate.

Remark 5 *Alg. 1 requires the global information, so it is only applied in the centralized training phase. Hence, the satisfaction of the CaTL+ specification in the execution is not guaranteed. However, simulation results show that by using the repair scheme as a guidance for the training, CatlNet can reach a high satisfying rate in the execution without the repair. Since we repair the trajectories one by one, the computation of Alg. 1 increases linearly with respect to the number of the agents.*

6. Case Studies

Consider the scenario and CaTL+ specification in Ex. 1. Let the dimension of the communication vector be 8, which contains the history information of each agent. Let $\mathcal{U}_j = [-1, 1]^2$, $j = 1, 2, 3, 4$, $\mathcal{U}_j = [-1.2, 1.2]^2$, $j = 5, 6$. We train CaTL+ using the algorithm described in Sec. 5.2. Detailed NN architecture can be found in our Github repository. A team trajectory generated by the CatlNet after first training (full communication and no repair) is shown in Fig. 3(b). The trajectory violates the CaTL+ specification, mostly due to Φ_5, i.e., only one ground vehicle can be on the bridge at a given time. To avoid appearing on the bridge at the same time, the agents tend to go across the bridge at the edge of the bridge, which is a local optimum with zero robustness. We use Alg. 1 to repair the trajectory in Fig. 3(b) which results in Fig. 3(c) with positive robustness. Then we retrain CatlNet with the dataset and repeat the above process. The final dataset contains 213 trajectories.

Next, we train the Comm-gate and retrain the policy networks with it. A trajectory given by the final CatlNet is shown in Fig. 3(d). The corresponding communication at each time is shown in Fig. 4. It can be seen that Comm-gate greatly reduces the total number of communications and the communication happens mainly when agents go across the river one by one. This makes sense as agents need to behave differently at this stage of the task and communication enables them to do this. We test the final CatlNet from 10000 random initial states and the success rate is 100.00%.

7. Conclusion and Future Work

We proposed a neural network-based model called CatlNet to learn both communication and distributed control policies from CaTL+ specifications. By using the repair algorithm during training, CatlNet can reach a high success rate of satisfying the specification. We plan to incorporate a lower level controller with CatlNet to avoid inter-agent collision and guarantee dense-time behaviors.

Acknowledgments

This work was partially supported by NSF under Grant IIS-2024606.
DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.

References

Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. Q-learning for robust satisfaction of signal temporal logic specifications. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6565–6570. IEEE, 2016.

Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. *Formal methods for discrete-time dynamical systems*, volume 15. Springer, 2017.

Ali Tevfik Buyukkocak, Derya Aksaray, and Yasin Yazıcıoğlu. Planning of heterogeneous multi-agent systems under signal temporal logic specifications with integral predicates. *IEEE Robotics and Automation Letters*, 6(2):1375–1382, 2021.

Mingyu Cai, Mohammadhosein Hasanbeig, Shaoping Xiao, Alessandro Abate, and Zhen Kan. Modular deep reinforcement learning for continuous motion planning with temporal logic. *IEEE Robotics and Automation Letters*, 6(4):7973–7980, 2021.

Yushan Chen, Xu Chu Ding, and Calin Belta. Synthesis of distributed control and communication schemes from global ltl specifications. In *2011 50th IEEE conference on decision and control and european control conference*, pages 2718–2723. IEEE, 2011.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Yann Gilpin, Vince Kurtz, and Hai Lin. A smooth robustness measure of signal temporal logic for symbolic control. *IEEE Control Systems Letters*, 5(1):241–246, 2020.

Lewis Hammond, Alessandro Abate, Julian Gutierrez, and Michael Wooldridge. Multi-agent reinforcement learning with temporal logic specifications. *arXiv preprint arXiv:2102.00582*, 2021.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *Advances in neural information processing systems*, 31, 2018.

Yiannis Kantaros and Michael M Zavlanos. Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems. *The International Journal of Robotics Research*, 39(7):812–836, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kevin Leahy, Zachary Serlin, Cristian-Ioan Vasile, Andrew Schoer, Austin M Jones, Roberto Tron, and Calin Belta. Scalable and robust algorithms for task-based coordination from high-level specifications (scratches). *IEEE Transactions on Robotics*, 2021.

Karen Leung and Marco Pavone. Semi-supervised trajectory-feedback controller synthesis for signal temporal logic specifications. *arXiv preprint arXiv:2202.01997*, 2022.

Karen Leung, Nikos Aréchiga, and Marco Pavone. Back-propagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 432–449. Springer, 2020.

Xiao Li, Zachary Serlin, Guang Yang, and Calin Belta. A formal methods approach to interpretable reinforcement learning for robotic planning. *Science Robotics*, 4(37):eaay6276, 2019.

Wenliang Liu and Calin Belta. Model-based safe policy search from signal temporal logic specifications using recurrent neural networks. *arXiv preprint arXiv:2103.15938*, 2021.

Wenliang Liu, Noushin Mehdipour, and Calin Belta. Recurrent neural network controllers for signal temporal logic specifications subject to safety constraints. *IEEE Control Systems Letters*, 6:91–96, 2021.

Wenliang Liu, Kevin Leahy, Zachary Serlin, and Calin Belta. Robust multi-agent coordination from catl+ specifications. *arXiv preprint arXiv:2210.01732*, 2022.

Xusheng Luo, Yiannis Kantaros, and Michael M Zavlanos. An abstraction-free method for multirobot temporal logic optimal control synthesis. *IEEE Transactions on Robotics*, 37(5):1487–1507, 2021.

Meiyi Ma, Ji Gao, Lu Feng, and John Stankovic. Stlnet: Signal temporal logic enforced multivariate recurrent neural networks. *Advances in Neural Information Processing Systems*, 33: 14604–14614, 2020.

Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

Devaprakash Muniraj, Kyriakos G Vamvoudakis, and Mazen Farhood. Enforcing signal temporal logic specifications in multi-agent adversarial environments: A deep q-learning approach. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4141–4146. IEEE, 2018.

Yash Vardhan Pant, Houssam Abbas, and Rahul Mangharam. Smooth operator: Control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1235–1240. IEEE, 2017.

Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. ieee, 1977.

Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, pages 81–87. IEEE, 2014.

Sadra Sadraddini and Calin Belta. Robust temporal logic model predictive control. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 772–779. IEEE, 2015.

Yunus Emre Sahin, Petter Nilsson, and Necmiye Ozay. Provably-correct coordination of large collections of agents with counting temporal logic constraints. In *2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)*, pages 249–258. IEEE, 2017.

Yunus Emre Sahin, Petter Nilsson, and Necmiye Ozay. Multirobot coordination with counting temporal logics. *IEEE Transactions on Robotics*, 36(4):1189–1206, 2019.

Philipp Schillinger, Mathias Bürger, and Dimos V Dimarogonas. Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. *The international journal of robotics research*, 37(7):818–838, 2018.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29, 2016.

Chuangchuang Sun, Xiao Li, and Calin Belta. Automata guided semi-decentralized multi-agent reinforcement learning. In *2020 American Control Conference (ACC)*, pages 3900–3905. IEEE, 2020.

Dawei Sun, Jingkai Chen, Sayan Mitra, and Chuchu Fan. Multi-agent motion planning from signal temporal logic specifications. *IEEE Robotics and Automation Letters*, 7(2):3451–3458, 2022.

Zhe Xu and A Agung Julius. Census signal temporal logic inference for multiagent group behavior analysis. *IEEE Transactions on Automation Science and Engineering*, 15(1):264–277, 2016.

Shakiba Yaghoubi and Georgios Fainekos. Worst-case satisfaction of stl specifications using feedforward neural network controllers: a lagrange multipliers approach. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–20, 2019.

Ningyuan Zhang, Wenliang Liu, and Calin Belta. Distributed control using reinforcement learning with temporal-logic-based reward shaping. In *Learning for Dynamics and Control Conference*, pages 751–762. PMLR, 2022.