Secure Planning Against Stealthy Attacks via Model-Free Reinforcement Learning

Alper Kamil Bozkurt, Yu Wang, and Miroslav Pajic

***Abstract*— We consider the problem of security-aware planning in an unknown stochastic environment, in the presence of attacks on control signals (i.e., actuators) of the robot. We model the attacker as an agent who has the full knowledge of the controller as well as the employed intrusion-detection system and who wants to prevent the controller from performing tasks while staying stealthy. We formulate the problem as a stochastic game between the attacker and the controller and present an approach to express the objective of such an agent and the controller as a combined linear temporal logic (LTL) formula. We then show that the planning problem, described formally as the problem of satisfying an LTL formula in a stochastic game, can be solved via model-free reinforcement learning when the environment is completely unknown. Finally, we illustrate and evaluate our methods on two robotic planning case studies.**

I. Introduction

Security is an important concern for robotic systems working in critical applications. Malicious attacks on these systems can happen from various sources, exploiting vulnerabilities in, for instance, system sensing (e.g., GPS [1]–[3] or automotive speed sensors [4]) or software design [5], [6]. To prevent such anomalous behaviors, a common approach is to incorporate an intrusion detection system (IDS) into the overall system design. These components monitor in runtime key parts of the system, raising alarms when unexpected changes or behaviours, potentially caused by attacks, are observed.

Although the use of IDS has significantly enhanced security guarantees in robotics, by limiting available attack vectors, sophisticatedly crafted attacks that are stealthy to IDS (i.e., undetected by the IDS) can still have significant impact on system performance. Stealthy attacks for control systems have been extensively studied. Using knowledge of the system models, various kinds of stealthy attacks have been designed, such as replay attacks (e.g., [7]), covert attacks (e.g., [8]), zero dynamic attacks (e.g., [9]), as well as attacks for specific types of IDSs (e.g., [10]–[12]). Also, without knowledge of the system models (i.e., black/grey-box attacks), machine learning-based methods have been recently used to design stealthy attacks [13]–[15].

In this work, we focus on defending against stealthy actuation attacks (i.e., on control signals) in robotic planning, where the dynamics are typically captured by finite-state probabilistic models. Specifically, we provide a reinforcement learning (RL)-based framework for security-aware design of control strategies that maximizes resilience of the robotic task to adversarial actions. To achieve this, we adopt a game-theoretic approach and capture the interaction between the (high-level) system controller and the attacks as turn-based *stochastic games* [16], which is an extension of Markov decision processes (MDPs) with two players.

The objectives for robotic planning tasks with temporal properties are commonly expressed by formulas in linear temporal logic (LTL) (e.g., [17], [18]). The fulfillment of an LTL objective may not only depend on the current state (e.g., safety) but also on the whole system execution (e.g., visiting a state infinitely often for surveillance). Hence, our security-aware planning focuses on design of control strategies that maximize the worst-case (due to the attacker actions) probability that the given LTL task specification φ_{TASK} is satisfied.

For the security measure, we consider a very general class of logic-based IDSs that monitor the system execution and trigger alarm if a given LTL formula φ_{IDS} is satisfied. This includes the common window-based IDSs that trigger alarm when there are several unexpected system transitions within a time window or all past moves in general [19]–[22]. Following this problem setup, we consider the defense against two types of stealthy attacks with an increasing level of aggressiveness. In Case I, the attacker tries to sabotage the task objective without triggering alarm from the IDS. In Case II, the attacker can take the risk of being detected by IDS if the probability of finally sabotaging the initial LTL objective is maximized. For these types of attacks, we solve the game between the attacker and controller, and derive the optimal defense strategies for the controller.

Furthermore, to allow the use of our security-aware planning methodology for robotic systems with unknown models, we adopt a model-free RL approach to solve the game between the attacker and controller. Although solving stochastic games with temporal logic objectives by model-based methods is well-studied, there are few works on model-free methods that do not depend on the knowledge on the transition probabilities. By exploiting recent results from [23], to the best of our knowledge, we introduce the first model-free RL method for stochastic games that are related to secure planning for LTL objectives. Finally, with two case studies focused on robotic surveillance and task sequencing, we demonstrate applicability of our methodology for security-aware robotic planning.

II. Preliminaries and Problem Statement

A. Stochastic Games

Stochastic games provide a powerful framework to model and reason about behavior of multiple self-interested agents in stochastic environment, capturing both nondeterministic and stochastic transitions. We focus on a scenarios where

This work is sponsored in part by the ONR under agreements N00014-17-1-2504, N00014-20-1-2745 and N00014-18-1-2374, AFOSR award number FA9550-19-1-0169, and the NSF CNS-1652544 grant.

Alper Kamil Bozkurt, Yu Wang, and Miroslav Pajic are with Duke University, Durham, NC 27708, USA, {alper.bozkurt, yu.wang094, miroslav.pajic}@duke.edu

there are only two agents who are strictly competitive (zero-sum), which means if one agent wins, then the other loses. The controller (i.e., Player 1) wants to perform a given task, whereas the attacker (i.e., Player 2) wants to prevent that.

Definition 1 (Stochastic Games): A (labeled turn-based) two-player stochastic game is a tuple $\mathcal{G} = (S, (S_\mu, S_\nu, S_p), A, P, s_0, \text{AP}, L)$, where $S = S_\mu \cup S_\nu \cup S_p$ is a finite set of states, with S_μ, S_ν and S_p being disjoint; S_μ and S_ν are the sets of states where the controller or the attacker, respectively, choose actions; S_p is the set of stochastic states; $s_0 \in S$ is an initial state; A is a finite set of actions and $A(s)$ denotes the set of actions that can be taken by the controller or the attacker if $s \in S_\mu$ and $s \in S_\nu$ respectively, and denotes a single dummy action for all $s \in S_p$; $P : S \times A \times S \to [0, 1]$ is a transition probability function such that $P(s, a, s') \in \{0, 1\}$ for $s \in S_\mu \cup S_\nu$, and $\sum_{s' \in S} P(s, a, s') = 1$ if $a \in A(s)$, and 0 otherwise for all $s \in S$; AP is a finite set of atomic propositions; and $L : S \to 2^{\text{AP}}$ is a labeling function.

In a stochastic game $\mathcal{G}$, the successor of a state s is chosen by the controller if $s \in S_\mu$ or by the attacker if $s \in S_\nu$; and if $s \in S_p$, it is randomly chosen according to the probability distribution $P(s, A(s), \cdot)$. A *path* $\sigma = s_0 s_1, \ldots$ is the infinite sequence of the visited states. For simplicity, we denote the state s_t by $\sigma[t]$ and the suffix $s_t s_{t+1} \ldots$ by $\sigma[t{:}]$.

The strategies of the controller and attacker are captured by a function that maps a finite prefix, i.e., a history of visited states, to a probability distribution over the actions that can be taken in the last state. Here, we focus on pure and finite-memory strategies as we will show later (in Sec. IV)) that they suffice for security-aware planning for LTL tasks [24].

Definition 2: A *finite-memory strategy* for a game $\mathcal{G}$ is a tuple $\pi = (M, \Delta, \alpha, m_0)$ where: M is a finite set of modes; $\Delta \colon M \times S \to M$ is a transition function; $\alpha \colon M \times S \setminus S_p \to A$ is a function that maps the current mode $m \in M$ and state s to an action in $A(s)$; and m_0 is an initial mode. A *controller strategy* μ is a finite-memory strategy that only maps states in S_μ to actions (i.e., $\alpha : M \times S_\mu \to A$). Similarly, an *attacker strategy* ν is a finite-memory strategy where $\alpha : M \times S_\nu \to A$.

An optimal controller strategy is defined as a strategy under which the probability that the controller successfully performs a given task is maximized in the worst-case.

B. Capturing Temporal Specifications

LTL extends the propositional logic with temporal modalities: next ($\bigcirc$) and until (U). LTL formulas are constructed via a recursive combinations of Boolean operators and temporal modalities using the following syntax [25]:

$$\varphi \coloneqq \text{true} \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2, \ a \in \text{AP}, \quad (1)$$

where AP is a set of atomic propositions.

LTL formulas specify properties of infinite paths of games. In addition to the satisfaction of the standard logical operations, the fulfillment of an LTL formula φ on a path σ of game $\mathcal{G}$, denoted by $\sigma \models \varphi$, is recursively defined as: σ satisfies an atomic proposition a, if $a \in L(\sigma[0])$; σ satisfies $\bigcirc\varphi$ if $\sigma[1{:}]$ satisfies φ; and finally, $\sigma \models \varphi_1 \mathsf{U} \varphi_2$, if $\exists i.\ \sigma[i] \models \varphi_2$ and $\forall j < i.\ \sigma[j] \models \varphi_1$. Also, we write (eventually) $\Diamond\varphi$ for $\text{true}\ \mathsf{U}\ \varphi$, (always) $\Box\varphi$ for $\neg(\Diamond\neg\varphi)$, and

$$\Diamond^{\leq k}\varphi \coloneqq \bigcirc\varphi \vee \bigcirc\bigcirc\varphi \vee \cdots \vee \underbrace{\bigcirc\bigcirc\cdots\bigcirc}_{k \text{ times}}\varphi. \quad (2)$$

Deterministic Rabin automata (DRAs) offers a systematic way for LTL model checking. An LTL formula can be represented by a DRA that accepts a path if and only if the path satisfies the LTL formula; the acceptance conditions of DRAs are defined for infinite paths [25].

Definition 3: A DRA is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, \text{Acc})$ where Q is a finite set of states; Σ is a finite alphabet; $\delta : Q \times \Sigma \to Q$ is the transition function; $q_0 \in Q$ is an initial state; Acc is a set of k accepting pairs $\{(C_i, B_i)\}_{i=1}^{k}$ such that $C_i, B_i \subseteq Q$ such that an infinite path σ is accepted if

$$\exists i : \ \text{inf}(\sigma) \cap C_i = \varnothing \ \wedge \ \text{inf}(\sigma) \cap B_i \neq \varnothing, \quad (3)$$

where $\text{inf}(\sigma)$ is the set of states visited infinitely often during the execution induced by the labels of σ.

A pair (μ, ν) of a controller strategy μ and an attacker strategy ν in a game $\mathcal{G}$ induces a Markov chain (MC), denoted by $\mathcal{G}_{\mu,\nu}$. The probability that a path σ sampled from an MC $\mathcal{G}_{\mu,\nu}$ satisfies the LTL formula φ is defined as:

$$Pr_{\mu,\nu}(\mathcal{G} \models \varphi) \coloneqq Pr_{\sigma \sim \mathcal{G}_{\mu,\nu}}\{\sigma \mid \sigma \models \varphi\}. \quad (4)$$

The objective in to find a controller strategy μ_φ such that

$$\mu_\varphi = \text{argmax}_\mu \min_\nu Pr_{\mu,\nu}(\mathcal{G} \models \varphi) \quad (5)$$

where μ and ν denote any finite-memory controller and attacker strategies, respectively. Note that μ_φ might not be unique; in that case, slightly abusing the notation, we let μ_φ to denote the set of all such controller strategies.

C. Problem Statement: Security-Aware Planning

This work is motivated by the reported susceptibility of autonomous systems to attacks [1], [3], [26]. Several methods are proposed to deal with attacks on system sensing, both for low-level control (e.g., [27]–[29]), and path planning (e.g., [30], [31]). However, to the best of our knowledge, no methods exist to design controller strategies for uncertain environments, such that the derived controllers are maximally resilient to actuation attacks – this is the focus of this work.

Specifically, we consider scenarios where a smart attacker can modify control actions with the goal of preventing the robot from performing the given task, captured by an LTL objective φ_{TASK}. We assume that the system has an Intrusion Detection System (IDS) used to detect system anomalies; thus, the attacker's objective is also to remain *stealthy*. This can be effectively achieved by injecting non-aggressive and incremental control perturbations that exploit probabilistic uncertainties in robot motion (captured by the system model).

Consequently, in this work, we address the problem of *synthesizing security-aware controller strategies* that maximize the worst-case (i.e., even for the most damaging attacker actions) probability of satisfying the LTL mission objectives φ_{TASK}. In addition, we assume that the controller does not know model of the system. Hence, the problem is formulated as a stochastic game $\mathcal{G}$ where the transition probabilities and the topology of the game are unknown, and our goal is to design a *model-free* reinforcement learning (RL)-based framework for synthesis of such security-aware controller strategies. Note that since no attacks on sensing are considered in this work, both the controller and the attacker have the knowledge of the current state of the robot.

In the rest of the paper, we first show how to 'combine' the attacker's stealthiness constraint with the control objective

(Sec. III); this enables us to formulate this problem as a control synthesis problem from such combined LTL formulas for stochastic games. Then, we show how model-free RL can be used to derive such maximally resilient (i.e., security-aware) controller strategies (Sec. IV), by exploiting suitably crafted rewards and discounting based on the LTL formula.

III. LTL Specification of Security-Aware Controller Objectives

In this section, we show how the IDS' triggering condition can be captured by a winning LTL specification for the game.

A. Controller Objective as Winning Condition

The controller's main objective is to successfully perform a given task φ_{TASK}. As with previous works e.g., [32], [33], we capture the task φ_{TASK} by LTL; this includes common planning tasks such as avoidance ($\square\neg\texttt{unsafe}$), liveness/recurrence ($\square\lozenge\phi$), persistence ($\lozenge\square\texttt{safe}$), coverage of other tasks ($\lozenge\phi_1\wedge\lozenge\phi_2\wedge\cdots\wedge\lozenge\phi_n$) or sequencing of other tasks ($\lozenge(\phi_1\wedge\lozenge(\phi_2\wedge\cdots\wedge\lozenge\phi_n)\dots)$).

In addition, we assume that after the attack is detected, the attacker is no longer capable of attacking, and thus additional requirement for the controller can be stated as attack detection φ_{IDS} – we discuss in detail how to construct LTL formula φ_{IDS} in Sec. III-B. Therefore, the winning condition for the controller can be captured in LTL as:

$$\varphi_{\text{WIN}} = \varphi_{\text{IDS}} \vee \varphi_{\text{TASK}}. \tag{6}$$

There are several important implications of φ_{WIN} in (6). First, if the controller can always satisfy φ_{TASK} regardless of being under an attack, then the controller does not need to use an IDS. Similarly, if φ_{IDS} can never be satisfied, the IDS mechanism does not help the controller at all. Second, φ_{IDS} should be satisfied only if there is an actual attack; otherwise, with the above winning condition, the controller has an incentive to look for situations where a false alarm can be raised, in order to win the game. Third, detecting an attack should not be enough to win the game as the controller still needs to be able to perform the task. In addition, the attacker might risk being detected for an attack that may cause the controller to end up in a state from which the task cannot be satisfied. Hence, φ_{IDS} needs to be extended to also cover these cases, as done in the rest of the section.

B. Intrusion Detection in Stochastic Environments

An IDS monitors system evolution, continuously evaluating if the observed behavior deviates from the expected ones. Analysis of several IDS is recently performed in the context of security-aware control (e.g., [10]–[12]). When executing in uncertain environments (e.g., modeled as stochastic games), such IDS commonly provide probabilistic guarantees, and can potentially cause (e.g., fixed-rate) false alarms.

For example, consider a planning scenario where a robot may not move in the intended direction with a probability of at most p_ε. In practice, effectively monitoring such probabilistic behaviors is mapped into counting the number of unintended moves in a pre-specified time-window or in all past moves in general [19]–[22]. Consider e.g., that for every four-time-step window, an alarm should be raised if the count is larger than 1; in this case, even without an attacker, when $p_\varepsilon = 0.1$, the likelihood of a false alarm is 0.0523 and the expected number of steps before an alarm is 47.

For any $p_\varepsilon{>}0$, any such window-based IDS eventually raises alarm, even without attacker. Thus, when considering large-enough (i.e., infinite) paths, if a false alarm is considered as an attacker being detected, then the controller wins the game by satisfying φ_{IDS} and thereby φ_{WIN}, even when the attacker is inactive. On the other hand, to deal with the false alarms in practice, the system would perform a thorough inspection on the next steps (with a significant overhead) such that any attacks are detected based on the employed attack vectors (i.e., actual attack surfaces [34], [35]). If the IDS does not detect any attack for a pre-determined number of time steps during the close inspection, the system goes back to its default execution mode.

Consequently, following [19]–[22], we provide an LTL formulation that can capture a large class of IDSs that satisfy the aforementioned constraints. Consider the following window-based IDS described by the LTL formula:

$$\varphi_{\text{ALARM}} = \lozenge\big(\texttt{anomaly}\wedge\bigcirc\lozenge^{\leq m}\texttt{anomaly}\big) \tag{7}$$

where $\texttt{anomaly}$ is the label of unexpected executions. The alarm formula φ_{ALARM} can be interpreted as: *raise an alarm if an anomaly occurs and after that another one occurs within the next $m+1$ time steps*. Such formula can be easily extended to count/allow for more anomalies within a different window size, by adding nested $\bigcirc\lozenge^{\leq m_i}$, as in e.g., $\varphi_{\text{ALARM}_2}{=}\lozenge(\texttt{anomaly}\wedge\bigcirc\lozenge^{\leq m_1}(\texttt{anomaly}\wedge\bigcirc\lozenge^{\leq m_2}\texttt{anomaly}))$ We can also count consecutive anomalies by setting some m_i to zeros. Many IDS mechanisms can be inherently described as a reachability property, which can be specified by the logical operators and temporal modalities $\bigcirc$, $\lozenge$ and $\lozenge^{\leq k}$.

After an alarm occurs, the system switches to a high-alert mode and can detect any new attacks. This is captured by

$$\varphi_{\text{DETECT}} = \lozenge^{\leq n}\texttt{attack}, \tag{8}$$

where $\texttt{attack}$ is the label of any transition considered as an attack by the IDS. The detection formula φ_{DETECT} simply means that any attack within $n{+}1$ time steps will be detected.

We can integrate a detection formula into an alarm formula to obtain a combined IDS formula. This is achieved by adding $\wedge\bigcirc\varphi_{\text{DETECT}}$ next to the last $\texttt{anomaly}$ in the alarm formula that triggers the alarm. For example, if we integrate φ_{DETECT} in (8) to φ_{ALARM} in (7), we obtain:

$$\varphi_{\text{IDS}} = \lozenge(\texttt{anomaly}\ \wedge\bigcirc\lozenge^{\leq m}(\texttt{anomaly}\wedge\bigcirc\lozenge^{\leq n}\texttt{attack})).$$

Note that the negation of φ_{IDS} nicely reflects the behavior of a stealthy attacker. An attacker with the objective $\neg\varphi_{\text{IDS}}$ wants to stay undetected at all costs and is reluctant to frequently take actions to avoid triggering alarm, so that the attacks are 'hidden' within the stochastic behavior of the environment.

One problem with this formulation is that it does not express the cost of carrying out the close system inspection. This, unfortunately, gives an incentive for the controller to trigger a false alarm. For example, the controller may try to move to the states that exhibit a high degree of randomness to increase its chance to switch to the high-alert mode, after which it can continue to perform its task without the fear of being under attack for the predefined number of steps. We discuss a strategy to overcome this problem in Sec. IV. Furthermore, even if the attacker is detected, the robot still

needs to perform its task; we address this as follows.

C. Performing Tasks After Attack Detection

The attacker's goal is to prevent satisfaction of the control task. Thus, it may be beneficial for the attacker to launch an attack even during the high-alert mode (i.e., at the cost of being detected and eliminated), if the controller could no longer fulfill its task after the attack.

This can be embedded into the previously described LTL formula φ_{IDS} by replacing $\texttt{attack}$ in φ_{IDS} with $\texttt{attack} \wedge \bigcirc\Diamond\texttt{attack}$. For example, the previous φ_{IDS} becomes $\varphi_{\text{IDS}}{=}\Diamond\big(\texttt{anomaly}\wedge\bigcirc\Diamond^{\leq m}\big(\texttt{anomaly}\wedge\bigcirc\Diamond^{\leq n}\big(\texttt{attack}\wedge\bigcirc\Diamond\texttt{attack}\big)\big)\big)$.

Although this might seem as giving a second chance to the attacker after being detected, (s)he cannot attack for a second time because that would result in the attacker immediately losing the game. Another concern with this formulation is that the IDS must be in the high-alert mode all the time after the first attack to be able to observe a second attack; however, this is not necessary because the second attack cannot happen in practice based on the assumption that the attacker is eliminated after the first attack. Hence, this IDS formulation allows the scenarios where the attacker is detected but might still win the game if the controller fails to perform its task.

IV. Model-Free Learning from the Winning LTL Specifications

In this section, we use model-free reinforcement learning to find optimal controller strategies that maximize the (worst-case) probability of satisfying the φ_{WIN} from (6) – i.e.,

$$\mu_{\varphi_{\text{WIN}}} = \text{argmax}_{\mu} \min_{\nu} Pr_{\mu,\nu}(\mathcal{G} \models \varphi_{\text{WIN}}). \tag{9}$$

Our method requires no knowledge of the transition probabilities or the topology of the game $\mathcal{G}$, which models the interaction between the controller and the attacker.

A. From LTL to Discounted Rewards for Model-Free RL

Model-free RL provides an efficient way to search for optimal strategies without reconstructing the explicit model (e.g., the transition probabilities) of the stochastic game from samples. However, existing model-free RL methods can only be used for cumulative rewards associated with the states (or transitions) and cannot be directly used for LTL objectives on stochastic games. Thus, we convert the LTL formula φ_{WIN} into a cumulative discounted reward below.

Let $G(\sigma)$ denote the return, the sum of discounted rewards of a path σ of $\mathcal{G}$, defined as:

$$G(\sigma) \coloneqq \sum\nolimits_{i=0}^{\infty} \left(\prod\nolimits_{j=0}^{i-1} \Gamma(\sigma[j]) \right) \cdot R(\sigma[i]); \tag{10}$$

here, $R : S \to [0, 1)$ and $\Gamma : S \to (0, 1)$ are the state-based reward and discount functions. We make the convention that $\prod_{j=0}^{-1} \coloneqq 1$. The goal of model-free RL is to find a controller strategy that maximizes the expected return in the worst case:

$$\mu_* = \text{argmax}_{\mu} \min_{\nu} \mathbb{E}_{\sigma\sim\mathcal{G}_{\mu,\nu}} [G(\sigma)]. \tag{11}$$

Our goal, now, becomes to design the functions R and a Γ in such a way that μ_* and $\mu_{\varphi_{\text{WIN}}}$ from (9) become equal.

Recently, there has been an increasing interest in developing of such model-free RL methods [23], [36]–[38]. Here, we adopt the method introduced in [23], which takes a turn-based stochastic game $\mathcal{G}$ where P is fully unknown, an LTL specification of the winning condition φ_{WIN}, a discount factor γ, and learns the controller strategy $\mu_{\varphi_{\text{WIN}}}$. For the discount factors sufficiently close to 1, if the LTL formula can be translated into a DRA with one accepting pair, the RL algorithm is guaranteed to converge to an optimal controller strategy, otherwise it converges to a controller strategy with a satisfaction probability above an established lower bound.

Algorithm 1: Model-free RL for security-aware control synthesis from LTL specifications.

Input: LTL formula φ_{WIN}, stochastic game $\mathcal{G}$
Translate φ_{WIN} to a DRA $\mathcal{A}_{\varphi_{\text{WIN}}}$
Construct the product $\mathcal{G}^{\times}$ of $\mathcal{G}$ and $\mathcal{A}_{\varphi}$
Learn the optimal state-action values Q_* for $\mathcal{G}^{\times}$ using minimaxQ
Obtain a greedy strategy $\mu_*^{\times}$ from Q_*
return a finite-memory controller strategy μ_* derived from $\mu_*^{\times}$

The idea is to augment the state space such that it is sufficient to consider only the pure and memoryless strategies so that a state-based R and Γ can be defined. Specifically, using the method from [23], we construct a product game from the original game $\mathcal{G}$ and the DRA $\mathcal{A}_{\varphi_{\text{WIN}}}$ obtained from the LTL formula φ_{WIN}, thus reducing the winning criteria to the satisfaction of the Rabin acceptance condition from (3).

Definition 4 (Product Games): A product game of $\mathcal{G} = (S, (S_\mu, S_\nu, S_p), A, P, s_0, \text{AP}, L)$, a labeled turn-based stochastic game, and a DRA $\mathcal{A} = (Q, 2^{\text{AP}}, \delta, q_0, \text{Acc})$, is the tuple $\mathcal{G}^{\times} = (S^{\times}, (S_\mu^{\times}, S_\nu^{\times}, S_p^{\times}), A^{\times}, P^{\times}, s_0^{\times}, \text{Acc}^{\times})$ where:

- $S^{\times} = S \times Q$ is the set of augmented states, and the initial state $s_0^{\times}$ is $\langle s_0, q_0 \rangle$;
- $S_\mu^{\times} = S_\mu \times Q$ is the set of augmented controller states;
- $S_\nu^{\times} = S_\nu \times Q$ is the set of augmented attacker states;
- $S_p^{\times} = S_p \times Q$ is the set of augmented stochastic states;
- $A^{\times} = A$ is the set of actions where $A^{\times}(\langle s, q \rangle) = A(s)$ for all $s \in S, q \in Q$;
- $P^{\times} : S^{\times} \times A^{\times} \times S^{\times} \to [0, 1]$ is the transition function:
$$P^{\times}(\langle s, q \rangle, a, \langle s', q' \rangle) = \begin{cases} P(s, a, s') & \text{if } q' {=} \delta(q, L(s)) \\ 0 & \text{otherwise} \end{cases}$$
- $\text{Acc}^{\times}$ is a set of k accepting pairs $\{(C_i^{\times}, B_i^{\times})\}_{i=1}^{k}$ where $C_i^{\times} = C_i \times Q$ and $B_i^{\times} = B_i \times Q$.

When there is only a single pair Rabin in the acceptance condition (i.e., $\text{Acc} = \{(C, B)\}$), there is always a pure and memoryless optimal strategies for both the controller and the attacker. This allows to construct simple R and Γ functions in (10). For example, a small positive reward can be received whenever a state in B visited since some states in B need to be visited infinitely often to win the game. Similarly, to discourage visiting the states in C infinitely many times, future rewards can be heavily discounted. Consequently, Algorithm 1 summarizes the steps of our approach.

Proposition 1: Consider a stochastic game $\mathcal{G}$ and an LTL formula φ_{WIN} that can be translated to a DRA with a single accepting pair. There exists γ' such that for all $\gamma \in (\gamma', 1)$, Algorithm 1 converges and returns a pure finite-memory controller strategy satisfying (9), if the reward R_γ and the discount Γ_γ functions are defined as:

$$R_\gamma(s^{\times}) \coloneqq \begin{cases} 1{-}\gamma_B(\gamma), & s^{\times} \in B^{\times} \\ 0, & s^{\times} \notin B^{\times} \end{cases}, \quad \Gamma_\gamma(s^{\times}) \coloneqq \begin{cases} \gamma_B(\gamma), & s^{\times} \in B^{\times} \\ \gamma_C(\gamma), & s^{\times} \in C^{\times} \\ \gamma, & \text{otherwise} \end{cases}$$

where γ_B and γ_C are functions of γ satisfying

$$\lim_{\gamma \to 1^-} \frac{1-\gamma}{1-\gamma_B(\gamma)} = \lim_{\gamma \to 1^-} \frac{1-\gamma_B(\gamma)}{1-\gamma_C(\gamma)} = 0. \quad (12)$$

Proof: The proof immediately follows from the proof of Theorem 1 in [23], showing that under a strategy pair, as $\gamma \to 1^-$, the expected return of a path goes to the probability that the path satisfies the Rabin acceptance condition. Finally, the obtained strategy in the product game satisfying (11), induces a finite-memory controller strategy (in the initial game $\mathcal{G}$) satisfying (9), where the the modes are the states of the DRA $\mathcal{A}_{\varphi_{\text{WIN}}}$ with the same transition function. ■

Algorithm 1 can be generalized to situations where the DRA $\mathcal{A}_{\varphi_{\text{WIN}}}$ has multiple Rabin pairs, using the approach from [23]; this results in controller strategies with a lower bound on the satisfaction probabilities (see [23] for details).

B. Efficiency of Learning Controller Strategies

The size of the DRA $\mathcal{A}_{\varphi_{\text{WIN}}}$ may be double-exponential in the length of φ_{WIN}. This cannot be prevented in the worst case if φ_{TASK} is any arbitrary LTL formula. To overcome this, we can restrict the controller tasks to a fragment of LTL, such as Generalized Rabin(1) [39], which can describe most commonly used robotic tasks, as well as be efficiently translated into a polynomial-size DRA with a single accepting pair.

Fortunately, any valid IDS mechanism can be expressed as a reachability property, which means if a path satisfies the property, then there must be a prefix of the path such that all the paths having the same prefix also satisfy the property. Consider a path σ, an infinite execution of the game, for which the IDS triggers the alarm – i.e., $\sigma \models \varphi_{\text{IDS}}$. If a finite prefix of σ is not enough for the IDS to decide, then even if it can be inferred that adversarial actions have occurred, the IDS could not trigger the alarm in a finite number of steps.

The reachability properties form only a small fragment of LTL for which efficient translation to a deterministic finite automata is usually possible [40]. We can think of such a DFA as a DRA with one accepting state where all the outgoing transitions are self-loops. Therefore, the disjunction of φ_{IDS} with φ_{TASK} in (6) only linearly increases the state space of the product game in the length of $\mathcal{A}_{\varphi_{\text{IDS}}}$. Additionally, we can utilize the fact that all the incoming transitions to the accept state in the DRA of φ_{IDS} are triggered by an action of the attacker. Thus, during the learning phase, whenever a state having a transition to the accepting state is reached, the attacker turn can be skipped.

Finally, we discuss the effect of choosing different discount factors. As the discount factor γ goes to 1, the convergence rate as well as the stability of the RL algorithms decrease. Hence, we start with a smaller γ and increase it slowly until the RL algorithm converges to a desired controller strategy. Using smaller γ also discourages the controller to wait for a false alarm to be triggered, and thus mitigates the problem discussed in Sec. III-B.

V. Case Studies

For the case studies, we use the *CSRL* tool [41] based on [23]. *CSRL* takes a 2-D labeled grid as a representation of the stochastic game with an LTL formula, and returns a finite-memory controller strategy obtained using the minimax-Q [42] method that follows ϵ-greedy strategies while learning.

A. System Models

We initialized the values of ϵ (the parameter for ϵ-greedy strategy) and α (the learning rate) to 0.5, and slowly decreased them to 0.05 during learning. We set the discount factor for each case study to be γ=0.999, and obtained the controller strategies after 512K episodes, each starting from a random state and stopping after 1K time steps. We used the IDS formula specified in Sec. III-C with m=0 and n=1:
$\varphi_{\text{IDS}}^{(*)}$=$\Diamond(\texttt{anomaly} \wedge \bigcirc(\texttt{anomaly} \wedge \bigcirc \Diamond^{\leq 1}(\texttt{attack} \wedge \bigcirc \Diamond \texttt{attack})))$

We used (7×9) grid worlds for our planning case studies. A robot can move from a cell to an adjacent cell using the controller actions: *North, South, East* and *West*. However, the attacker can observe any action chosen by the controller and replace it with another action. The actions of the controller and attacker are depicted as black and red arrows in the figures, respectively. Due the stochastic environment, the robot moves in the intended direction with probability of 0.8, and sideways with probability of 0.2 (0.1 for each side). The robot stays in the cell, when it tries to move beyond the grid.

A grid cell corresponds to a controller state in the stochastic game. The label of a controller state is displayed in a small circle, in the lower part of the corresponding cell in the figures. After the controller takes an action, the game moves to an attacker state. The attacker observes this transition and either takes the same action or another one, and then the game moves to a stochastic state. For every cell, the controller and attacker actions, we have a different stochastic state in the game. Thus, every stochastic state has a unique parent attacker state and a unique grandparent control state.

A stochastic state is labeled with `attack` if and only if the transition from its grandparent state to its parent state, and the one from its parent to itself, are triggered by different actions, i.e., *the attacker modifies the controller action*. Even if the `attack` labels are visible to the controller, it acts as if they do not exist unless an alarm is triggered; due to the structure of the IDS formula φ_{IDS} from Sec. III-B, the DRA of φ_{IDS} changes its state by an `attack` label only after an alarm.

In a stochastic state, a random transition is made according to the described transition model, to a dummy state first, then to the controller state of the reached cell. The dummy state is labeled with `anomaly` if the controller state it is connected to is not expected to be reached after the action taken in the great-grandparent controller state. For example, if the controller takes *North* and the robot moves east, the dummy state visited is labeled with `anomaly`. We did not explicitly represent the dummy states to keep the learning completely model-free; it is enough to make the corresponding DRA transition as if an `anomaly` label is consumed whenever such stochastic transitions occurs during learning.

B. Case Study I: Surveillance

In this study, the robot tries to repeatedly cover the cells labeled with b and c. In addition, after a certain point in time, the robot aims to stay in the *safe* region labeled with d, i.e.,

$$\varphi_{\text{TASK}}^{(1)} = \Box\Diamond b \wedge \Box\Diamond c \wedge \Diamond\Box d. \quad (13)$$

The learned strategies are shown Fig. 1. For simplicity, we consider only the part of the task where the robot needs to go from b to c, but similar results were also obtained for travelling from c to b. Although there is a very short

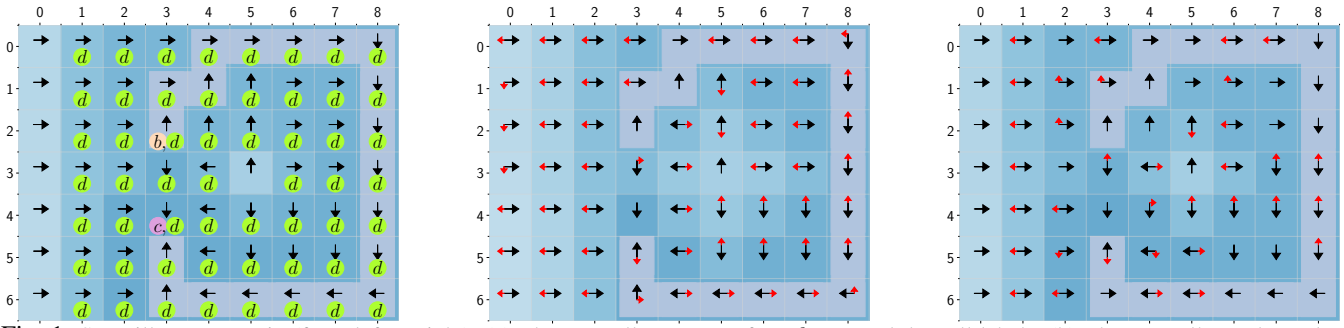

Fig. 1: Surveillance scenario (from left to right): (a) The controller strategy from b to c and the cell labels; (b) The controller and attacker strategies from b to c before any anomaly occurs; (c) The controller and attacker strategies from b to c after one anomaly.

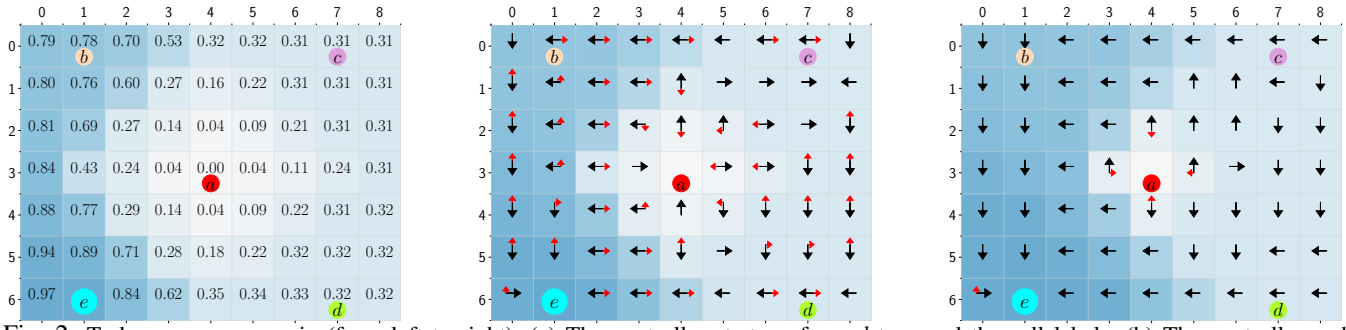

Fig. 2: Task sequence scenario (from left to right): (a) The controller strategy from d to e and the cell labels; (b) The controller and attacker strategies from d to e right after an anomaly occurs; (c) The controller and attacker strategies from d to e right after an alarm.

path from b to c, the learned controller strategy prefers a quite long path. There is only one cell between b and c, and this cell and all the surrounding cells are in the safe region; however, this is not enough to make the path passing through it secure, because once this cell is visited, the attacker can take two consecutive *East* actions to make the robot visit an unsafe cell with a probability of 0.64. Thus, the robot would be out of the safe region once in a while, violating φ_{TASK}.

Fig. 1b and 1c show the attacker strategies before an anomaly happens and right after the alarm. In three out of the first four cells on the most likely path, the attacker chooses to do nothing (Fig. 1b). The reason is that the attacker does not want to create an unnecessary anomaly in the early part of the path, 'saving', in some sense, for the future the ability to create two consecutive anomalies without raising alarm. The attacker strategy in the other parts of the grid forces the robot outside the safe region and prevents it from reaching c.

C. Case Study II: Sequence of Tasks

In this scenario, we plan for a sequence of tasks represented by the labels b, c, d, e, to be performed in order. There is a danger zone labeled with a, to be avoided – i.e.,

$$\varphi^{(2)}_{\text{TASK}} = \Diamond\Big(b \wedge \Diamond\big(c \wedge \Diamond(d \wedge \Diamond e)\big)\Big) \wedge \Box\neg a. \qquad (14)$$

Here, we present the results and the strategies for performing only the last task e, i.e., visiting the cell e; however, similar conclusions can be drawn for the other tasks. Fig. 2a shows the estimated satisfaction probabilities when the IDS has not detected any anomalies. As expected, the probabilities near the danger zone are very low and the probabilities are usually getting lower as the distance to e is increasing.

The satisfaction probabilities in the right part of the grid are significantly lower than the left part. The reason is that while moving from the right part to the left, the minimum number of cells between the robot and a can be at most two, e.g., at $(6, 4)$, which allows the attacker to take three consecutive actions, e.g., $3\times$*North* from $(6, 4)$, to drag the robot into the danger zone with a probability larger than 0.5. If all the attacks are successful, i.e., the robot moves in the direction the attacker desired, after the first two actions the IDS raises an alarm. After the alarm, although the attacker knows (s)he will be detected, (s)he attacks again if the robot is next to a, as there is a high probability (0.8) that the robot moves into the danger zone, making the attacker the winner.

Fig. 2b shows the controller and attacker strategies after an anomaly occurs. Again, the attacker is more passive in the upper right part of the grid, as (s)he does not want to trigger an alarm when the controller is far from reaching e. Fig. 2c shows the strategies when the IDS is in the high-alert mode, in which any attack is immediately detected. The attacker takes an action only in five cells. In the cell at the bottom-left corner, (s)he makes one last attempt to prevent the controller from reaching e because even if (s)he is detected, if (s)he does nothing the controller wins the game with a probability slightly lower than 1. The other cells are the ones next to the danger zone, where the attacker sacrifices stealthiness for the high probability that the robot ends up in the danger zone.

VI. Conclusion

We studied planning for temporal logic tasks in an unknown stochastic environment in the presence of actuation attacks. We formulated the interaction between the controller and stealthy attacker as a stochastic game, where the attacker, knowing the intrusion detection systems (IDS), the task, and the controller; aims to undermine the task while remaining stealthy. We then show this planning problem can be solved using model-free reinforcement learning without knowledge of the environment model. Our case studies showed the applicability of our method for security-aware robotic planning.

REFERENCES

[1] Andrew J. Kerns, Daniel P. Shepard, Jahshan A. Bhatti, and Todd E. Humphreys. Unmanned aircraft capture and control via GPS spoofing. *Journal of Field Robotics*, 31(4):617–636, 2014.

[2] Mark L. Psiaki, Todd E. Humphreys, and Brian Stauffer. Attackers can spoof navigation signals without our knowledge. Here's how to fight back GPS lies. *IEEE Spectrum*, 53(8):26–53, 2016.

[3] Cheolhyeon Kwon, Scott Yantek, and Inseok Hwang. Real-time safety assessment of unmanned aircraft systems against stealthy cyber attacks. *Journal of Aerospace Information Systems*, 13(1):27–45, 2016.

[4] Yasser Shoukry, Paul Martin, Paulo Tabuada, and Mani Srivastava. Non-invasive spoofing attacks for anti-lock braking systems. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 55–72. Springer, 2013.

[5] Gilles Barthe, Pedro R. D'Argenio, Bernd Finkbeiner, and Holger Hermanns. Facets of software doping. In *International Symposium on Leveraging Applications of Formal Methods*, pages 601–608. Springer, 2016.

[6] Abdullahi Chowdhury, Gour Karmakar, and Joarder Kamruzzaman. Survey of recent cyber security attacks on robotic systems and their mitigation approaches. In *Cyber Law, Privacy, and Security: Concepts, Methodologies, Tools, and Applications*, pages 1426–1441. IGI Global, 2019.

[7] Yilin Mo and Bruno Sinopoli. Secure control against replay attacks. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing*, pages 911–918, 2009.

[8] Roy S. Smith. Covert Misappropriation of Networked Control Systems: Presenting a Feedback Structure. *IEEE Control Systems Magazine*, 35(1):82–92, 2015.

[9] Andre Teixeira, Iman Shames, Henrik Sandberg, and Karl H. Johansson. Revealing stealthy attacks in control systems. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing*, pages 1806–1813, Monticello, IL, USA, October 2012. IEEE.

[10] Yilin Mo and Bruno Sinopoli. False data injection attacks in control systems. In *First workshop on Secure Control Systems*, pages 1–6, 2010.

[11] Cheolhyeon Kwon, Weiyi Liu, and Inseok Hwang. Analysis and design of stealthy cyber attacks on unmanned aerial systems. *Journal of Aerospace Information Systems*, 11(8):525–539, 2014.

[12] Ilija Jovanov and Miroslav Pajic. Relaxing integrity requirements for attack-resilient cyber-physical systems. *IEEE Transactions on Automatic Control*, 64(12):4843–4858, Dec 2019.

[13] Jiangnan Li, Jin Young Lee, Yingyuan Yang, Jinyuan Stella Sun, and Kevin Tomsovic. ConAML: Constrained Adversarial Machine Learning for Cyber-Physical Systems. *arXiv:2003.05631 [cs]*, March 2020.

[14] Giulio Zizzo, Chris Hankin, Sergio Maffeis, and Kevin Jones. Adversarial Machine Learning Beyond the Image Domain. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, pages 1–4, Las Vegas, NV, USA, June 2019. Association for Computing Machinery.

[15] Cheng Feng, Tingting Li, Zhanxing Zhu, and Deeph Chana. A deep learning-based framework for conducting stealthy attacks in industrial control systems. *arXiv:1709.06397 [cs]*, September 2017.

[16] Lloyd S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953.

[17] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Where's Waldo? sensor-based temporal logic motion planning. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3116–3121. IEEE, 2007.

[18] Meng Guo, Karl H. Johansson, and Dimos V. Dimarogonas. Revising motion planning under linear temporal logic specifications in partially known workspaces. In *2013 IEEE International Conference on Robotics and Automation*, pages 5025–5032. IEEE, 2013.

[19] Borzoo Bonakdarpour, Jyotirmoy V. Deshmukh, and Miroslav Pajic. Opportunities and challenges in monitoring cyber-physical systems security. In *International Symposium on Leveraging Applications of Formal Methods*, pages 9–18. Springer, 2018.

[20] Ramy Medhat, Borzoo Bonakdarpour, Deepak Kumar, and Sebastian Fischmeister. Runtime monitoring of cyber-physical systems under timing and memory constraints. *ACM Transactions on Embedded Computing Systems (TECS)*, 14(4):1–29, 2015.

[21] Klaus Havelund and Grigore Roşu. Synthesizing monitors for safety properties. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 342–356. Springer, 2002.

[22] Moonjoo Kim, M. Viswanathan, H. Ben-Abdallah, S. Kannan, I. Lee, and O. Sokolsky. Formally specified monitoring of temporal properties. In *Proceedings of 11th Euromicro Conference on Real-Time Systems. Euromicro RTS'99*, pages 114–122, June 1999.

[23] Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic. Model-free reinforcement learning for stochastic games with linear temporal logic objectives, 2020. arXiv:2010.01050 [cs.RO].

[24] Krishnendu Chatterjee and Thomas A. Henzinger. A survey of stochastic ω-regular games. *Journal of Computer and System Sciences*, 78(2):394 – 413, 2012. Games in Verification.

[25] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, Cambridge, MA, USA, 2008.

[26] Miroslav Pajic, James Weimer, Nicola Bezzo, Oleg Sokolsky, George J. Pappas, and Insup Lee. Design and implementation of attack-resilient cyberphysical systems: With a focus on attack-resilient state estimators. *IEEE Control Systems Magazine*, 37(2):66–81, April 2017.

[27] Miroslav Pajic, Insup Lee, and George J. Pappas. Attack-resilient state estimation for noisy dynamical systems. *IEEE Transactions on Control of Network Systems*, 4(1):82–92, March 2017.

[28] Nicola Bezzo, James Weimer, Miroslav Pajic, Oleg Sokolsky, George J. Pappas, and Insup Lee. Attack resilient state estimation for autonomous robotic systems. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3692–3698, Sept 2014.

[29] Young Hwan Chang, Qie Hu, and Claire J. Tomlin. Secure estimation based Kalman filter for cyber–physical systems against sensor attacks. *Automatica*, 95:399–412, 2018.

[30] Mahmoud Elfar, Yu Wang, and Miroslav Pajic. Security-aware synthesis using delayed-action games. In *Computer Aided Verification (CAV)*, pages 180–199. Springer International Publishing, 2019.

[31] Mahmoud Elfar, Haibei Zhu, Mary L. Cummings, and Miroslav Pajic. Security-aware synthesis of human-UAV protocols. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8011–8017, May 2019.

[32] Georgios E. Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, February 2009.

[33] Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. Synthesis for Robots: Guarantees and Feedback for Robot Behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):211–236, 2018.

[34] Vuk Lesi, Ilija Jovanov, and Miroslav Pajic. Network scheduling for secure cyber-physical systems. In *2017 IEEE Real-Time Systems Symposium (RTSS)*, pages 45–55, Dec 2017.

[35] Monowar Hasan, Sibin Mohan, Rakesh B. Bobba, and Rodolfo Pellizzoni. Exploring opportunistic execution for integrating security into legacy hard real-time systems. In *2016 IEEE Real-Time Systems Symposium (RTSS)*, pages 123–134. IEEE, 2016.

[36] Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10349–10355. IEEE, 2020.

[37] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular objectives in model-free reinforcement learning. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 395–412. Springer, 2019.

[38] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Model-free reinforcement learning for stochastic parity games. In *31st International Conference on Concurrency Theory (CONCUR 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[39] Rüdiger Ehlers. Generalized Rabin(1) synthesis with applications to robust system synthesis. In *NASA Formal Methods Symposium*, pages 101–115. Springer, 2011.

[40] Timo Latvala. Efficient model checking of safety properties. In *International SPIN Workshop on Model Checking of Software*, pages 74–88. Springer, 2003.

[41] CPSL@Duke. CSRL, 2020. `https://gitlab.oit.duke.edu/cpsl/csrl`.

[42] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.