Synthesis of optimal multi-objective attack strategies for controlled systems modeled by probabilistic automata

Rômulo Meira-Góes, Raymond H. Kwong, Stéphane Lafortune

***Abstract*— We study the security of control systems in the context of the supervisory control layer of stochastic discrete-event systems. Control systems heavily rely on correct communication between the plant and the controller. In this work, we consider that such communication is partially compromised by a malicious attacker. The attacker has the ability to modify a subset of the sensor readings and mislead the supervisor, with the goal of inducing the system into an unsafe state. We consider this problem from the attacker's viewpoint and investigate the synthesis of an attack strategy for systems modeled as probabilistic automata. Specifically, we investigate the synthesis of attack functions constrained by multiple objectives. We proceed in two steps. First, we quantify each attack strategy based on the likelihood of successfully reaching an unsafe state. Based on this quantification, we study the problem of synthesizing attack functions with the maximum likelihood of successfully reaching an unsafe state. Second, we consider the problem of synthesizing attack functions that have the maximum likelihood of successfully reaching an unsafe state while minimizing a cost function, i.e., the synthesis of attack functions is constrained by multiple objectives. Our solution methodology is based on mapping these problems to optimal control problems for Markov decision processes, specifically, a probabilistic reachability problem and a stochastic shortest path problem.**

***Index Terms*— Discrete event systems; Supervisory control; Stochastic systems; Automata; Sensor deception attacks.**

I. Introduction

The correct behavior of control systems depends heavily on correct communication between the plant and the controller. Most works in the existing literature on control with robust communication focus on random communication faults, such as delays, information loss, and so forth. However, in recent years, there has been interest in considering that a "smart" agent, or attacker, could be responsible for communication faults. In this paper, we consider such an attack model, where the goal of the attacker is to induce the controller to steer the system to an unsafe state by altering the communications from the system sensors to the controller. Our analysis is at the supervisory control layer; hence, we adopt a discrete event modeling formalism, where system operation and communications are event-based and the controller is a *supervisor*. In contrast to prior work on sensor deception attacks for Discrete Event Systems (DES), where logical models are used, we model the system as a *Probabilistic Finite-State Automaton*. This allows us to quantify, in a probabilistic sense, attack strategies.

As a consequence of the stochastic control system model that we adopt, it is possible to quantify each attack strategy by the likelihood of reaching an unsafe state of the plant. In this manner, a quantitative measure is introduced in the synthesis problem of attack strategies. First, we investigate the synthesis of an *attack function* that generates the maximum likelihood of reaching an unsafe state. This problem is denoted as the *probabilistic reachability attack function problem*. In this problem, only a set of compromised sensor readings constrains the attacker on how to alter the communication channel between the system sensors and the supervisor. For this reason, we investigate a second problem where the attacker is penalized for each sensor modification. The second investigated problem is the *synthesis of attack functions that satisfy multiple objectives* (multi-objective). The attack function must reach an unsafe state with maximum probability while minimizing a cost function based on the sensor modifications.

Our solution methodology employs results from the area of stochastic control systems, more specifically Markov Decision Processes (MDPs). First, we show how to build the "right" MDP that captures the interaction of the attacker and the controlled system. Next, we show that the solution of the probabilistic reachability attack function problem is reducible to the probabilistic reachability problem in MDPs [1], [2]. Based on the solution of the first problem, we trim the previously constructed MDP to obtain a solution space for the multi-objective attack function problem. Lastly, we show that the solution of the multi-objective attack function problem is reducible to the stochastic shortest path problem in MDPs [1], [2].

The main contributions of this paper are in posing the new multi-objective attack function problem, and in the solution methodology of the two investigated problems. Although reducing supervisory control problems to MDP problems is not new [3], the solution methodology of reducing these two

R.M.G. and S.L. are with the Department of EECS, University of Michigan, MI 48109 USA (e-mail:{romulo,stephane}@umich.edu). Their work is supported in part by US NSF grants CNS-1446298, CNS-1738103 and CNS-1801342.

R.H.K. is with the ECE Department, University of Toronto, Toronto, ON M5S 3G4, Canada (email: kwong@control.utoronto.ca) . His work was supported by the Natural Sciences and Engineering Research Council of Canada (grant RGPIN-2015-04273).

investigated problems in the area of cyber-security in stochastic DES to well-known problems in the area of stochastic control systems is new, to the best of our knowledge. By showing this reduction, we can leverage the relevant theoretical results in stochastic control, along with employing existing software tools for MDPs to compute optimal solutions to these problems.

In recent years, several works addressed problems of cyber-security in the field of DES. We give a very brief overview of the existing work and compare it to our paper. For a more complete review of the area, please see [4]. Of particular relevance to this paper is our previous work in [5]–[7]; we use herein a similar framework for how attacks take place on the communication channel from sensors to the supervisor. In [5]–[7], an attacker has compromised a subset of the sensors and is able to delete sensor readings or insert fictitious ones in the communication channel. The problem investigated is the synthesis of stealthy *sensor deception attacks* for a known *logical* control system. In [7], we also considered the problem of synthesizing *sensor deception attacks* for *stochastic* systems, but only the probabilistic reachability attack function synthesis problem was investigated. We leveraged results from the area of stochastic games [8] to provide a solution methodology in [7]. In this paper, we leverage results from the area of MDPs to provide a solution methodology. Moreover, we also investigate a completely new problem, the multi-objective attack function synthesis problem. The works in [9]–[12] investigate synthesis of attackers but none of them consider stochastic models.

In [13]–[17], the authors develop diagnostic tools to detect when controlled systems are being attacked. Their work is closely related to the work on fault diagnosis in DES, and it applies to both sensor and/or actuator attacks. Moreover, [13], [16], [17] consider stochastic models, while the other works consider logical DES models. Our problem differs from the problems considered in these works since we aim to compute an attack function that successfully causes the system to reach the critical state. Nonetheless, these diagnostic tools could be incorporated into our framework in order to add additional constraints to the attack function synthesis problems.

There is also a vast literature on robust control in DES [18]. However, robustness in this literature is related to communication delays or loss of information [19]–[22], or model uncertainty [23]–[26]. Our work differs intrinsically from the ones above as they treat unreliable communication and uncertainty as "benign" malfunctions instead of malicious attacks. The problem of synthesizing supervisors robust against attacks is investigated in [9], [27]–[30]. These works are complementary to ours since we focus on the attack side (attack functions) while these works focus on the defense side (robust supervisors). Moreover, these previous works only consider logical DES models. The problem of synthesizing supervisors that are robust against actuator deception attacks is investigated in [31], [32].

Our work also shares some similarities with works investigating the opacity property and its enforcement [33]–[38]. In the opacity property, the attacker is normally considered to be an eavesdropper without any means of altering the behavior of the underlying system, i.e., the attacker is a passive entity. Recently, active attackers have been considered for the opacity problem, see [39]. Our work differs from these works on opacity since they study information release properties of the system, whereas our work assesses the impact of an active attacker over physical parts of the controlled system.

It is also important to point out the difference between our work and the work of [37] using edit functions and the work of [35] using supervisory control as a means to enforce opacity. Even though edit functions are similar to our definition of attack functions, their usage is completely different since they are applied in different contexts. Edit functions are used in open-loop systems, i.e., no supervisor is present, whereas attack functions are used in closed-loop systems. The use of supervisory control to enforce opacity is comparable to the works of synthesis of robust supervisors [9], [27], [29]. Lastly, we are not aware of any work that combines the defense mechanism of supervisory control and edit functions, i.e., they are presented as two different and independent techniques to enforce the opacity property.

Our presentation is organized as follows. Section II introduces necessary background on Supervisory Control Theory. The two investigated problems are formulated in Section III. We briefly review necessary concepts about MDPs in Section IV and present two well-known MDP problems: the probabilistic reachability problem and the stochastic shortest path problem. In Section V, we present the results on the synthesis of maximal probability attack functions. Section VI presents the results with respect to the multi-objective problem. An illustrative example is given in Section VII. Finally, we conclude the paper in Section VIII.

II. Modeling of Controlled Systems

A. Supervisory Control

We consider the supervisory layer of a feedback control system, where the uncontrolled system (plant) is modeled as a Deterministic Finite-State Automaton (DFA) in the discrete-event modeling formalism. A DFA is denoted by $G := (X_G, \Sigma, \delta_G, x_{0,G})$, where X_G is the finite set of states, Σ is the finite set of events, $\delta_G : X_G \times \Sigma \to X_G$ is the partial transition function and $x_{0,G}$ is the initial state. The function δ_G is extended, in the usual manner, to the domain $X_G \times \Sigma^*$. The language generated by G is defined by $\mathcal{L}(G) := \{s \in \Sigma^* \mid \delta_G(x_{0,G}, s)!\}$, where ! means that the function is defined for these arguments. The set of feasible events in state $x \in X_G$ is denoted as $\Gamma_G(x) := \{e \in \Sigma \mid \delta(x, e)!\}$.

In the context of supervisory control theory of DES [40], the plant G is controlled by a *supervisor* that dynamically enables/disables events. To model limited actuation capabilities, the event set Σ is partitioned into the sets of controllable and uncontrollable events, Σ_c and Σ_{uc}. Since uncontrollable events cannot be disabled by the supervisor, the supervisor's control decisions are limited to the set $\Gamma := \{\gamma \subseteq \Sigma \mid \Sigma_{uc} \subseteq \gamma\}$. Therefore, a supervisor is a mapping $S : \mathcal{L}(G) \to \Gamma$ defined to satisfy specifications on G, e.g., make a state in G unreachable. The closed-loop behavior of G under supervision of S is denoted by S/G and generates the closed-loop language $\mathcal{L}(S/G)$; see, e.g., [41]. Without loss of generality, we assume

that S is realized by an automaton $R = (X_R, \Sigma, \delta_R, x_{0,R})$, i.e., $S(s) = \Gamma_R(\delta_R(x_{0,R}, s))$. Both notations S and R are used interchangeably hereafter.

For any string $s \in \Sigma^*$, we use the following notation. We denote by $s[i]$ the i^{th} event of s such that $s = s[1]s[2]\dots s[|s|]$, where $|s|$ denotes the length of s. We denote by s^i the i^{th} prefix of s, namely $s^i = s[1]s[2]\dots s[i]$ and $s^0 = \epsilon$. Moreover, we denote by $\bar{s}$ the set of all prefixes of s. Finally, we use $\mathbb{N}$ to be the set of natural numbers, $[n]$ and $[n]^+$ to be, respectively, the set of natural numbers and the set of positive natural numbers both bounded by $n \in \mathbb{N}$.

B. Stochastic Discrete Event Systems

We consider a stochastic DES modeled as a Probabilistic Finite-State Automaton (PFA) that is defined similar to a DFA. A PFA is defined by $H := (X_H, \Sigma, Pr_H, x_{0,H})$, where the probabilistic transition function (PTF) $Pr_H : X_H \times \Sigma \times X_H \to [0, 1]$ replaces δ_G. This PTF specifies the probability of moving from state x to state y with event e.

In this work, we limit H further than its general definition. First, we assume that each state in H transitions with probability 1 or deadlocks, i.e., $\sum_{e\in\Sigma}\sum_{y\in X_H} Pr_H(x, e, y) \in \{0, 1\}$ for any x. Nevertheless, our methodology can be easily extended to the general case [42]. Next, we assume that H is deterministic, i.e., $\nexists y, y^* \in X_H,\ y^* \neq y$ such that $Pr_H(x, e, y) > 0$ and $Pr_H(x, e, y^*) > 0$. For convenience, we define the transition function δ_H as $\delta_H(x, e) := y$ if $Pr_H(x, e, y) > 0$. Finally, we define the language generated by H as $\mathcal{L}(H) := \{s \in \Sigma^* \mid \delta_H(x_{0,H}), s)!\}$.

Although the language of PFA H is defined similarly as for DFA G, each string in $\mathcal{L}(H)$ can be quantified by its probability of execution. For this reason, language $\mathcal{L}(H)$ is extended to the notion of probabilistic language (p-language) [43]. Formally, the p-language $L_p(H) : \Sigma^* \to [0, 1]$ is defined recursively for $s \in \Sigma^*$ and $e \in \Sigma$ as: $L_p(H)(\epsilon) := 1$, $L_p(H)(se) := L_p(H)(s)Pr_H(x, e, y)$ if $x = \delta_H(x_{0,H}, s)$, $e \in \Gamma_H(x)$ and $y = \delta_H(x, e)$, and 0 otherwise. Intuitively, $L_p(H)(s)$ represents the probability of executing string s.

In the context of stochastic supervisory control theory, the plant H is controlled by a supervisor as in the previously-described supervisory control framework. Nonetheless, there are different ways of studying its closed-loop behavior [42], [44], [45]. In this paper, we use the results of supervisory control of stochastic DES introduced in [42], where only the plant behaves stochastically. That is, both the specification and the supervisor are deterministic and defined as in the previously-described logical supervisory control framework. However, the supervisor *alters* the probabilistic behavior of the plant via the control actions it takes (disabling events). Recall that in the supervisory control framework, the event set of H is partitioned into controllable, Σ_c, and uncontrollable, Σ_{uc}, events, where the supervisor does not disable uncontrollable events. Conditions for the existence of a supervisor for the above control problem are provided in [42].

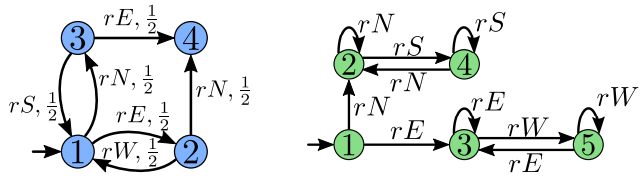
Formalizing the previous discussion, the plant modeled by PFA H is controlled by a supervisor modeled by DFA R. And although R is deterministic, its events disablement proportionally increases the probability of the enabled ones. Given a state $x \in X_H$, a state $y \in X_R$, and an event $e \in \Gamma_H(x) \cap \Gamma_R(y)$, the probability of e being executed is given by the standard normalization:

$$Pr_e^{x,y} = \frac{Pr_H(x, e, \delta_H(x, e))}{\sum_{e'\in\Gamma_H(x)\cap\Gamma_R(y)} Pr_H(x, e', \delta_H(x, e'))} \tag{1}$$

The set $\Gamma_H(x) \cap \Gamma_R(y)$ describes the events that can be executed by H in state x restricted by the events enabled by R in state y. If every event in $\Gamma_H(x)$ is enabled by R in state y, then their probabilities of execution remains unaltered, i.e., the denominator in Eq. (7) is equal to 1. However, if at least one event in $\Gamma_H(x)$ is disabled by R in state y, its probability of execution is proportionally redistributed to the remaining enabled and executable events. Therefore, R/H generates a p-language different, in general, than the p-language of H.

For simplicity and without loss of generality, we assume that the plant H has one deadlock critical state and R ensures that this state is not reachable in R/H. Specifically, we assume that the language $\mathcal{L}(R/H) \subseteq \{s \in \mathcal{L}(H) \mid \delta_H(x_{0,H}, s) \neq x_{crit}\}$ where $x_{crit} \in X_H$ is the critical state and $\mathcal{L}(R/H)$ is controllable, see, e.g., [41], [46]. Normally, one would find a supervisor that generates the supremal controllable sublanguage of $\{s \in \mathcal{L}(H) \mid \delta_H(x_{0,H}, s) \neq x_{crit}\}$, but we do not assume such a supervisor is selected, see Example II.1. For the definition of the supremal controllable sublanguage see, e.g., [41], [46]. Lastly, we define the set of unsafe strings as $U_{uns} = \{s \in \Sigma^* \mid \delta_H(x_{0,H}, s) = x_{crit}\}$ and the set of unsafe state pairs for the controlled system R/H by $X_{uns} := \{x_{crit}\} \times X_R$

Example II.1. Consider a robot that is navigating an area that has been partitioned as a grid with two rows and two columns. The robot is modeled by the PFA H shown in Fig. 1(a). Every event is controllable and the probability transition function is defined as $Pr_H(x, e, y) = \frac{1}{2}$ for the transitions defined by the model in Fig. 1(a). Moreover, we consider that $x_{crit} = 4$, i.e., there is an obstacle in that region and the robot must avoid the obstacle. We show a supervisor in Fig. 1(b) that guarantees the safety of the robot. Note that this supervisor is more restrictive than necessary, i.e., the language $\mathcal{L}(R/G)$ is not the supremal controllable sublanguage. We select this supervisor for illustrative purposes in the following sections. Even though the self-loops in this supervisor do not create new behavior with respect to the plant (enabling infeasible events), they will be used in Example **III.2**.



(a) Robot in a 2 by 2 grid (b) Supervisor R

Fig. 1: Robot example

III. Problem Formulation

In this section, we pose the two problems that are solved in this paper: the probabilistic reachability attack function problem and the multi-objective optimal attack function. We start by formally describing stochastic supervisory control under sensor deception attacks. Following this description, we pose these two problems over the newly described framework. Table I summarizes the notation introduced in this section.

TABLE I: Table of Notation for Systems under Attack

Σ_a	Set of compromised events - $\Sigma_a \subseteq \Sigma$
Σ_i	Set of inserted events - $\{e_{ins} \mid e \in \Sigma_a\}$
Σ_d	Set of deleted events - $\{e_{del} \mid e \in \Sigma_a\}$
Σ_a^e	$\Sigma_i \cup \Sigma_d$
Σ_m	$\Sigma \cup \Sigma_a^e$
$\mathcal{M}(e)$	Removes subscripts ins and del from event $e \in \Sigma_m$
$P^H(e)$	Projects event $e \in \Sigma_m$ to its actual execution in H
$P^S(e)$	Projects event $e \in \Sigma_m$ to its actual observation by R
$A(s)$	Attack function - Def. **III.1**
Π_A	Set of complete and consistent attack functions
$S_A(s)$	Attacked supervisor - $(S \circ P^S \circ A)(s)$
S_A/H	Attacked system
win_A	Winning level of A - Eq. (2)
$cost(s)$	Cumulative cost of executing string s in S_A/H - Eq. (4)
Uns	Set of strings that reach the critical state in S_A/H
Unr	Set of strings that cannot reach the critical state in S_A/H
$E^A[cost(s)]$	Expected cost of generating a string in $Uns \cup Unr$

A. Stochastic supervisory control under sensor deception attacks

Our goal is to investigate the performance of R/H when an attacker undermines the communication channel between the plant and the supervisor, i.e, an attacker hijacks the sensor readings. We define as $\Sigma_a \subseteq \Sigma$ the set of *compromised events*. Any event in Σ_a can be manipulated by the attacker, where "manipulate" means the ability to insert fictitious events in the channel or to erase events from the channel. Under this unreliable communication model, the supervisor R may not guarantee the unreachability of the critical state. This possible unsafe behavior is the focus of our paper.

To identify attack actions from events generated by H, let $\Sigma_i = \{e_{ins} \mid e \in \Sigma_a\}$ and $\Sigma_d = \{e_{del} \mid e \in \Sigma_a\}$ be the set of inserted events and the set of deleted events, respectively. We assume that $\Sigma_a, \Sigma_i, \Sigma_d$ are pairwise disjoint. For convenience, we define $\Sigma_a^e = \Sigma_i \cup \Sigma_d$ and $\Sigma_m = \Sigma \cup \Sigma_a^e$.

Although we can identify the attack actions, the supervisor does not observe these decisions neither does the plant executes them. For example, an event deletion is not seen by the supervisor whereas a compromised event is executed by H before its possible deletion. We define projection operators to identify how these attack actions are portrayed in H and S. For example, event deletion e_{del} is projected to ε by the supervisor projection operator, while it is projected to e by the plant projection operator. Formally, we define three projection operators with Σ_m as domain and Σ as codomain
(1) $\mathcal{M}(e_{ins}) = \mathcal{M}(e_{del}) = \mathcal{M}(e) = e$ for $e \in \Sigma_a$ and $\mathcal{M}(e) = e$ for $e \in \Sigma \setminus \Sigma_a$
(2) $P^H(e) = \mathcal{M}(e)$ for $e \in \Sigma \cup \Sigma_d$ and $P^H(e) = \varepsilon$ for $e \in \Sigma_i$
(3) $P^S(e) = \mathcal{M}(e)$ for $e \in \Sigma \cup \Sigma_i$ and $P^S(e) = \varepsilon$ for $e \in \Sigma_d$.
The mask $\mathcal{M}$ removes subscripts, when present, from events in Σ_m, P^H projects an event in Σ_m to its actual event execution in H, and P^S projects an event in Σ_m to its event observation by R. Finally, we formally define the model of an attacker.

Definition III.1. An attacker that hijacks events in $\Sigma_a \subseteq \Sigma$ in the communication channel between the plant and the supervisor is defined as a partial map $A : \Sigma_m^* \times (\Sigma \cup \{\varepsilon\}) \to \Sigma_m^*$, that satisfies for any $t \in \Sigma_m^*$ and $e \in \Sigma \cup \{\varepsilon\}$:
1) $A(\varepsilon, \varepsilon) \in {\Sigma_i}^*$ and $A(t, \varepsilon) = \varepsilon$ for $t \neq \varepsilon$
2) If $e \in \Sigma_a$, then $A(t, e) \in \{e, e_{del}\}{\Sigma_i}^*$
3) If $e \in \Sigma \backslash \Sigma_a$, then $A(t, e) \in \{e\}{\Sigma_i}^*$

The attack function A defines a deterministic strategy given the last event executed e and the previous attacker modification t, i.e., the string $A(t, e)$ substitutes event e executed by H. For convenience and with an abuse of notation, we extend the function A to recursively concatenate these modifications for any string $s \in \Sigma^*$. Let $s \in \Sigma^*$, then $A(s) := A(s^{|s|-1})A(A(s^{|s|-1}), s[|s|])$ and $A(\varepsilon) := A(\varepsilon, \varepsilon)$.

A new controlled behavior is generated when the attack function A is placed in the communication channel between the plant and the supervisor. Namely, a new supervisor denoted by S_A is defined, where $S_A(s) = (S \circ P^S \circ A)(s)$ is the resulting control action, under attack, after string s has been executed by the system[1]. The language $\mathcal{L}(S_A/H) \subseteq \mathcal{L}(H)$ is defined as usual [41] and S_A/H is denoted as the *attacked system*. This language is defined over Σ, and not Σ_m, due to the projection P^S of the attacker editions. Note that, S_A/H generates a p-language in the same manner as S/H.

***Remark* III.1.** In the definition of the language of S_A/H, the attacker completes its string modification without any interruption of the plant H. In other words, the plant H does not execute any event in the middle of the attacker editions following each event executed by the plant.

Example III.2. Let us provide examples of attacked systems based on Example **II.1** and $\Sigma_a = \{rE\}$. First, we define attack A_1 as follows: $A_1(s) = rE_{ins}rE_{ins}s$ for any $s \in \Sigma^*$. Note that attack A_1 can insert event rE twice since supervisor R has a self-loop with event rE in state 3[2]. The initial control decision is $S_{A_1}(\varepsilon) = S(P^S(A_1(\epsilon))) = \Gamma_R(3) = \{rE, rW\}$, which allows the execution of event rE in state 1 of the plant H with probability 1, i.e., $Pr_{rE}^{1,3} = 1$. It follows that $\mathcal{L}(S_{A_1}/H) = \{\varepsilon, rE, rErW, rErWrE, \dots\}$ since the attacker inserts $rE_{ins}rE_{ins}$ when the system is initialized. This attacked system is depicted in Fig. 4(a), where states are tuples in $X_H \times X_R$. Similarly, we define A_2 as: $A_2(\varepsilon) = \varepsilon$, $A_2(rEs) = rE_{del}s$, and $A_2(rNs) = rNs$ for any $s \in \Sigma^*$. Then, it follows that $\mathcal{L}(S_{A_2}/H) = \{\epsilon, rE, rN, rErN, rNrS, rNrSrN, \dots\}$.

Before concluding this section, we introduce two assumptions on attack functions. First, we assume that the attacker

[1] $\circ$ denotes the function composition operator.
[2] An intrusion detection module would be able to detect attack A_1 since the plant cannot execute $rErE$. For illustrative purposes of our example, we assume that no intrusion detection module is deployed for this system.

always "knows" what to do when the plant executes a new event. We say that the attack function is *complete* when it is defined for every string in the new controlled behavior $\mathcal{L}(S_A/H)$. Moreover, we assume that the attacker does not insert an event that is not allowed by the current supervisor's control decision. An attack function is *consistent* if its insertions are *consistent* with the control decision of the supervisor.

Definition III.2. An attack function A is *complete* w.r.t. H and S if for any s in $\mathcal{L}(S_A/H)$, we have that $A(s)$ is defined. Moreover, A is *consistent* if for any $e \in \Sigma$, $s \in \mathcal{L}(S_A/H)$ such that $se \in \mathcal{L}(S_A/H)$ with $A(s,e) = t$, then $S(P^S(A(s)t^i))!$ and $t[i+1] \in S(P^S(A(s)t^i))$ for all $i \in [|t|-1]$. We define Π_A as the set of all complete and consistent attack functions w.r.t. H and S.

***Remark* III.2.** Although we did not consider a detection module in this framework, one can be incorporated into the supervisor if we slightly modify supervisor R. Namely, we can introduce a supervisor that embeds an intrusion detection module. Intuitively, $\tilde{R} := (X_{\tilde{R}} := X_R \cup \{dead\}, \Sigma, \delta_{\tilde{R}}, x_{0,\tilde{R}})$ is a copy of R augmented with a deadlock state called $dead$ that is only reached via strings in $\Sigma^* \setminus \mathcal{L}(R/H)$, see [6] for more details. The state $dead$ is used to capture an intrusion detection mechanism, where an attacker is detected when the supervisor reaches this state.

B. The maximal reachability problem

Since the controlled behavior under the influence of attack function A is well defined by $\mathcal{L}(S_A/H)$, we can define the objective of the attacker based on this language. The attack function is successful if the attacked system generates an unsafe string, i.e., $U_{uns} \cap \mathcal{L}(S_A/H) \neq \emptyset$. Because each unsafe string in the attacked system is quantified by $L_p(S_A/H)$, we quantify each attack function A by the total probability of generating these unsafe strings. The *winning level of* A is defined as the probability that S_A/H generates unsafe strings. Formally,

$$win_A = \sum_{s \in U_{uns}} L_p(S_A/H)(s) \tag{2}$$

It follows from U_{uns} that $win_A \leq 1$ for any $A \in \Pi_A$, i.e., win_A captures the *probability* that S_A/H generates unsafe strings.

The definition of the value win_A makes it possible to compare attack functions. For example, attack A_1 in Example **III.2** has a winning level of 0, $win_{A_1} = 0$, since $U_{uns} \cap \mathcal{L}(S_A/H) = \emptyset$. Whereas, attack A_2 in Example **III.2** has a winning level of 0.5 since $U_{uns} \cap \mathcal{L}(S_A/H) = \{rErN\}$ and $L_p(S_{A_2}/H)(rErN) = 0.5$. Given two attack functions A and A', if $win_A \geq win_{A'}$, then it means that strategy A is equally or more likely to reach the critical state of H than A'. A natural question to ask is if there exists an attack function that is more likely to reach the critical state than any other attack function. Formally, the problem is posed as follows.

Problem III.1. [The probabilistic reachability attack function problem] Given a plant modeled as PFA H, a supervisor modeled as DFA R, and the set of compromised events $\Sigma_a \subseteq \Sigma$, synthesize $A^{max} \in \Pi_A$, if it exists, such that:

$$win_{A^{max}} = \sup_{A \in \Pi_A} win_A \tag{3}$$

***Remark* III.3.** As was mentioned in the introduction, the definition of attack functions resembles the definition of edit functions in [37], [47]. Because of this similarity, the problem statements of synthesizing edit functions and of synthesizing attack functions appear similar from a high-level perspective. However, these problems differ in an important way. The attack function of this paper affects the *behavior executed* by the plant and the *behavior observed* by the supervisor. In the case of opacity, although the edit function affects the *observed behavior* of the considered model, it *does not* affect the *behavior executed* by this model.

C. The multi-objective problem

In Problem **III.1**, the attacker is "eager" to reach the critical state and it is not constrained, except by Σ_a, in how to do so. Finding a cost-optimal attack function would be a natural extension for the investigated problem [48], [49], when insertions and deletions are costly actions by the attacker. Nonetheless, the reachable-optimal and the cost-optimal problems are conflicting. In this manner, we investigate a multiple-objective problem, i.e., maximal reachability and minimal expected cost.

In this paper, we follow a similar approach as in [50], [51] where objectives have priorities. Namely, the attacker has multiple objectives to satisfy: (i) maximize the probability of reaching the critical state; (ii) minimize the expected cost while performing (i). First, the attacker prioritizes the probability of reaching the critical state and finds all attack functions that maximize this probability. Second, within these maximal attack functions, it searches for an attack function with the minimum expected cost.

We start by assuming that the cost of a string modified by the attacker is given by summing the cost of each insertion and deletion in this string. We assume that the attacker terminates its attack in two cases: (1) when it reaches the critical state; (2) when it *cannot* reach the critical state. Therefore, we define the expected cost on the modification of strings that meet one of two termination criteria.

To depict this idea, we again use Example **III.2** with attacker A_2. In S_{A_2}/H, we know that string $rErN$ reaches the critical state, whereas the execution of string rN removes any chance of the attacker to reach the critical state. In this example, the expected cost is defined based on executing these two strings and their modifications, i.e., $A_2(rErN)$ and $A_2(rN)$. Namely, the probability of executing $rErN$ is given by $L_p(S_{A_2}/H)(rErN) = 0.5$ and let the cost of $A_2(rErN)$ be 2. On the other hand, the probability of executing rN is given by $L_p(S_{A_2}/H)(rN) = 0.5$ and $A_2(rN)$ has zero cost for the attacker since $A_2(rN) = rN$. In this case, the expected cost of A_2 is defined as $2 \times 0.5 + 0 \times 0.5 = 1$.

Formalizing the above discussion, let $w : \Sigma_m \to [0, \infty)$ define the cost of each event, i.e., $w(e) \geq 0$ if e is an insertion or deletion event, otherwise $w(e) := 0$. Next given a string $t \in \Sigma_m^*$, the cost of t is $\sum_{i=1}^{|t|} w(t[i])$. Following this, we

define the terminating language based on the two described termination criteria. The first criterion is met by strings in $Uns := \mathcal{L}(S_A/H) \cap U_{uns}$. The second criterion is met by strings that cannot reach the critical state. These strings are defined by $Unr := \{s \in \mathcal{L}(S_A/H) \setminus \overline{Uns} \mid s^{|s|-1} \in \overline{Uns}\}$. Since Unr and Uns are defined over Σ^*, we must define a cost function over Σ^* and not Σ_m^*. Formally, the cost function $cost : \Sigma^* \to [0, \infty)$ is defined for $s \in \Sigma^*$ as follows:

$$cost(s) = \begin{cases} \sum_{i=1}^{|A(s)|} w\big(A(s)[i]\big) & \text{if } A(s)! \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Since we are interested in the expected cost value of strings in $Uns \cup Unr$, we define the expected cost value, as in [43], based on this set of strings and $L_p(S_A/H)$. It follows that any string in $Uns \cup Unr$ is not a prefix of any other string in this language and $\sum_{s \in Uns \cup Unr} L_p(S_A/H)(s) = 1$ for any $A \in \Pi_A$. In this manner, we can define a probability mass function on $Uns \cup Unr$ based on $L_p(S_A/H)$ [43], [52]. Intuitively, we "force" the termination of $\mathcal{L}(S_A/H)$ since only strings in $Uns \cup Unr$ are of interest. The expected value is defined as follow:

$$\mathbb{E}^{Uns \cup Unr, S_A/H}[cost(s)] = \sum_{s \in Uns \cup Unr} cost(s) L_p(S_A/H)(s) \tag{5}$$

where string $s \in \mathcal{L}(S_A/H)$ is a random variable.

Posing a minimization problem based on this cost function is an ill-posed problem since an attacker that does not attack is a solution. Thus, this cost function only works, because the minimization problem is preceded by a maximization problem.

Problem III.2. [Multi-Objective Optimal Attack Function Synthesis Problem] Given a plant H, a supervisor R, the set of compromised events $\Sigma_a \subseteq \Sigma$, a solution of Problem **III.1** A^{max}, and the cumulative cost function $cost$, synthesize A^{multi}, if it exists, such that $win_{A^{multi}} = win_{A^{\max}}$ and

$$\mathbb{E}^{A^{multi}}[cost(s)] = \inf_{\substack{A \in \Pi_A \text{ s.t.} \\ win_A = win_{A^{max}}}} \mathbb{E}^{A}[cost(s)] \tag{6}$$

where $\mathbb{E}^A$ is an abbreviation for $\mathbb{E}^{Uns \cup Unr, S_A/H}$ as in Eq. (5).

We search for an attack function in Π_A that satisfies Problem **III.1** and produces the minimal expected cumulative cost. This problem is well-posed since we search for an attack function that generates the minimum expected cost within the policies that reach the critical state with the maximum winning level. Therefore, the two problems we pose (Problems **III.1** and **III.2**) can be solved in sequence.

IV. Markov Decision Processes

We briefly review concepts and notations of Markov decision processes (MDPs). An MDP is a tuple $M := (Q, Act, \delta_M, q_{0,M})$, where Q is a finite set of states, $q_{0,M} \in Q$ is an initial state, Act is a finite set of actions, and $\delta_M : Q \times Act \times Q \to [0, 1]$ is a PTF, where $\sum_{q' \in Q} \delta_M(q, a, q') \in \{0, 1\}$ for all $q \in Q$ and $a \in Act$.

An infinite run in M is an infinite sequence of pairs (state, action) that satisfies the PTF, i.e., $\rho := q^0 a^0 q^1 a^1 \ldots$ such that $\delta_M(q^j, a^j, q^{j+1}) > 0$ for all $j \in \mathbb{N}$. The set of all infinite runs in M initialized at state q is defined as $Runs_q(M)$, and $Runs(M) := Runs_{q_{0,M}}(M)$. Similarly, a finite run ρ in M is a finite prefix of an infinite run ending in a state. The length of a finite run ρ, denoted by $|\rho|$, is defined by the number of actions in this run, e.g., $|\rho| = 1$ for $\rho = q^0 a^0 q^1$. The set of all finite runs in M initialized at state q is defined as $Pref_q(M)$, and $Pref(M) := Pref_{q_{0,M}}(M)$.

Connecting the worlds of PFA and MDPs, a PFA can be seen as an MDP with only one action and labeled transitions, i.e., a Markov chain with labeled transitions [53]. Since there is a single action, only the environment stochastically decides a labeled transition to execute. However, MDPs possibly have more than one action to be selected by an "agent". Only after the "agent" selects an action will the environment stochastically decide a transition to execute. Therefore, the environment's stochastic decision is coupled to the decision made by the "agent".

This "agent" is defined by a deterministic strategy $\pi : \cup_{q \in Q} Pref_q(M) \to Act$ that maps finite runs into actions. The set of all (deterministic) strategies of M is defined as Π_M. Under a fixed strategy $\pi \in \Pi_M$, the behavior of M is fully probabilistic and it can be represented by an induced discrete-time Markov chain [49], [53]. This fixed strategy leads us to the standard definition of a probability measure $Pr^{\pi}_{M,q}$ (Pr^{π}_{M}) over $Runs_q(M)$ ($Runs(M)$) [49]. We use the notation $Runs^{\pi}_q(M), Runs^{\pi}(M), Pref^{\pi}_q(M)$, and $Pref^{\pi}(M)$ to denote the set of runs generated by the MDP with $\pi \in \Pi_M$.

Again connecting PFA and MDPs, the definitions of the probability measures $L_p(H)$ and Pr^{π}_M are *almost* identical. The measure Pr^{π}_M is only defined for infinite runs [49], whereas $L_p(H)$ is defined for finite and infinite strings [43].

In this paper, we are interested in two MDP problems, the *probabilistic reachability problem* [2], [48] and the *stochastic shortest path problem* [1], [2]. First, we define the probability of reaching an absorbing set $Q_{abs} \subseteq Q$ from state $q \in Q$ as:

$$p^{\pi}_{M,q}(Q_{abs}) := Pr^{\pi}_{M,q}(\{\rho \in Runs^{\pi}_q(M) | \exists j \in \mathbb{N},\ q^j \in Q_{abs}\}) \tag{7}$$

Similar to the definition of runs, we define $p^{\pi}_M(Q_{abs}) = p^{\pi}_{M,q_{0,M}}(Q_{abs})$. Policies $\pi \in \Pi_M$ that reach Q_{abs} with probability 1 are denoted as proper policies [1]. Intuitively, the *probabilistic reachability problem* searches for the maximal probability of reaching a set of absorbing states.

Problem IV.3. Given an MDP M and a set of absorbing states $Q_{abs} \subseteq Q$, find

$$p^*_M(Q_{abs}) := \sup_{\pi \in \Pi_M} p^{\pi}_M(Q_{abs}) \tag{8}$$

Let $\mathbb{E}^{\pi}_{M,q}(f)$ denote the expected value of a measurable function $f : Runs_q(M) \to [0, \infty]$ with respect to $Pr^{\pi}_{M,q}$, see, e.g., [48], [49] for details. Intuitively, the *stochastic shortest path problem* searches for the minimal expected cost to reach a set of absorbing states.

Problem IV.4. [1] Given an MDP M, a set of absorbing states $Q_{abs} \subseteq Q$, and a cost function $c : Q \times Act \to [0, \infty]$, find

$$c^*_M(Q_{abs}) := \inf_{\pi \in \Pi_M} \mathbb{E}^{\pi}_M[cumul(Q_{abs}, \rho)] \tag{9}$$

where $\rho \in Runs(M)$ and

$$cumul(Q_{abs}, \rho) = \begin{cases} \sum_{k=0}^{j_\rho} c(q^k, a^k) & \text{if } \exists j \in \mathbb{N} \text{ s.t. } q^j \in Q_{abs} \\ \infty & \text{otherwise} \end{cases}$$

and $j_\rho = \min\{j \in \mathbb{N} \mid q^j \in Q_{abs}\}$.

Problem **IV.4** has a finite solution if and only if there exists at least one strategy that reaches the absorbing set with probability 1, i.e., $c^*_M < \infty$ if and only if $\sup_{\pi \in \Pi_M} p^\pi_M(Q_{abs}) = 1$.

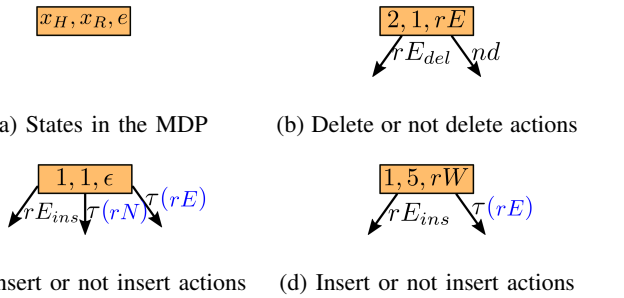
Both problems can be solved using standard MDP algorithms, e.g., value iteration, policy iteration, or linear programming, see [1], [2], [49]. The computational complexity of solving these problems is pseudo-polynomial via the linear programming method [48] Moreover, both problems accept memoryless and deterministic strategies as solutions [1], [49].

V. Solution of the Probabilistic Reachability Attack Function Problem

The syntax of Problems **III.1** and **IV.3** is identical when we compare Eqs. (3) and (8). However, these problems are semantically different since they are defined in two different, though similar, frameworks (PFA vs. MDP). In this section, we reduce Problem **III.1** to Problem **IV.3** showing that they are semantically equivalent with the right MDP construction.

A. Construction of the MDP

As was mentioned, a PFA can be seen as an MDP with only one action and labeled transitions. Similarly, a PFA can be seen as an MDP with a given strategy and labeled transitions. Therefore, a given attacked controlled system S_A/H can be seen as an MDP under a fixed strategy defined based on an attack function. Since our goal is to synthesize attack functions, we construct an MDP where actions represent attacker actions in some attacked controlled system.



(a) States in the MDP (b) Delete or not delete actions

(c) Insert or not insert actions (d) Insert or not insert actions

Fig. 2: States and Actions in the MDP

Let us intuitively describe the construction of this MDP that simulates the behavior of attacked systems. Figure 2(a) depicts the information in each MDP state. The variables x_H and x_R represent states of the plant H and the supervisor R, which are necessary since the evolution of the plant depends on them, see Eq. (1). The variable e represents the last event executed by the plant since the attacker needs to identify them, see Def. **III.1**.

Moving on, the actions in this MDP represent the attacker actions in an attacked controlled system. The attacker can insert events (Σ_i), delete events (Σ_d), not delete events (nd), or allow the system to execute events (τ). These actions are the only possible actions in this MDP.

Next, using Example **II.1** and $\Sigma_a = \{rE\}$, we describe the possible transitions in this MDP. In Fig. 2(b), the last event executed is $rE \in \Sigma_a$, for which the attacker can decide to delete (rE_{del}) or to not delete (nd) this event. These two actions are only possible after the execution of a compromised event. In Fig. 2(c), the plant did not execute an event ($e = \varepsilon$) in the last transition. Therefore, the attacker can insert rE_{ins} since $rE \in \Gamma_R(1)$ or allow the plant to execute an event with decision τ. If the attacker selects τ, the plant can execute either rN or rE, in blue[3], in this current state. Similarly in Fig. 2(d), the attacker can insert rE_{ins} or let the plant proceed with action τ. Note that only action τ introduces randomness since the plant H randomly execute events (nondeterminism in Fig. 2(c)).

Finally, we can formally introduce the MDP as follows.

Definition V.3. Given plant H, supervisor R, and compromised event set Σ_a, the MDP $M = (Q, Act, \delta_M, q_{0,M})$ is constructed in the following manner.

- $Q \subseteq X_H \times X_R \times (\Sigma \cup \{\varepsilon\})$. For a state q, we denote by q_H, q_R and q_e the plant state, the supervisor state, and the last executed event, i.e., $q = (q_H, q_R, q_e)$.
- The set $Act := \{\tau, nd\} \cup \Sigma_i \cup \Sigma_d$.
- The initial state $q_{0,M} := (x_{H,0}, x_{R,0}, \varepsilon)$.
- The PTF δ_M is defined as follows:

For action τ and $x, y \in Q$:

$$\delta_M(x, \tau, y) := \begin{cases} Pr^{x_H, x_R}_{y_e} & \text{if } y_e \in \Sigma \setminus \Sigma_a, y_H = \delta_H(x_H, y_e) \\ & y_R = \delta_R(x_R, y_e) \\ Pr^{x_H, x_R}_{y_e} & \text{if } y_e \in \Sigma_a, y_H = \delta_H(x_H, y_e) \\ & y_e \in \Gamma_R(x_R), y_R = x_R \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

For action nd and $x, y \in Q$:

$$\delta_M(x, nd, y) := \begin{cases} 1 & \text{if } x_e \in \Sigma_a,\ y_H = x_H, \\ & y_R = \delta_R(x_R, x_e),\ y_e = \varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Lastly, for action $a \in \Sigma^e_a$ and $x, y \in Q$:

$$\delta_M(x, a, y) := \begin{cases} 1 & \text{if } a \in \Sigma_i,\ (x_H, x_R) \notin X_{uns}, \\ & y = (x_H, \delta_R(x_R, P^S(a)), \varepsilon) \\ 1 & \text{if } a \in \Sigma_d,\ (x_H, x_R) \notin X_{uns}, \\ & x_e = \mathcal{M}(a),\ y = (x_H, x_R, \varepsilon) \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

- Post-Processing: for any $x \in Q$ such that $\sum_{y \in Q} \delta_M(x, a, y) = 0$ for all $a \in Act \setminus \Sigma_i$, δ_M is augmented for x by defining: $\delta_M(x, \tau, (x_H, x_R, \varepsilon)) := 1$ and $\delta_M((x_H, x_R, \varepsilon), \tau, (x_H, x_R, \varepsilon)) := 1$.

Equation (10) provides conditions for the τ action, which are divided into two cases based on compromised or not compromised events. If the event executed is not compromised,

[3]The information in blue is not part of the τ action. We show it here to illustrate which events can be executed.

then the states of H and R are updated. When the event executed is compromised, then only the state of H is updated; the attacker must decide next to delete or not delete this event. Equation (11) represents the attacker's choice of not deleting the just executed compromised event; and the state of R gets updated. The delete action is specified in the second case of Eq. (12), where the states of H and R are not updated. Lastly, the first case of Eq. (12) represents an insertion of a fictitious event; only the state of R transitions.

Using our running example, we explain Def. **V.3** in detail, where we interpret Eqns. (10-12) and connect them with our previous discussion on the construction of M.

Example V.3. In order to illustrate the construction of M, we use Example **II.1** with compromised event set $\Sigma_a = \{rE\}$. The MDP M for this example is depicted in Fig. 3. In Fig. 3, the transition function is represented by the directed edges with a respective action and probability value. We can fully see the attacker actions and their results compared to our initial discussion with Fig. 2. From state $(2,1,rE)$, the attacker can either delete or not delete the last executed event rE. Delete implies that the supervisor remains in the same state, Eq. (12), as seen in the finite run $(1,1,\varepsilon)\tau(2,1,rE)rE_{del}(2,1,\varepsilon)$. On the other hand, not delete implies that the supervisor receives event rE and moves to state 3, Eq. (11), as seen in the finite run $(1,1,\varepsilon)\tau(2,1,rE)nd(2,3,\varepsilon)$. We also analyze the action τ in state $(1,1,\varepsilon)$, where the plant can either execute event rN with probability $Pr^{1,1}_{rN} = 1/2$ (first case in Eq. 10) or rE with probability $Pr^{1,1}_{rE} = 1/2$ (second case in Eq. 10). The insertion rE_{ins} in state $(1,1,\varepsilon)$ only alters the supervisor state, Eq. (12), as seen in the run $(1,1,\varepsilon)rE_{ins}(1,3,\varepsilon)$. Lastly, state $(4,2,\varepsilon)$ is added in the post-processing phase since state $(4,2,rN)$ was a deadlock state. The post-processing can also provide an action to break possible infinite insertion cycles in M. For example, if action τ was not defined in state $(1,3,\varepsilon)$ by Eq. (10), then the post-processing would add a self-loop with action τ in this state.

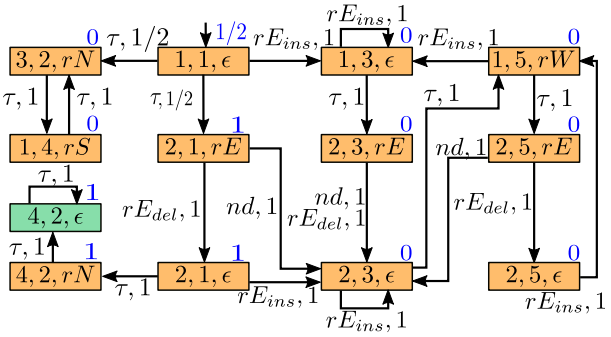


Fig. 3: MDP for controlled system R/H and $\Sigma_a = \{rE\}$

To connect the MDP M to attacked system S_A/H, we extract two strings from a given finite run in M. In Example **V.3**, let $\rho = (1,1,\varepsilon)\tau(2,1,rE)rE_{del}(2,1,\varepsilon)rE_{ins}(2,3,\varepsilon)$, then $s_\rho = \varepsilon\ rE\ \varepsilon\ \varepsilon = rE$ is the string executed by H while $t_\rho = \varepsilon\ rE_{del}\ rE_{ins} = rE_{del}rE_{ins}$ is the string modified by the attacker. Namely, string s_ρ concatenates the events in each state of run ρ, i.e., $s_\rho := q_e^0 q_e^1 \dots q_e^{|\rho|}$ for $\rho \in Pref(M)$. On the other hand, string t_ρ contains the modifications made by an attacker, i.e., it defines the modification on s_ρ for $\rho \in Pref(M)$. For $\rho \in Pref(M)$ such that $q_e^{|\rho|} \notin \Sigma_a$, we define $t_\rho = e_0 e_1 \dots e_{|\rho|-1}$ where

$$e_i = \begin{cases} a^i & \text{if } a^i \in \Sigma_a^e \\ q_e^{i+1} & \text{if } q_e^{i+1} \in \Sigma \setminus \Sigma_a \text{ or } q_e^{i+1} \in \Sigma_a \text{ and } a_\rho^{i+1} = nd \\ \varepsilon & \text{otherwise} \end{cases} \tag{13}$$

Note that in the construction of t_ρ, to confirm that an event in H is executed between states i and $i+1$ we look at q_e^{i+1}. Lastly, we define the objective set Obj as the set of absorbing states where the attacker has reached its goal.

$$Obj = \{(q_H, q_R, \varepsilon) \in Q \mid (q_H, q_R) \in X_{uns}\} \tag{14}$$

B. Maximal reachability attack function

The definition of winning level is directly related to an attack function A. In other words, once an attack function A is fixed, we obtain the language $\mathcal{L}(S_A/H)$ and consequently the value of win_A. The same idea applies to the MDP M; once a strategy π is fixed, we can calculate the probability of reaching the states in Obj. The construction of MDP M and the set Obj ties these two problems together. The following theorem connects the solution of Problem **IV.3** for MDP M and set Obj with the solution of Problem **III.1**.

Theorem V.1. Consider the MDP M and the set Obj, then

$$win_{A^{max}} = p^*_M(Obj) \tag{15}$$

Although Problem **IV.3** and Problem **III.1** have similarities, to formally prove Theorem **V.1** a sequence of intermediate results is needed. These intermediate results are intuitively stated in the following subsection and their formal proofs are attached in Appendix I-A. Recall that the solution procedure of Problem **IV.3** not only outputs the maximum probability $p^*_M(Q_{abs})$ but also outputs $p^*_{M,q}(Q_{abs})$ for all $q \in Q$. Since in Theorem **V.1** $Q_{abs} = Obj$, one can use standard methods to extract an optimal strategy π^* that achieves $p^{\pi^*}_M(Obj) = p^*_M(Obj)$. In Section V-C, we show how to construct an attack function A^{max} based on π^* such that A^{max} is a solution of Problem **III.1**.

Example V.4. We solve Problem **IV.3** for the MDP depicted in Fig. 3. The solution procedure of Problem **IV.3** outputs $p^*_q(Obj)$ for each $q \in Q$; these values are shown in blue in Fig. 3. The probability of reaching the set $Obj = \{4,2,\varepsilon\}$ (in green) is $p^*_M(Obj) = 0.5$, which implies that there exists $A^{\max}$ such that $win_{A^{\max}} = 0.5$. In fact, attacker A_2 defined in Example **III.2** is this attacker since $win_{A_2} = 0.5$ This is of course the expected solution for Problem **III.1** since deleting event rE after its execution leads the supervisor to allow a move to the critical state 4.

***Remark* V.4.** The construction of the MDP is defined in a similar manner as the construction of graph-games in [6], [29] when we ignore the probabilistic part of the MDP. An MDP is a special case of a graph-game where only one player is present [8], [54]. Therefore, our results related to the logical part (without probabilities) of the MDP are not surprising

since they are similar to results in [6], [29]. Nonetheless, the similarities stop at the logical part. Thus, one of the key contributions of our paper is to bridge the gap in the probabilistic part of the MDP by showing that we can fully use our MDP construction since we connect both the logical and probabilistic parts of it to the investigated problem.

C. Discussion on Theorem V.1

In order to prove Theorem **V.1**, we present a sequence of equivalences between an attacked system and the MDP M instantiated by a strategy $\pi \in \Pi_M$.

Although attack functions can only generate finite insertions (arbitrarily long), see Def. **III.1**, the construction of M allows strategies that represent infinite sequences of insertions. For example, a strategy that selects action rE_{ins} in state $(1,3,\varepsilon)$ in M of Fig. 3 would simulate the attacker inserting infinitely many events. Fortunately, there exist optimal strategies that do not generate these infinite insertions sequences.

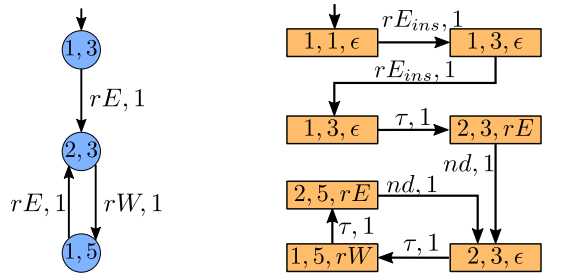
First, we show in Lemma **V.1** that states caught in an infinite cycle of insertion actions cannot reach the objective set Obj, i.e., they have probability zero of reaching Obj. Intuitively, the infinite cycle of insertion actions prevents runs from leaving this cycle once in it. Since none of the states in this cycle is in the objective set, the runs cannot reach this objective set. In Markov chain notation, these states form a closed class in the Markov chain generated by this strategy [53].

Lemma V.1. Let $\pi \in \Pi_M$. For all $\rho \in Runs^{\pi}(M)$ such that $(\exists j \in \mathbb{N})(\forall k \geq j)[\pi(q^0 \dots q^k) \in \Sigma_i]$, then $p^{\pi}_{M,q^j}(Obj) = 0$.

Let Π^f_M be the set of all strategies of M that *do not* generate infinite insertions, i.e., $\Pi^f_M = \{\pi \in \Pi_M \mid \rho \in Runs^{\pi}(M) \Rightarrow (\forall j \in \mathbb{N})(\exists k \geq j)[\pi(q^0 \dots q^k) \notin \Sigma_i]\}$. Based on Lemma **V.1**, Prop. **V.1** states that there exists an optimal strategy in Π^f_M.

Proposition V.1. $p^*_M(Obj) = \max_{\pi \in \Pi^f_M} p^{\pi}_M(Obj)$.

The proof of Proposition **V.1** also provides that a memoryless strategy in Π^f_M achieves $p^*_M(Obj)$ since a memoryless strategy in Π_M achieves $p^*_M(Obj)$ [1], [49]. Consequently, an optimal strategy can be found in the space of memoryless strategies in Π^f_M.



(a) Attacked system S_{A_1}/H (b) MDP M under strategy π_{A_1}

Fig. 4: Connection attacked system S_{A_1}/H with M

Our next step will connect the set of strategies in Π^f_M to the set of attack functions in Π_A. Recall the definition of A_1 in Example **III.2**, where S_{A_1}/H is shown in Fig. 4(a). Taking string ε with its modification $A_1(\varepsilon) = rE_{ins}rE_{ins}$, we can map this pair of strings to the finite run $(1,1,\varepsilon)rE_{ins}(1,3,\varepsilon)rE_{ins}(1,3,\varepsilon)$ in M, Fig. 3. Actually, we define a strategy π_{A_1} that generates this run: $\pi_{A_1}((1,1,\varepsilon)) = rE_{ins}$ and $\pi_{A_1}((1,1,\varepsilon)rE_{ins}(1,3,\varepsilon)) = rE_{ins}$. Strategy π_{A_1} is completely defined by selecting a string s in $\mathcal{L}(S_{A_1}/H)$ and its modification $A_1(s)$, mapping this pair $(s, A_1(s))$ to a finite run in M, and defining strategy π_{A_1} for this run. Figure 4(b) depicts the MDP M under strategy π_{A_1}. In fact, the reverse direction is also feasible, in which a strategy in M defines an attack function, i.e., strategy π_{A_1} defines attack function A_1. Our next steps shows that this discussion is indeed possible.

First, we show that a string s in $\mathcal{L}(S_A/H)$ and its modification $A(s)$ are mapped to a finite run in M. In this manner, we can always find a finite run for a given pair $(s, A(s))$ for $s \in \mathcal{L}(S_A/H)$. The converse is also true, where a given finite run generates a pair $(s, A(s))$ for some attacked system S_A/H.

Proposition V.2. Given $A \in \Pi_A$, for any $s \in \mathcal{L}(S_A/H)$ and $t = A(s)$, there exists a unique $\rho \in Pref(M)$ such that $s = s_\rho$, $t = t_\rho$, and $(a^i = \tau \Rightarrow q_e^{i+1} \neq \varepsilon)$ for any $i \in [|\rho| - 1]$.

The condition $(a^i = \tau \Rightarrow q_e^{i+1} \neq \varepsilon)$ eliminates the "superfluous" behavior introduced by the post-processing. For example, using attacker A_2 of Example **III.2**, string $rErN \in \mathcal{L}(S_{A_2}/H)$ and $A_2(rErN) = rE_{del}rN$ can be mapped to more than one finite run in M, e.g., $\rho = (1,1,\varepsilon)\tau(2,1,rE)rE_{del}(2,1,\varepsilon)\tau(4,2,rN)$ and $\rho\tau(4,2,\varepsilon)$. However, transition $(4,2,rN)\tau(4,2,\varepsilon)$ was introduced in the post-processing to make M live. Thus, only the run $(1,1,\varepsilon)\tau(2,1,rE)rE_{del}(2,1,\varepsilon)\tau(4,2,rN)$ satisfies the condition on Prop. **V.2**.

Using this map between $(s, A(s))$ and finite runs in M, we construct a strategy for M that simulates the attacked system S_A/H. For example, the pair $(rE, rE_{ins}rE_{ins}rE)$ for S_{A_1}/H is mapped to run $(1,1,\varepsilon)rE_{ins}\dots\tau(2,3,rE)$, for which we define $\pi_{A_1}((1,1,\varepsilon)) = rE_{ins}$, $\pi_{A_1}((1,1,\varepsilon)rE_{ins}(1,3,\varepsilon)) = rE_{ins}$, and so on. For completeness, we complete the definition of this strategy for unmapped finite runs, which include the "superfluous" finite runs and finite runs that *are not* generated with this strategy.

Definition V.4. Given $A \in \Pi_A$, we define $\pi_A \in \Pi^f_M$ using the result of Prop. **V.2**. Let $s \in \mathcal{L}(S_A/H)$, π_A is defined along the unique $\rho \in Pref(M)$ mapped to the pair $(s, A(s))$ as in Prop. **V.2**:

$$\pi_A(q^0 a^0 q^1 \dots q^j) = a^j, \ j \in [|\rho| - 1] \tag{16}$$

For any other $\rho \in Pref(M)$, then $\pi_A(\rho) = \tau$ if $q_e^{|\rho|} \notin \Sigma_a$ and $\pi_A(\rho) = nd$ if $q_e^{|\rho|} \in \Sigma_a$.

Conversely, we construct an attack function based on a strategy in Π^f_M. Recall that action τ in M represents the attacker allowing the system to execute an event. Therefore, action τ delineates the end of the attacker's modification for the last executed event. For example, let $\pi \in \Pi^f_M$ be the strategy depicted in Fig. 4(b), for which $\pi((1,1,\varepsilon)rE_{ins}(1,3,\varepsilon)rE_{ins}(1,3,\varepsilon)) = \tau$. Since the string ε is executed by this run and $rE_{ins}rE_{ins}$ is the string modification, we can define $A_\pi(\varepsilon) = rE_{ins}rE_{ins}$. Hence,

given a strategy $\pi \in \Pi_M^f$, we define an attack function based on runs in $\rho \in Pref^{\pi}(M)$ where $\pi(\rho) = \tau$, i.e., the next action after ρ is τ. Note that these runs always exist since we select π from Π_M^f, i.e., no infinite insertions.

Definition V.5. Let $\pi \in \Pi_M^f$ and $\rho \in Pref^{\pi}(M)$ such that $\pi(\rho) = \tau$, then we define $A_\pi \in \Pi_A$ as:

$$A_\pi(s_\rho) = t_\rho \tag{17}$$

Until this point, we have connected finite runs in M to strings in attacked systems and vice versa, i.e., connecting the logical part between M and attacked systems S_A/H. The next step is to show that the probabilistic part of these two frameworks also coincides. For example, given the attack function A_1 of Example **III.2**, we know that $L_p(S_{A_1}/H)(rE) = Pr_{rE}^{1,3} = 1$. Based on A_1 and Def. **V.4**, we construct strategy π_{A_1}, which generates the finite run $\rho = (1,1,\varepsilon)rE_{ins}(1,3,\varepsilon)rE_{ins}(1,3,\varepsilon)\tau(2,3,rE)$. The probability of generating ρ is defined as:

$$\begin{aligned} Pr_M^{\pi_{A_1}}(\langle\rho\rangle) &= \delta_M(q^0,a^0,q^1)\delta_M(q^1,a^1,q^2)\delta_M(q^2,a^2,q^3) \\ &= 1 \times 1 \times Pr_{rE}^{1,3} = 1 \end{aligned}$$

where $\langle\rho\rangle$ is the set of infinite runs extended from ρ since the probability measure $Pr_M^{\pi_{A_1}}$ computes measures of sets with infinite runs. Although we have only shown that $L_p(S_{A_1}/H)(rE) = Pr_M^{\pi_{A_1}}(\langle\rho\rangle)$, these steps are extended to any $s \in \mathcal{L}(S_{A_1}/H)$, which shows the equivalence of $L_p(S_{A_1}/H)$ and $Pr_M^{\pi_{A_1}}$.

Therefore, we show that the measure $L_p(S_A/H)$ coincides with $Pr_M^{\pi_A}$ for a given attack function A. Conversely, we also show that the probability measure Pr_M^{π} is equivalent to $L_p(S_{A_\pi}/H)$ for a given strategy $\pi \in \Pi_M^f$. This result is shown in Prop. **V.3**. But first, let us formalize, as in [49], the probability of generating a finite run $\rho \in Pref^{\pi}(M)$ for a given strategy $\pi \in \Pi_M^f$:

$$Pr_M^{\pi}(\langle\rho\rangle) = \prod_{j\in[|\rho|-1]} \delta_M(q^j,a^j,q^{j+1}) \tag{18}$$

where $\langle\rho\rangle = \{q^{*0}a^{*0}\cdots \in Runs^{\pi}(M) \mid q^{*j} = q^j, \forall j \in [|\rho|] \wedge a^{*k} = a^k, \forall k \in [|\rho|-1]\}$.

Proposition V.3. Given $A \in \Pi_A$, then for any $s \in \mathcal{L}(S_A/H)$:

$$L_p(S_A/H)(s) = Pr_M^{\pi_A}(\langle\rho\rangle) \tag{19}$$

where $\rho \in Pref^{\pi_A}(M)$, $s = s_\rho$, and $A(s) = t_\rho$. Conversely, given $\pi \in \Pi_M^f$, then for any $\rho \in Pref^{\pi}(M)$:

$$Pr_M^{\pi}(\langle\rho\rangle) = L_p(S_{A_\pi}/H)(s_\rho) \tag{20}$$

Finally, we state a proposition which is a direct consequence of Prop. **V.3**. This proposition states that the winning level of an attack function is equal to the probabilitity of reaching the objective set in M using the strategy generated by this attack function. Similarly, the probability of reaching the objective set in M via a given strategy is equal to the winning level of the attack function generated by this strategy.

Proposition V.4. Given a strategy $\pi \in \Pi_M^f$, then $p_M^{\pi}(Obj) = win_{A_\pi}$. Conversely, let $A \in \Pi_A$, then $win_A = p_M^{\pi_A}(Obj)$.

Finally, the proof of Theorem **V.1** follows directly from Prop. **V.4**. Let $\pi^* \in \Pi_M^f$ be a strategy such that $p_M^*(Obj) = p_M^{\pi^*}(Obj)$, then $win_{A_{\pi^*}} = p_M^{\pi^*}(Obj)$. For any $A \in \Pi_A$, we have that $win_A = p_M^{\pi_A}(Obj) \leq p_M^*(Obj) = p_M^{\pi^*}(Obj) = win_{A_{\pi^*}}$. Consequently, $win_{A^{max}} = win_{A_{\pi^*}}$ and A_{π^*} is a solution of Problem **III.1**.

VI. Solution of the Multi-Objective Problem

A. Construction of the MDP

As was mentioned before, Problems **III.1** and **III.2** can be solved in sequence. In this manner, MDP M is trimmed such that only strategies that are a solution of Problem **IV.3** remain. Based on this trimmed MDP, we are going to relate Problem **IV.4** to Problem **III.2** as in Section V.

In Problem **IV.3**, we obtain the maximum probability of reaching the absorbing state set Q_{abs} from the initial state of an MDP. In fact, we also obtain the maximum probability of reaching the absorbing set for each $q \in Q$, i.e., $p_{M,q}^*(Q_{abs})$.

Back to our specific MDP M as in Definition **V.3**, we obtain $p_{M,q}^*(Obj)$ for all $q \in Q$ by solving Problem **IV.3** for the absorbing state set Obj. The trimmed MDP is defined based on actions that guarantee the optimal value $p_{M,q}^*(Obj)$ in state $q \in Q$. Namely, we use the principle of optimality to identify optimal actions at any given state $q \in Q$. We also create a fictitious state called q_{abs} that aggregates states of Q such that $p_{M,q}^*(Obj) = 0$, i.e., states that cannot reach the set Obj. Formally, the trimmed MDP is defined as follows:

Definition VI.6. Given the MDP M as in Def. **V.3**, the set Obj, and the values $p_{M,q}^*(Obj)$ obtained by solving Problem **IV.3**, we construct the trimmed MDP $M_{tr} = (Q_{tr}, Act, \delta_{tr}, q_{0,tr})$ as follows:

- $Q_{tr} \subseteq Q \cup \{q_{abs}\}$ is defined as $Q_{tr} = \cup_{j\geq 1} Q_j \cup \{q_{abs}\}$ for Q_j defined recursively as:

$$\begin{aligned} Q_1 &= \{q_{0,M}\} \\ Q_{j+1} &= \{q \in Q \mid \delta_M(q',a^*,q) > 0 \text{ for } q' \in Q_j, \\ &\qquad a^* \in \arg\max_{a\in Act} N(q',a),\ p_{M,q}^*(Obj) \neq 0\} \end{aligned}$$

  where $N(q',a) = \sum_{q\in Q} p_{M,q}^*(Obj)\delta_M(q',a,q)$. By the principle of optimality $\max_{a\in Act} N(q',a) = p_{M,q'}^*(Obj)$.
- The set $Act = \{\tau, nd\} \cup \Sigma_i \cup \Sigma_d$.
- The initial state $q_{0,tr} = q_{0,M}$.
- The PTF δ_{tr} is defined for $x,y \in Q_{tr} \setminus \{q_{abs}\}$ and action $a \in Act$:

$$\delta_{tr}(x,a,y) = \delta_M(x,a,y) \tag{21}$$

$$\delta_{tr}(x,\tau,q_{abs}) = 1 - \sum_{q\in Q_{tr}\setminus\{q_{abs}\}} \delta_{tr}(x,\tau,q) \tag{22}$$

$$\delta_{tr}(q_{abs},\tau,q_{abs}) = 1 \tag{23}$$

***Remark* VI.5.** Equation (22) is only defined for the τ action since all other actions occur with probability 1 as defined in Eqs. (11-12).

Example VI.5. Let M be the MDP depicted in Fig. 3, where $p_{M,q}^*$ for $q \in Q$ is given as in Example **V.4**. In this manner,

we construct M_{tr} as specified in Def. **VI.6**. The trimmed MDP is depicted in Fig. 5. States $(1,3,\varepsilon)$, $(2,3,\varepsilon)$, $(1,5,rW)$, $(1,5,rW)$, $(2,5,rE)$, and $(2,5,\varepsilon)$ are not part of Q_{tr} since they are reached by nonoptimal actions. On the other hand, states $(3,2,rN)$ and $(1,4,rS)$ are replaced by state q_{abs} since they are reachable by an optimal action but both cannot reach the set Obj.

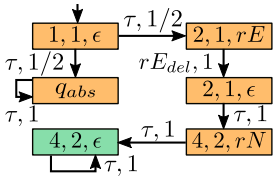


Fig. 5: Trimmed MDP M_{tr}

In order to have a complete instance of Problem **IV.4** for M_{tr}, a cost function must be defined. Since we want to connect Problem **IV.4** to Problem **III.2**, we define c based on the weight function w for events in Σ_m.

$$c(q,a) = \begin{cases} w(a) & \text{if } a \in \Sigma_e^a \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

We finalize this section by providing important properties of MDP M_{tr}. The first property states that there exists a strategy in $\Pi_{M_{tr}}$ that reaches the set $Obj \cup \{q_{abs}\}$ with probability 1, i.e., there exists at least one proper strategy. Recall that Problem **IV.4** has a finite solution if and only if it has at least one proper strategy with respect to Q_{abs}. The second property ties the maximum probability of reaching the set Obj in M with the probability of reaching Obj in M_{tr} with a proper strategy. Its proof is in Appendix I-B.

Proposition VI.5. Given M_{tr}, then:
(1) $\max_{\pi\in\Pi_{M_{tr}}} p^{\pi}_{M_{tr}}(Obj \cup \{q_{abs}\}) = 1$;
(2) $p^{\pi}_{M_{tr}}(Obj) = p^{*}_{M}(Obj)$ for any $\pi \in \Pi_{M_{tr}}$ such that $p^{\pi}_{M_{tr}}(Obj \cup \{q_{abs}\}) = 1$.

Proposition **VI.5** states that Problem **IV.4** for M_{tr}, $Obj \cup \{q_{abs}\}$, and c as Eq. (24) has a finite solution. Moreover, the strategy that provides the optimal solution for this problem can be extended to M and provide an optimal solution for Problem **IV.3**.

B. Solution Procedure

We follow a similar methodology as in Section V but we connect Problem **III.2** to Problem **IV.4**. Once an attack function A that is a solution of Problem **III.2** is fixed, we obtain the language $\mathcal{L}(S_A/H)$ and consequently the value of $\mathbb{E}^{A}[cost(s)]$. The same idea applies to the MDP M_{tr}; once a proper strategy is fixed, we can calculate the expected cumulative cost value $\mathbb{E}^{\pi}_{M_{tr}}[cumul(Obj \cup \{q_{abs}\}, \rho)]$. The construction of MDP M_{tr} and the set Obj ties these two problems together. The following theorem connects the solution of Problem **IV.4** for M_{tr}, the set $Q_{abs} = Obj \cup \{q_{abs}\}$, and the cost function c with the solution of Problem **III.2**.

Theorem VI.2. Consider the MDP M_{tr}, the set $Q_{abs} = Obj \cup \{q_{abs}\}$, and the cost function c, then

$$\mathbb{E}^{A_{multi}}[cost(s)] = c^{*}_{M_{tr}}(Q_{abs}) \tag{25}$$

Again, the proof of the above theorem needs to be written with care even though it appears plausible that the solution of Problem **IV.4** provides a solution for Problem **III.2**. We intuitively provided intermediate results in the following subsection and their formal proofs are attached in Appendix I-B. Similar to Problem **IV.3**, the solution procedure of Problem **IV.4** not only outputs the minimum expected cost $c^{*}_{M_{tr}}(Q_{abs})$ but also outputs the minimum cost $c^{*}_{M_{tr},q}(Q_{abs})$ for each $q \in Q$. In this manner, one can use standard methods to extract an optimal strategy π^{*} that achieves $c^{*}_{M_{tr}}(Q_{abs})$. Therefore, an attack function A^{multi} based on π^{*} such that A^{multi} is a solution of Problem **III.2** can be constructed based on Def. **V.5**.

Example VI.6. We return to our running example, where we solve Problem **IV.4** for the MDP depicted in Fig. 5. For simplicity, we assume that $c(q,a) = 2$ if $a \in \Sigma_a^e$. In this case, only one strategy remains in MDP M_{tr}, i.e., only one strategy reaches Obj with probability 0.5. It follows that $c^{*}_{M_{tr}}(Q_{abs}) = 2 \times 0.5 = 1$, which implies that $\mathbb{E}^{A^{multi}}[cost(s)] = 1$. The attack function A^{multi} is the same as A^{max} defined in Example **V.4**. This expected cost is the attacker's average cost to reach the critical state (set Obj) or to reach a state where the critical state is unreachable (state q_{abs}).

C. Discussion on Theorem VI.2

Since M_{tr} is constructed directly from M, it is clear that any strategy in M_{tr} can be extended to a strategy in M. With an abuse of notation, we apply Defs. **V.4** and **V.5** for strategies in $\Pi_{M_{tr}}$ instead of Π^{f}_{M}.

Recall that in Section III-C, we assumed that an attacker stops its attack once a string in Uns is executed (the attacker reached the critical state), or a string in Unr is executed (the attacker cannot reach the critical state at this point). In order to prove Theorem **VI.2**, we first show that there is a one-to-one map between the strings in $Uns \cup Unr$ to finite runs in M_{tr} that reach the absorbing set Q_{abs}. Strings in Uns are mapped to finite runs that reach the objective set Obj, whereas strings in Unr are mapped to finite runs that reach state q_{abs}. Let the set R^{π}_{abs} represent the set of all finite runs that reach Q_{abs}. Formally, this set is defined as:

$$R^{\pi}_{abs} = \{\rho \in Pref^{\pi}(M_{tr}) \mid q^{|\rho|} \in Q_{abs} \wedge q^{|\rho|-1} \notin Q_{abs}\} \tag{26}$$

Proposition VI.6. Given a proper strategy $\pi \in \Pi_{M_{tr}}$ with respect to Q_{abs}, there exists a bijection between R^{π}_{abs} and $Uns \cup Unr$ constructed based on A_{π}. Conversely, given an attack function $A \in \Pi_A$ such that $win_A = win_{A^{max}}$, there exists a bijection between $R^{\pi_A}_{abs}$ and $Uns \cup Unr$.

Based on Prop. **VI.6**, we relate the expected cost for attacked systems with the expected cost for MDPs. Let $c^{\pi}_{M_{tr}}(Q_{abs}) = \mathbb{E}^{\pi}_{M_{tr}}[cumul(Q_{abs}, \rho)]$ denote the expected cost given strategy $\pi \in \Pi_{M_{tr}}$.

Proposition VI.7. Given a strategy $\pi \in \Pi_{M_{tr}}$ such that π is proper with respect to $Q_{abs} = Obj \cup \{q_{abs}\}$, then $c^{\pi}_{M_{tr}}(Q_{abs}) = \mathbb{E}^{A_\pi}[cost(s)]$. Conversely, given a solution of Problem **III.1** $A \in \Pi_A$, then $\mathbb{E}^{A}[cost(s)] = c^{\pi_A}_{M_{tr}}(Q_{abs})$.

Finally, the proof of Theorem **VI.2** follows directly from Props. **VI.5** and **VI.7**. Let $\pi^* \in \Pi_{M_{tr}}$ be a strategy such that $c^{\pi^*}_{M_{tr}}(Q_{abs}) = c^{*}_{M_{tr}}(Q_{abs})$, then $c^{\pi^*}_{M_{tr}}(Q_{abs}) = \mathbb{E}^{A_{\pi^*}}[cost(s)]$ and $p^{\pi^*}_{M}(Obj) = win_{A^{max}}$. It follows that for any $A \in \Pi_A$ such that $win_A = win_{A^{max}}$, $\mathbb{E}^{A}[cost(s)] = c^{\pi_A}_{M_{tr}}(Q_{abs}) \geq c^{*}_{M_{tr}}(Q_{abs}) = c^{\pi^*}_{M_{tr}}(Q_{abs}) = \mathbb{E}^{A_{\pi^*}}[cost(s)]$. Consequently, $\mathbb{E}^{A^{multi}}[cost(s)] = \mathbb{E}^{A_{\pi^*}}[cost(s)]$ and $A^{multi} = A_{\pi^*}$ is a solution of Problem **III.2**.

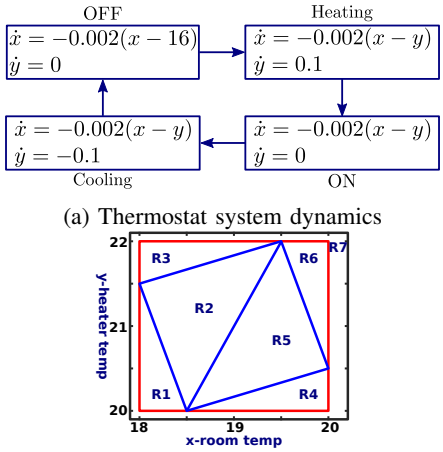
VII. Example

We developed a tool to automatically construct the MDP M, described in Definition **V.3**, as input to the PRISM model checker [55] and to construct an attack function based on the optimal strategy outputted by PRISM. PRISM has support for solving both MDP Problems **IV.3** and **IV.4**. In fact, PRISM supports multi-objective strategy generation [56]. The tool includes efficient model checking engines, e.g., binary decision diagrams, and supports different MDP solution methods, e.g., linear programming, value iteration, etc. Our evaluation was done on a Linux machine with 2.2GHz CPU and 16GB memory. Our software tool and the complete models of the example described in this section are available at: `https://gitlab.eecs.umich.edu/M-DES-tools/desops`.

We consider the thermostat system described in [57]. The system has four modes with dynamics described by differential equations as depicted in Fig. 6(a), where x denotes the room temperature and y denotes the temperature of the heater. Moreover, it transitions among the different modes in the order shown in 6(a). This system is abstracted to a discrete model based on the partition plane with 7 regions as shown in Fig. 6(b). The discrete model has 7 uncontrollable events (R1-R7) and 4 controllable events (OFF, Heating, ON, Cooling). The transition relation among the regions in each mode is obtained based on its dynamics.

Differently from [57], we consider a probabilistic transition relation among these regions, where the probability captures the likelihood of transitioning between regions. In [57], a supervisor guarantees that the controlled system *never* reaches region 7 (R7). A simulation result illustrating a continuous implementation of this supervisor is shown in Fig. 7(a). The system starts in the ON mode in region 6 (R6). Moreover, the plant has 62 states and the supervisor has 20 states.

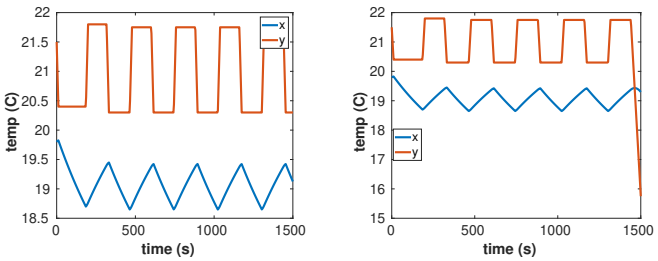
First, we synthesize a maximal reachability attack function assuming that the attacker can manipulate both the room temperature and the heater temperature. As expected, there exists an attack function that reaches region 7 with probability one. One attack function is depicted in Fig. 8. The attacker waits for the system to reach region 4 and then deletes the reading R4 and replaces it by R5 until the system reaches R7. A simulation result illustrating a continuous implementation of this attacked system is shown in Fig. 7(b). The attack starts at time 1375 and by time 1460 the controlled system has reached region 7.



(a) Thermostat system dynamics

(b) State space partition

Fig. 6: Four-mode thermostat system



(a) Simulation without attack

(b) Simulation with max. reach. attack

Fig. 7: Four-mode thermostat simulation

Next we consider the synthesis of an attack function based on multiple-objectives (Section VI). We consider that each attack modification has weight one, i.e., $w(e) = 1$ if $e \in \Sigma^e_a$. In this manner, we can synthesize an attack function that reaches region 7 with probability one and with an expected cost of 8.5, i.e., it takes on average 8.5 edits until the system reaches region 7.

In this example, the MDP M has 427 states while M_{tr} has 22 states. Moreover, it takes 3.50 seconds to obtain a solution of Problem **III.1** while it takes 7.93 seconds to obtain a solution of Problem **III.2**.

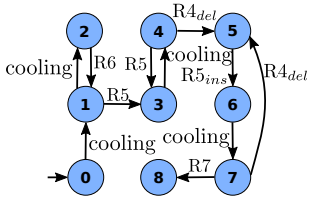


Fig. 8: Max. reach. attack function encoded as automaton

VIII. Conclusion

We have considered the synthesis of attack functions for sensor deception attacks at the supervisory layer of feedback control systems, where the system is modeled as a probabilistic finite-state automaton controlled by a given deterministic

supervisor. Given the stochastic nature of the system model, attack functions are quantified by the likelihood of reaching the critical state. We investigated the problem of synthesizing attack functions with the maximum likelihood of reaching the critical state. In order to constrain the attacker behavior, we posed a second problem where each attack edit is costly. In the second problem, the attacker has two objectives: (i) reach the critical state with the maximum likelihood; (ii) minimize the expected cost while performing (i).

In order to solve these problems, we leveraged techniques from MDPs. We reduced these problems to two well-known problems in MDPs: the probabilistic reachability problem and the stochastic shortest path problem. Finally, we presented a computational example to demonstrate the results of our paper.

In the future, it would be of interest to investigate defense measures for the supervisor to prevent such attacks from succeeding. In a similar manner, we could further improve the synthesis of supervisors by quantifying them based on similar likelihood criteria as the ones adopted in this paper.

APPENDIX I
PROOFS

A. Section V

Proof of Lemma **V.1**.

Proof. Let $\rho \in Runs^{\pi}(M)$ such that $(\exists j \in \mathbb{N})(\forall k \geq j)[\pi(q^0 \dots q^k) \in \Sigma_i]$. Then $p^{\pi}_{M,q^j}(Obj) < 1$ otherwise $\exists k \geq j$ such that $\pi(q^0 \dots q^k) = \tau$. It implies that $q^j \notin Obj$ and $q^j_H \neq x_{crit}$. Moreover, $\pi(q^0 \dots q^k) \in \Sigma_i$ for all $k \geq j$, then by construction of M we have $\delta_M(q^k, \pi(q^0 \dots q^k), q^{k+1}) = 1$. Again by construction of M, $q^k_H \neq x_{crit}$ for all $k \geq j$. It follows that $p^{\pi}_{M,q^k}(Obj) = 0$. □

Proof of Proposition **V.1**

Proof. We show that if $\pi \in \Pi_M$ is optimal and it has infinite insertions, then we can construct a $\pi^f \in \Pi^f_M$ based on π that is also optimal. Assume $\pi \in \Pi_M \setminus \Pi^f_M$ and $p^{\pi}_M(Obj) = p^*_M(Obj)$. Let I be the set of runs $\rho \in Runs^{\pi}(M)$ that satisfies $(\exists j \in \mathbb{N})(\forall k \geq j)[\pi(q^0 \dots q^k) \in \Sigma_i]$. By Lemma **V.1**, for any $\rho \in I$ then there exists $j \in \mathbb{N}$ such that $p^{\pi}_{M,q^j}(Obj) = 0$. This implies that we can modify π for prefixes of runs in I without altering the probability of reaching Obj. Without loss of generality, we can assume that π is a memoryless strategy [1], [49]. Let π^f be defined in the following manner for any $q \in Q$: $\pi^f(q) = \tau$, if $\forall \rho \in Runs^{\pi}_q(M).a^j \in \Sigma_i$ for all $j \in \mathbb{N}$; $\pi^f(q) = \pi(q)$, otherwise. It follows that $\pi^f \in \Pi^f_M$ and $p^{\pi^f}_M(Obj) = p^{\pi}_M(Obj) = p^*_M(Obj)$. □

Proof of Proposition **V.2**.

Proof. We use induction on the length of s to prove the result the existence of $\rho \in Pref(M)$.

Induction basis: $s = \varepsilon \in \mathcal{L}(S_A/H)$ and $t = A(\varepsilon)$.

By the definition of A, the string t is finite. Let $a^{j-1} = t[j]$ for $j \in [|t|]^+$, by Def. **III.1** each $t[j] \in \Sigma_i$. Also, let $q^j = (x_{0,H}, \delta_R(x_{0,R}, t^j), \varepsilon)$ for $j \in [|t|]$. Each $q^j \in Q$ since A is consistent, $\delta_R(x_{0,R}, t^j)!$, and by construction of M. Thus, $\delta_M(q^j, a^j, q^{j+1}) = 1$ by construction of M. Therefore, $\rho = q^0 a^0 \dots q^{|t|} \in Pref(M)$, $q^{|t|}_e \notin \Sigma_a$, $s_\rho = s$ and $t_\rho = t$.

Induction hypothesis: The proposition holds for all $s \in \mathcal{L}(S_A/H)$ with $|s| \leq n$.

Induction step: Let $|s| = n + 1$.

From the induction hypothesis, $\exists \rho \in Pref(M)$ such that the proposition holds for s^n. Recall that $A(s) = A(s^n)A(s^n, s[n+1])$. Let $t^* = A(s^n, s[n+1])$, then it is enough to show that $\exists \rho^* = q^{*0}a^{*0} \dots q^{*|\rho^*|} \in Pref_{q^{|\rho|}}(M)$ such that $s[n+1] = q^{*1}_e = q^{*1}_e \dots q^{*|\rho^*|}_e$ and $t^* = t_{\rho^*}$. It is enough to find ρ^*, because we concatenate ρ with $a^{*0} \dots q^{*|\rho^*|}$ to produce the finite run of $A(s)$, where the first state q^{*0} is removed since $q^{*0} = q^{|\rho|}$. Based on Def. **III.1**, $t^*[1] \in \Sigma \cup \Sigma_d$ and $\mathcal{M}(t^*[1]) = s[n+1]$.

Assume that $t^*[1] \in \Sigma \setminus \Sigma_a$ and let $a^{*0} = \tau$. Since $s \in \mathcal{L}(S_A/H)$ and by the induction hypothesis we have:

$$\begin{aligned}\delta_H(x_{0,H}, s) &= \delta_H(\delta_H(x_{0,H}, s^n), s[n+1]) = \delta_H(q^{*0}_H, t^*[1]) \\ &:= q_H \\ \delta_R(x_{0,R}, A(s^n)t^*[1]) &= \delta_R(q^{*0}_R, t^*[1]) \\ &:= q_R\end{aligned}$$

Thus, $\delta_M(q^{*0}, \tau, q^{*1}) > 0$ by construction of M, where $q^{*1} = (q_H, q_R, s[n+1])$. Since $t^*[j+1] \in \Sigma_i$ for $j \in [|t^*|-1]^+$ by Def. **III.1**, we can apply the same strategy as in the induction basis. Namely, $a^{*j+1} = t^*[j+2]$ and $q^{*j+2} = (q_H, \delta_R(q_R, t^{*j+2}), \varepsilon) \in Q$ for $j \in [|t^*|-2]$. Again since A is consistent, it follows that $\delta_M(q^{*j+1}, a^{*j+1}, q^{*j+2}) = 1$ for $j \in [|t^*|-2]$. In this manner $\rho^* = q^{*0}a^{*0} \dots q^{*|t^*|} \in Pref_{q^{|\rho|}}(M)$ and $\rho a^{*0} \dots q^{*|t^*|} \in Pref(M)$ such that $s_{\rho a^{*0} \dots q^{*|t^*|}} = s$ and $t_{\rho a^{*0} \dots q^{*|t^*|}} = A(s^n)t^* = A(s)$. Similar arguments can be made when $t^*[1] \in \Sigma_a \cup \Sigma_d$. In this case, $t^*[1]$ defines $q^{*0}, q^{*1}, q^{*2}, a^{*0}$ and a^{*1} while q^{*j+3} and a^{*j+2} are defined based on $t^*[j+2]$ for $j \in [|t^*|-2]$. This concludes the first assertion of the proof.

The condition $(a^j = \tau \Rightarrow q^{j+1}_e \neq \varepsilon)$ for any $j \in [|\rho|]$ is satisfied by the construction of ρ in the induction proof. Finally, uniqueness follows from the fact that A, H and R are deterministic. □

Proof of Proposition **V.3**.

Proof. We start with Eq. (19). The equality trivially holds when $s = \varepsilon$. Let $s \in \mathcal{L}(S_A/H)$ such that $|s| > 0$, then

$$L_p(S_A/H)(s) = \prod_{j \in [|s|]^+} Pr^{\delta_H(x_{0,H}, s^{j-1}), \delta_R(x_{0,R}, A(s^{j-1}))}_{s[j]} \tag{27}$$

Using Def. **V.4**, let $\rho \in Pref^{\pi_A}(M)$ such that $s = s_\rho$ and $t_\rho = A(s)$. Such ρ always exists since we can construct as in Def. **V.4**. Then for $j \in [|s|]$ there exists $k \geq j$ such that $\delta_H(x_{0,H}, s^j) = q^k_H$, $\delta_R(x_{0,R}, A(s^j)) = q^k_R$, $s^j = s_{q^0 a^0 \dots q^k}$, and $A(s^j) = t_{q^0 a^0 \dots q^k}$. It follows that:

$$L_p(S_A/H)(s) \overset{\text{Def. V.3}}{=} \prod_{j \in [|\rho|-1] \text{ s.t. } q^{j+1}_e \in \Sigma} Pr^{q^j_H, q^j_R}_{q^{j+1}_e} \tag{28}$$

$$\overset{\text{Def. V.3}}{=} \prod_{j \in [|\rho|-1]} \delta_M(q^j, a^j, q^{j+1}) \tag{29}$$

$$\overset{\text{Eq. (18)}}{=} Pr_M^{\pi_A}(\langle\rho\rangle) \tag{30}$$

This concludes the first part of the proof.

We now show Eq. (20). The equality trivially holds for $\rho \in Pref^\pi(M)$ such that $\pi(\rho) = \tau$ and $(\forall j \in [|\rho|])[q_e^j = \varepsilon]$. Let $\rho \in Pref^\pi(M)$ such that $\pi(\rho) = \tau$ and $(\exists j \in [|\rho|])[q_e^j \in \Sigma]$

$$Pr_M^\pi(\langle\rho\rangle) \overset{\text{Eq. (18)}}{=} \prod_{j\in[|\rho|-1]} \delta_M(q^j, a^j, q^{j+1}) \tag{31}$$

$$\overset{\text{Def. V.3}}{=} \prod_{j\in[|\rho|-1] \text{ s.t. } q_e^{j+1}\in\Sigma} Pr_{q_e^{j+1}}^{q_H^j, q_R^j} \tag{32}$$

Using Def. **V.3** and Def. **V.5**, we have that for $j \leq |\rho|$ such that $q_e^j \in \Sigma$, $q_H^{j-1} = \delta_H(x_{0,H}, s_{q^0a^0\ldots q^{j-1}})$ and $q_R^{j-1} = \delta_R(x_{0,R}, t_{q^0a^0\ldots q^{j-1}}) = \delta_R(x_{0,R}, A_\pi(s_{q^0a^0\ldots q^{j-1}}))$ and $s_\rho \in \mathcal{L}(S_{A_\pi}/H)$. Hence

$$= \prod_{\substack{j\in[|\rho|-1] \text{ s.t.} \\ q_e^{j+1}\in\Sigma}} Pr_{q_e^{j+1}}^{\delta_H(x_{0,H}, s_{q^0a^0\ldots q^j}), \delta_R(x_{0,R}, A_\pi(s_{q^0a^0\ldots q^j}))} \tag{33}$$

$$= \prod_{j\in[|s_\rho|-1]} Pr_{s_\rho[j+1]}^{\delta_H(x_{0,H}, s_\rho^j), \delta_R(x_{0,R}, A_\pi(s_\rho^j))} = L_p(S_{A_\pi}/H)(s_\rho) \tag{34}$$

This concludes our proof. □

Proof of Proposition **V.4**.

Proof. First, we show that $p_M^\pi(Obj) = win_{A_\pi}$ for any strategy $\pi \in \Pi_M^f$. Define the set $Win^\pi = \{\rho \in Pref^\pi(M) \mid q^{|\rho|} \in Obj \wedge q^{|\rho|-1} \notin Obj\}$. Let $\mathbb{1}_A(x)$ for any given set A denote the indicator function, i.e., $\mathbb{1}_A(x) = 1$ if $x \in A$ and $\mathbb{1}_A(x) = 0$ otherwise. Then, the value $p_M^\pi(Obj)$ can be obtained by:

$$p_M^\pi(Obj) = \mathbb{E}_M^\pi[\mathbb{1}_{Win^\pi}(\rho)] \tag{35}$$

Next, we show that there is a one-to-one map between Win^π and Uns. For that, let $L_W := \{s \in \mathcal{L}(H) \mid \exists \rho \in Win^\pi.\ s_\rho = s\}$. Using Def. **V.5**, it can be shown that $L_W = Uns$. Therefore, we need to show that any $s \in L_W$ is generated by a unique run $\rho \in Win^\pi$. But, this last statement is true, because the strategy π and the plant H are deterministic. Now, since Win^π is countable, it follows:

$$p_M^\pi(Obj) \overset{\text{Eq.(35)}}{=} \sum_{\rho\in Win^\pi} Pr_M^\pi(\langle\rho\rangle) \tag{36}$$

$$\overset{\text{Def.V.5}}{=} \sum_{s\in Uns} L_p(S_{A_\pi}/H)(s) \tag{37}$$

$$= \sum_{s\in U_{uns}} L_p(S_{A_\pi}/H)(s) \overset{\text{Eq.(2)}}{=} win_{A_\pi} \tag{38}$$

In the same manner, we can show that for a given attack function A and its derived strategy π_A, the equality $win_A = p_M^{\pi_A}(Obj)$ holds. The steps are the same as in the first part of the proof. □

B. Section VI

Proof of Proposition **VI.5**

Proof. We first prove (1). Let $\pi \in \Pi_M^f$ such that $p_{M,q}^\pi(Obj) = p_q^*(Obj)$ for all $q \in Q$. Similar to the proof of Lemma **V.1**, we can show that for any $\rho \in Runs^\pi(M)$ either $\exists j \in \mathbb{N}$ such that $q^j \in Obj$ or $p_{M,q^j}^\pi(Obj) = 0$, or the set of runs that do not satisfy the first condition has probability zero of being generated. By construction of M_{tr} and since π is optimal, we can convert π to be in $\Pi_{M_{tr}}$ and $p_{M_{tr}}^\pi(Obj \cup \{q_{abs}\}) = 1$. It follows that $\max_{\pi\in\Pi_{M_{tr}}} p_{M_{tr}}^\pi(Obj \cup \{q_{abs}\}) = 1$. The second property follows from (1) and the construction of M_{tr}. □

Proof of Proposition **VI.6**

Proof. Let us define the set $L_{abs}^\pi = \{s \in \mathcal{L}(H) \mid s = s_\rho \text{ for } \rho \in R_{abs}^\pi\}$. It is enough to show $L_{abs}^\pi = Unr \cup Uns$ since each $\rho \in R_{abs}^\pi$ generate a unique string in L_{abs}^π. The last statement follows since π is deterministic.

First, we show that $L_{abs}^\pi \subseteq Unr \cup Uns$. Assume that $s \in L_{abs}^\pi$ is generated by $\rho \in R_{abs}^\pi$ such that $q^{|\rho|} = q_{abs}$. Since $p_{M,q_{abs}}^\pi(Obj) = 0$ and $p_{M,q^{|\rho|-1}}^\pi(Obj) > 0$, it follows that $s \in Unr$. Now, assume that $s \in L_{abs}^\pi$ is generated by $\rho \in R_{abs}^\pi$ such that $q^{|\rho|} \in Obj$. Then $q_H^{|\rho|} = x_{crit}$, and, by Def. **V.5**, $s \in Uns$.

Showing that $Unr \cup Uns \subseteq L_{abs}^\pi$ follows the same steps. Let $s \in Unr \cup Uns$, then $A_\pi(s)$ is bounded since π is proper. Using Prop. **V.2** and Def. **V.5**, $\exists \rho \in Pref^\pi(M_{tr})$ such that $s_\rho = s$ and $t_\rho = A_\pi(s)$. If $s \in Unr$, then $q^{|\rho|} = q_{abs}$ which implies that $s \in L_{abs}^{\pi_A}$. If $s \in Uns$, then $\rho' = \rho\tau(q_H^{|\rho|}, q_R^{|\rho|}, \varepsilon) \in R_{abs}^\pi$ and $s_{\rho'} = s_\rho\varepsilon = s \in L_{abs}^\pi$.

The second equality $Unr \cup Uns \cap \mathcal{L}(S_A/H) = L_{abs}^{\pi_A}$ follows in the same manner. □

Proof of Proposition **VI.7**

Proof. Given $\pi \in \Pi_{M_{tr}}$, π being a proper strategy, then since R_{abs}^π is countable:

$$c_{M_{tr}}^\pi(Q_{abs}) = \mathbb{E}_{M_{tr}}^\pi[cumul(Q_{abs}, \rho)] \tag{39}$$

$$\overset{p_{M_{tr}}^\pi(Q_{abs})=1}{=} \mathbb{E}_{M_{tr}}^\pi[\sum_{j=0}^{\min\{k:q^k\in Q_{abs}\}} c(q^j, a^j)] \tag{40}$$

$$\overset{\text{Eq.(26)}}{=} \sum_{\rho\in R_{abs}^\pi} \sum_{j=0}^{|\rho|-1} c(q^j, a^j) Pr_{M_{tr}}^\pi(\langle\rho\rangle) \tag{41}$$

$$\overset{\text{Prop.V.3}}{=} \sum_{\rho\in R_{abs}^\pi} \sum_{j=0}^{|\rho|-1} c(q^j, a^j) L_p(S_{A_\pi}/H)(s_\rho) \tag{42}$$

$$\overset{\substack{\text{Prop.VI.6} \\ \text{Eq.(24)}}}{=} \sum_{s\in L_{abs}^\pi} \sum_{j=1}^{|A(s)|} w(A(s)[j]) L_p(S_{A_\pi}/H)(s) \tag{43}$$

$$\overset{\text{Eq.(4)}}{=} \sum_{s\in L_{abs}^\pi} cost(s) L_p(S_{A_\pi}/H)(s) \tag{44}$$

$$= \mathbb{E}^{A_\pi}[cost(s)] \tag{45}$$

The equality $\mathbb{E}^A[cost(s)] = c^{\pi_A}_{M_{tr}}(Q_{abs})$ follows by reversing the previous steps, i.e., starting from Eq. (45) to Eq. (39). Reversing the previous steps, we can show that for a given attack function A and its derived strategy π_A, then $\mathbb{E}^A[cost(s)] = c^{\pi_A}_{M_{tr}}(Q_{abs})$. □

REFERENCES

[1] L. de Alfaro, "Computing minimum and maximum reachability times in probabilistic systems," in *CONCUR'99 Concurrency Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 66–81.

[2] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. II*, 3rd ed. Athena Scientific, 2007.

[3] J. N. Tsitsiklis, "On the control of discrete-event dynamical systems," in *26th IEEE Conference on Decision and Control*, vol. 26, 1987, pp. 419–422.

[4] A. Rashidinejad, B. Wetzels, M. Reniers, L. Lin, Y. Zhu, and R. Su, "Supervisory control of discrete-event systems under attacks: An overview and outlook," in *2019 18th European Control Conference (ECC)*, June 2019, pp. 1732–1739.

[5] R. Meira-Góes, E. Kang, R. Kwong, and S. Lafortune, "Stealthy deception attacks for cyber-physical systems," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec. 2017, pp. 4224–4230.

[6] R. Meira-Góes, E. Kang, R. H. Kwong, and S. Lafortune, "Synthesis of sensor deception attacks at the supervisory layer of cyber–physical systems," *Automatica*, vol. 121, p. 109172, 2020.

[7] R. Meira-Góes, R. Kwong, and S. Lafortune, "Synthesis of sensor deception attacks for systems modeled as probabilistic automata," in *2019 American Control Conference (ACC)*, July 2019.

[8] A. Condon, "The complexity of stochastic games," *Information and Computation*, vol. 96, no. 2, pp. 203 – 224, 1992.

[9] R. Su, "Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations," *Automatica*, vol. 94, pp. 35 – 44, 2018.

[10] Q. Zhang, Z. Li, C. Seatzu, and A. Giua, "Stealthy attacks for partially-observed discrete event systems," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2018, pp. 1161–1164.

[11] Y. Wang and M. Pajic, "Supervisory control of discrete event systems in the presence of sensor and actuator attacks," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 5350–5355.

[12] L. Lin, S. Thuijsman, Y. Zhu, S. Ware, R. Su, and M. Reniers, "Synthesis of supremal successful normal actuator attackers on normal supervisors," in *2019 American Control Conference (ACC)*, 2019, pp. 5614–5619.

[13] D. Thorsley and D. Teneketzis, "Intrusion detection in controlled discrete event systems," in *Proceedings of the 45th IEEE Conference on Decision and Control*, Dec. 2006, pp. 6047–6054.

[14] L. K. Carvalho, Y. C. Wu, R. Kwong, and S. Lafortune, "Detection and mitigation of classes of attacks in supervisory control systems," *Automatica*, vol. 97, pp. 121 – 133, 2018.

[15] P. M. Lima, M. V. S. Alves, L. K. Carvalho, and M. V. Moreira, "Security against communication network attacks of cyber-physical systems," *Journal of Control, Automation and Electrical Systems*, vol. 30, no. 1, pp. 125–135, Feb. 2019.

[16] Z. Wang, R. Meira-Góes, S. Lafortune, and R. Kwong, "Mitigation of classes of attacks using a probabilistic discrete event system framework," in *15th IFAC Workshop on Discrete Event Systems WODES 2020*, 2020.

[17] R. Meira-Góes, C. Keroglou, and S. Lafortune, "Towards probabilistic intrusion detection in supervisory control of discrete event systems," in *21st IFAC World Congress (to appear)*, 2020.

[18] Y. Zhu, L. Lin, R. Tai, and R. Su, "Overview of networked supervisory control with imperfect communication channels," *arXiv:2010.11491*, 2020.

[19] K. Rohloff, "Bounded sensor failure tolerant supervisory control," *11th IFAC Workshop on Discrete Event Systems*, vol. 45, no. 29, pp. 272 – 277, 2012.

[20] F. Lin, "Control of networked discrete event systems: Dealing with communication delays and losses," *SIAM Journal on Control and Optimization*, vol. 52, no. 2, pp. 1276–1298, 2014.

[21] F. Wang, S. Shu, and F. Lin, "Robust networked control of discrete event systems," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 1528–1540, 2016.

[22] M. V. S. Alves, A. E. C. da Cunha, L. K. Carvalho, M. V. Moreira, and J. C. Basilio, "Robust supervisory control of discrete event systems against intermittent loss of observations," *International Journal of Control*, vol. 0, no. 0, pp. 1–13, 2019.

[23] F. Lin, "Robust and adaptive supervisory control of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 38, no. 12, pp. 1848–1852, Dec. 1993.

[24] J. Cury and B. Krogh, "Robustness of supervisors for discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 44, no. 2, pp. 376–379, 1999.

[25] S. Takai, "Maximizing robustness of supervisors for partially observed discrete event systems," *Automatica*, vol. 40, no. 3, pp. 531 – 535, 2004.

[26] S. Xu and R. Kumar, "Discrete event control under nondeterministic partial observation," in *2009 IEEE International Conference on Automation Science and Engineering*, Aug. 2009, pp. 127–132.

[27] M. Wakaiki, P. Tabuada, and J. P. Hespanha, "Supervisory control of discrete-event systems under attacks," *Dynamic Games and Applications*, Sep. 2018.

[28] Y. Wang and M. Pajic, "Attack-resilient supervisory control with intermittently secure communication," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 2015–2020.

[29] R. Meira-Góes, S. Lafortune, and H. Marchand, "Synthesis of supervisors robust against sensor deception attacks," *IEEE Transactions on Automatic Control*, 2021.

[30] R. Meira-Góes, H. Marchand, and S. Lafortune, "Towards resilient supervisors against sensor deception attacks," in *2019 IEEE 58th Annual Conference on Decision and Control (CDC)*, Dec. 2019.

[31] L. Lin, Y. Zhu, and R. Su, "Towards bounded synthesis of resilient supervisors," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 7659–7664.

[32] Y. Zhu, L. Lin, and R. Su, "Supervisor obfuscation against actuator enablement attack," in *2019 18th European Control Conference (ECC)*, 2019, pp. 1760–1765.

[33] A. Saboori and C. N. Hadjicostis, "Notions of security and opacity in discrete event systems," in *46th IEEE Conference on Decision and Control*, Dec 2007, pp. 5056–5061.

[34] F. Lin, "Opacity of discrete event systems and its applications," *Automatica*, vol. 47, no. 3, pp. 496–503, Mar. 2011.

[35] J. Dubreil, P. Darondeau, and H. Marchand, "Supervisory control for opacity," *IEEE Transactions on Automatic Control*, vol. 55, no. 5, pp. 1089–1100, 2010.

[36] F. Cassez, J. Dubreil, and H. Marchand, "Synthesis of opaque systems with static and dynamic masks," *Formal Methods in System Design*, vol. 40, no. 1, pp. 88–115, 2012.

[37] Y.-C. Wu and S. Lafortune, "Synthesis of insertion functions for enforcement of opacity security properties," *Automatica*, vol. 50, no. 5, pp. 1336 – 1348, 2014.

[38] R. Jacob, J.-J. Lesage, and J.-M. Faure, "Overview of discrete event systems opacity: Models, validation, and quantification," *Annual Reviews in Control*, vol. 41, pp. 135 – 146, 2016.

[39] L. Hélouet, H. Marchand, and L. Ricker, "Opacity with powerful attackers," *14th IFAC Workshop on Discrete Event Systems WODES 2018*, vol. 51, no. 7, pp. 464 – 471, 2018, 14th IFAC Workshop on Discrete Event Systems WODES 2018.

[40] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, Jan. 1987.

[41] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2008.

[42] R. Kumar and V. K. Garg, "Control of stochastic discrete event systems modeled by probabilistic languages," *IEEE Transactions on Automatic Control*, vol. 46, no. 4, pp. 593–606, Apr. 2001.

[43] V. K. Garg, R. Kumar, and S. I. Marcus, "A probabilistic language formalism for stochastic discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 44, no. 2, pp. 280–293, Feb. 1999.

[44] M. Lawford and W. M. Wonham, "Supervisory control of probabilistic discrete event systems," in *Proceedings of 36th Midwest Symposium on Circuits and Systems*, Aug. 1993, pp. 327–331.

[45] V. Pantelic, S. M. Postma, and M. Lawford, "Probabilistic supervisory control of probabilistic discrete event systems," *IEEE Transactions on Automatic Control*, vol. 54, no. 8, pp. 2013–2018, 2009.

[46] W. M. Wonham and K. Cai, *Supervisory Control of Discrete-Event Systems*. Springer International Publishing, 2018.

[47] Y. Ji, X. Yin, and S. Lafortune, "Opacity enforcement using nondeterministic publicly known edit functions," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4369–4376, 2019.

[48] C. Derman, *Finite State Markovian Decision Processes*. Orlando, FL, USA: Academic Press, Inc., 1970.

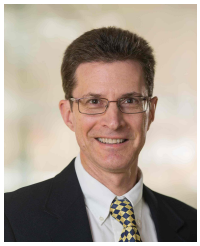
[49] H. Kushner, *Introduction to Stochastic Control*. Holt, Rinehart and Winston, 1971.

[50] A. Kolobov, Mausam, and D. S. Weld, "A theory of goal-oriented mdps with dead ends," in *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'12, 2012, pp. 438–447.
[51] B. Lacerda, D. Parker, and N. Hawes, "Optimal policy generation for partially satisfiable co-safe ltl specifications," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, p. 1587–1593.
[52] W. Feller, *An introduction to probability theory and its applications*, 3rd ed., ser. Wiley series in probability and mathematical statistics. Wiley, 1967.
[53] J. R. Norris, *Markov Chains*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.
[54] A. Condon, "On algorithms for simple stochastic games," in *Advances in Computational Complexity Theory, volume 13 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1993, pp. 51–73.
[55] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 585–591.
[56] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, and H. Qu, "Quantitative multi-objective verification for probabilistic systems," in *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 112–127.
[57] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 58, no. 7, pp. 1771–1785, July 2013.

Rômulo Meira-Góes received the B.Eng degree from Universidade Tecnológica Federal do Paraná-Curitiba in 2015, the M.Eng degree from the University of Michigan in 2017, all in electrical engineering, and the Ph.D. degree in Electrical and Computer Engineering at the University of Michigan. As of January 2021, he is a Postdoctoral Researcher with joint appointments at the Institute for Software Research at the Carnegie Mellon University and the Department of Electrical Engineering and Computer Science at the University of Michigan. His research interests include supervisory control of discrete event systems, formal methods and game theory, specially, their application in cyber security of cyber-physical systems.

Raymond H. Kwong received the S.B., S.M. and Ph.D. degrees in Electrical Engineering from the Massachusetts Institute of Technology, Cambridge, in 1971, 1972, and 1975, respectively.

From 1975 to 1977, he was a visiting Assistant Professor of Electrical Engineering at McGill University and a Research Associate at the Centre de Recherches Mathématiques, Université de Montréal, Montreal, Canada. Since August 1977, he has been with the Edward S. Rogers Sr. Department of Electrical and Computer Engineering at the University of Toronto, where he is a Professor. He has held visiting professor positions at University of Rennes, University of Maryland, and University of Michigan. His current research interests include estimation and stochastic control, fault diagnosis and fault-tolerant control, discrete event systems, and cyber-security of control systems.



Stéphane Lafortune received the B.Eng degree from École Polytechnique de Montréal in 1980, the M.Eng degree from McGill University in 1982, and the Ph.D degree from the University of California at Berkeley in 1986, all in electrical engineering. Since September 1986, he has been with the University of Michigan, Ann Arbor, where he is the N. Harris McClamroch Collegiate Professor of Electrical Engineering and Computer Science. Dr. Lafortune is a Fellow of the IEEE (1999) and of IFAC (2017). He received the Presidential Young Investigator Award from the National Science Foundation in 1990 and the Axelby Outstanding Paper Award from the Control Systems Society of the IEEE in 1994 (for a paper co-authored with S.-L. Chung and F. Lin) and in 2001 (for a paper co-authored with G. Barrett). Dr. Lafortune's research interests are in discrete event systems and include multiple problem domains: modeling, diagnosis, control, optimization, and applications to computer and software systems.