**UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC**
**CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT**
**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA – PPGEEL**

**SAMUEL SILVA DE OLIVEIRA**

# PROTECTION AND ROBUST RECOVERY STRATEGIES FOR CYBER-PHYSICAL DISCRETE EVENT SYSTEMS UNDER ACTIVE ATTACKS

**JOINVILLE**
**2026**

**SAMUEL SILVA DE OLIVEIRA**

# PROTECTION AND ROBUST RECOVERY STRATEGIES FOR CYBER-PHYSICAL DISCRETE EVENT SYSTEMS UNDER ACTIVE ATTACKS

Tese submetida ao Programa de Pós–Graduação em Engenharia Elétrica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Doutor em Engenharia Elétrica.

Orientador: André Bittencourt Leal
Coorientador: Marcelo Teixeira

**JOINVILLE**
**2026**

**Ficha catalográfica elaborada pelo programa de geração automática da**

**Biblioteca Universitária Udesc,**

**com os dados fornecidos pelo(a) autor(a)**

Oliveira, Samuel
Protection and Robust Recovery Strategies for Cyber-Physical Discrete Event Systems under Active Attacks / Samuel Oliveira. -- 2026.
100 p.

Orientador: André Bittencourt Leal
Coorientador: Marcelo Teixeira
Tese (doutorado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Engenharia Elétrica, Joinville, 2026.

1. Sistemas ciberfísicos. 2. Sistemas a eventos discretos. 3. Ciberataques. 4. Protetibilidade viva. 5. Recuperação robusta. I. Bittencourt Leal, André. II. Teixeira, Marcelo. III. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Engenharia Elétrica. IV. Titulo.

**SAMUEL SILVA DE OLIVEIRA**

**PROTECTION AND ROBUST RECOVERY STRATEGIES FOR CYBER-PHYSICAL DISCRETE EVENT SYSTEMS UNDER ACTIVE ATTACKS**

Tese submetida ao Programa de Pós–Graduação em Engenharia Elétrica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Doutor em Engenharia Elétrica.

**BANCA EXAMINADORA:**

Prof. Dr. André Bittencourt Leal
Universidade do Estado de Santa Catarina (UDESC)

Membros:

Prof. Dr. Benjamin Grando Moreira
Universidade Federal de Santa Catarina (UFSC)

Prof. Dr. Lucas Vinícius Ribeiro Alves
Universidade Federal de Minas Gerais (UFMG)

Prof. Dr. Marcos Vicente de Brito Moreira
Universidade Federal do Rio de Janeiro (UFRJ)

Prof. Dr. Rômulo Meira-Góes
Pennsylvania State University (PSU) - EUA

Joinville, 25 de Fevereiro de 2026

A Deus, fonte de todo conhecimento e sabedoria, que tudo criou e sustenta. À minha esposa Tháfinys, à minha filha Alice e aos meus pais Paulo e Marilene, pelo apoio incondicional nesta jornada.

## AGRADECIMENTOS

Agradeço a Deus pela vida repleta de graça e misericórdia.

À minha esposa Tháfinys e à minha filha Alice, que têm me acompanhado e pacientemente me apoiado durante todos esses anos de estudo. Amo vocês!

Aos meus pais, Paulo e Marilene, que sempre me incentivaram e ativamente me acompanharam com conversas, orações e conselhos.

Aos demais familiares — minhas irmãs Viviane e Tatiana, meus sogros, meus tios e tias, primos e primas — agradeço imensamente pelo apoio e incentivo constantes.

Aos muitos amigos que fiz em Joinville, tantos que nem posso nomear, que ao longo desses anos de estudo caminharam ao meu lado, oferecendo apoio, encorajamento e edificação na fé em Jesus.

Agradeço especialmente ao Prof. Dr. André Bittencourt Leal pela confiança depositada em mim ao aceitar me orientar. Ele assumiu o desafio de me guiar na área de Sistemas a Eventos Discretos (SEDs), mesmo sabendo que eu iniciaria minha imersão nesse campo somente no doutorado. Sua experiência, paciência, generosidade e amizade foram determinantes para o desenvolvimento deste trabalho e para meu amadurecimento como professor e pesquisador.

Ao Prof. Dr. Marcelo Teixeira, meu coorientador, agradeço pela constante disponibilidade, pelas valiosas contribuições na escrita dos artigos e pelos feedbacks criteriosos ao longo do desenvolvimento da pesquisa. Seu acompanhamento atento e suas sugestões, oferecidos à medida que os avanços iam sendo alcançados, foram fundamentais para a consolidação dos resultados apresentados neste trabalho.

Ao Prof. Dr. Rômulo Meira-Góes, que gentilmente me recebeu na Universidade do Estado da Pensilvânia (The Pennsylvania State University - PSU) através do Programa de Doutorado Sanduíche no Exterior (PDSE/CAPES), expresso minha profunda gratidão. As reuniões semanais de orientação, as aulas sobre SEDs, os encontros do grupo de pesquisa e o cotidiano na PSU me proporcionaram uma experiência enriquecedora, que se estendeu também à minha família. Obrigado por todo apoio e generosidade! Estendo meus agradecimentos aos colegas do Casa Góes Lab que me acolheram com tanta cordialidade e paciência, sempre dispostos ao diálogo mesmo diante das minhas limitações iniciais com o idioma.

Aos colegas de laboratório, em especial ao Gabriel Abatti e ao Paulo Feller, agradeço pela colaboração e amizade. Os momentos de estudo, conversas e descontração foram fundamentais e marcaram esta jornada.

Aos professores Yuri e Karine, bem como ao colega Anderson, agradeço pela participação no grupo de conversação em Inglês. Esses encontros foram muito positivos para o aprimoramento da minha comunicação no idioma e, ao mesmo tempo, proporcionaram momentos agradáveis de convivência, amizade e descontração.

À Universidade Federal do Amapá (UNIFAP), pela concessão do afastamento para qualificação, que possibilitou minha dedicação integral ao doutorado.

Ao CNPq, pela bolsa de doutorado concedida de outubro de 2023 a fevereiro de 2026.

À CAPES, pela bolsa do Programa de Doutorado Sanduíche no Exterior, que me permitiu realizar parte do doutorado nos Estados Unidos entre novembro de 2024 e abril de 2025.

“He is no fool who gives what he cannot keep to gain what he cannot lose.” (Elliot, 1978)

## RESUMO

A conectividade demandada pela Indústria 4.0 impõe novos desafios para os Sistemas Ciberfísicos, do Inglês, Cyber-Physical Systems (CPS), em particular, o reforço de sua resiliência contra ataques cibernéticos. Esta tese de doutorado aborda os desafios de proteção e recuperação em Sistemas Ciberfísicos a Eventos Discretos (CPDES) sujeitos a ataques ativos. Embora as estratégias tradicionais de mitigação dependam frequentemente da detecção de ataques, tais abordagens podem falhar ao enfrentar ataques ocultos ou em sistemas que não conseguem evitar comportamentos inseguros uma vez que um ataque tenha sido detectado. Esta pesquisa identifica e aborda duas lacunas principais na literatura atual: a proteção de sistemas contra ataques ocultos e a subsequente recuperação de sistemas sob ataques. A primeira parte desta tese introduz uma função de edição de atuadores projetada para garantir a integridade e a continuidade do sistema sob ataques ocultos. Ao acoplar esta função de edição à camada de controle supervisório, o mecanismo intervém nos comandos de controle para evitar que o sistema atinja estados inseguros. A propriedade de *protetibilidade viva* é formalizada, fornecendo as condições necessárias e suficientes para proteger um sistema contra ataques ocultos a sensores e atuadores enquanto se garante a continuidade operacional. Um algoritmo de síntese é proposto para computar tais funções de edição, resolvendo efetivamente o problema de verificação associado. A segunda parte da pesquisa move o foco da preservação da integridade para a recuperação. Apresentamos um framework de recuperação robusta focado em ataques de habilitação de atuadores (AE), considerando também o impacto de ataques de desabilitação de atuadores (AD) durante o processo de recuperação. Este framework introduz a propriedade de *recuperabilidade robusta*, que garante que um sistema possa ser conduzido a uma região alvo, definida como um subconjunto de estados para operação pós-ataque, apesar da interferência contínua de atacantes. Um algoritmo de síntese para computar supervisores maximalmente permissivos, não bloqueantes e robustamente recuperáveis é apresentado. A eficácia deste framework é demonstrada por meio de um estudo de caso baseado na Simulação de Fábrica Fischertechnik 24V™, que confirma a viabilidade prática da recuperação robusta mesmo sob ataques persistentes a atuadores.



**Palavras-chave**: Sistemas ciberfísicos; Sistemas a eventos discretos; Ciberataques; Protetibilidade viva; Recuperação robusta.

**ABSTRACT**

The connectivity demanded by Industry 4.0 poses new challenges for cyber-physical systems (CPS), particularly in enhancing their resilience against cyberattacks. This doctoral thesis addresses the challenges of protection and recovery in Cyber-Physical Discrete Event Systems (CPDES) under active attacks. While traditional mitigation strategies often rely on attack detection, they might fail when facing covert (undetectable) attacks or in systems that cannot prevent unsafe behaviors once an attack is detected. This research identifies and addresses two primary gaps in the current literature: the protection of systems against covert attacks and the recovery of attacked systems. The first part of this thesis introduces an actuator edit function designed to ensure system integrity and liveness under covert attacks. By coupling this edit function to the supervisory control layer, the mechanism intervenes in control commands to prevent the system from reaching unsafe states. We formalize the property of *live protectability*, providing the necessary and sufficient conditions to protect a system against covert sensor and actuator attacks while guaranteeing liveness. A synthesis algorithm is proposed to compute these edit functions, effectively solving the associated verification problem. The second part of the research shifts from integrity preservation to recovery. We present a robust recovery framework particularly focused on Actuator Enablement (AE) attacks while also considering the impact of Actuator Disablement (AD) attacks during recovery. This framework introduces the property of *robust recoverability*, which ensures that a system can be driven to a target region, i.e., a subset of states defined for post-attack operation, despite persistent interference from attackers. We provide a synthesis algorithm to compute maximally permissive, nonblocking, and robustly recoverable supervisors. The effectiveness of our robust recovery framework is illustrated in a case study based on the Fischertechnik Factory Simulation 24V™, demonstrating the practical feasibility of robust recovery even under persistent actuator enablement and disablement attacks.

**Keywords**: Cyber-Physical Systems; Discrete Event Systems; Cyberattacks; Live protectability; Robust Recovery.

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| DES | Discrete Event Systems |
| CPS | Cyber-Physical Systems |
| CPDES | Cyber-Physical Discrete Event Systems |
| SCT | Supervisory Control Theory |
| DoS | Denial of Service |
| AD | Actuator disablement |
| AE | Actuator enablement |
| SD | Sensor deletion |
| SI | Sensor insertion |
| GF | General form |
| FST | Finite-state Transducers |
| PSEP | Performance Safety Enforcing Problem |
| ABSRA | Attack with Bounded Sensor Reading Alterations |
| IEEE | Institute of Electrical and Electronics Engineers |
| IEC | International Electrotechnical Commission |
| NIST | National Institute of Standards and Technology |
| PDSE | Programa de Doutorado Sanduíche no Exterior |
| CAPES | Coordenação de Aperfeiçoamento de Pessoal de Nível Superior |
| GASR | Grupo de Automação de Sistemas e Robótica |
| CAIS | Control and Automation for Intelligent Systems |
| UDESC | Universidade do Estado de Santa Catarina |
| PSU | Pennsylvania State University |

## LIST OF SYMBOLS

| | |
|---|---|
| $G$ | Deterministic finite-state automaton |
| $X$ | Set of states |
| $\Sigma$ | Set of events |
| $f$ | Transition function |
| $x_0$ | Initial state |
| $X_m$ | Set of marked/final states |
| $\Gamma$ | Active event set |
| $\sigma$ | Event symbol ($\sigma \in \Sigma$) |
| $\Sigma^*$ | Set of all finite-length strings formed by events in $\Sigma$ |
| $\varepsilon$ | Empty string (string with no events) |
| $\mathcal{L}$ | Language |
| $\mathcal{L}(G)$ | Language generated by $G$ |
| $\mathcal{L}_m(G)$ | Language marked by $G$ |
| $\overline{\mathcal{L}}$ | Prefix closure of $L$ |
| $\Sigma_c$ | Set of controllable events |
| $\Sigma_{uc}$ | Set of uncontrollable events |
| $\Sigma_o$ | Set of observable events |
| $\Sigma_{uo}$ | Set of unobservable events |
| $\mathcal{S}$ | Supervisor (function) |
| $S$ | Supervisor (automaton) |
| $f_S$ | Transition function of $S$ |
| $S/G$ | Closed-loop system ($S$ controlling $G$) |
| $\mathcal{L}(S/G)$ | Language generated by $S/G$ |
| $\mathcal{L}_m(S/G)$ | Language marked by $S/G$ |
| $K$ | Desired behavior |
| $\overline{K}$ | Prefix closure of $K$ |
| $SupC(K, L(G))$ | Supremal controllable sublanguage of $K$ w.r.t. $L(G)$ |
| $G_1 \| G_2$ | Parallel composition between automata $G_1$ and $G_2$ |
| $P_o$ | Natural projection |
| $P_o^{-1}$ | Inverse projection |

| | |
|---|---|
| $\mathcal{M}$ | Mask function |
| $\Sigma_{c,d}$ | Set of potentially damaging events |
| $\Sigma^a_{c,d}$ | Set of attacked events |
| $\sigma^a$ | Attacked event symbol ($\sigma^a \in \Sigma^a_{c,d}$) |
| $G^a$ | Modified plant model |
| $S^a$ | Modified supervisor model |
| $f_{S^a/G^a}$ | Transition function of $S^a/G^a$ |
| $\mathcal{S}(s)$ | Enabled events after string $s$ |
| $\neg\mathcal{S}(s)$ | Disabled events after string $s$ |
| $\Upsilon^s$ | Set of events that, from a particular state, lead to a safe state |
| $\Upsilon^{us}$ | Set of events that, from a particular state, lead to an unsafe state |
| $H$ | Control specification |
| $X_H$ | Set of states of $H$ |
| $\Sigma_{all}$ | Complete set of events of an attacked system |
| $f_H$ | Transition function of $H$ |
| $x_{0,H}$ | Initial state of $H$ |
| $X_{m,H}$ | Set of marked states of $H$ |
| $X_{us}$ | Set of unsafe states |
| $\widehat{X}_{us}$ | Set of states from which an unsafe state is uncontrollably reachable |
| $X_{safe}$ | Set of safe states |
| $\mathcal{F}$ | Edit function |
| $\Gamma_H(x)$ | Active event set of $H$ at state $x$ |
| $\Sigma^x_{bad}$ | Set of events that lead to unsafe states from a given state $x$ |
| $\Sigma^x_{good}$ | Set of events that lead to safe states from a given state $x$ |
| $\textsc{CoAc}(\cdot)$ | Coaccessible operation |
| $X_{coac}$ | Set of coaccessible states |
| $Ac(G)$ | Accessible part of $G$ |
| $G_{robust}$ | Automaton representing the robust region |
| $G_{target}$ | Automaton representing the target region |
| $\mathcal{L}_d(G)$ | Language of damage |
| $\mathcal{L}(G)/s$ | Post-language of $L(G)$ after string $s$ |

| | |
|---|---|
| $\mathcal{L}_{min}$ | Shortest strings reaching $G_{target}$ |
| $x_d$ | A detection state |
| $X_{det}$ | Set of detection states |
| $K_{x_d}$ | Robust recovery language from state $x_d$ |
| $\mathcal{R}_{x_d}$ | Recovery strategy from a detection state |

**CONTENTS**

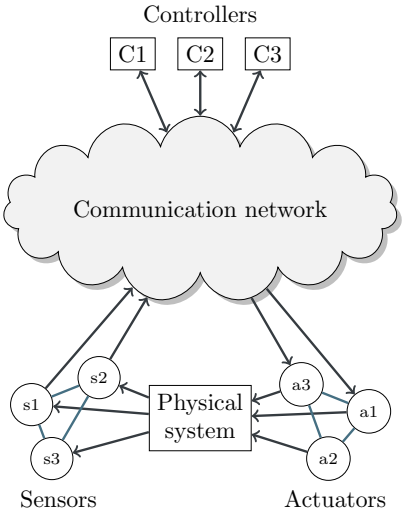
# 1 INTRODUCTION

Cyber-physical systems (CPS) play a crucial role in modern automation, including applications such as advanced manufacturing plants, communication systems, smart power grids, and transportation networks. In these systems, communication networks integrate control logic with physical devices, which typically consist of sensors and actuators. This connectivity, however, also makes such systems vulnerable to cyberattacks that can cause them to deviate from their intended behavior. These deviations may result in severe damage to infrastructure and equipment, leading to significant operational costs and safety risks.

The Discrete Event Systems (DES) and Supervisory Control Theory (SCT) frameworks have been widely used to address cybersecurity problems in CPS. These frameworks effectively model both normal and attack-induced interactions among system components using discrete events and state transitions. They also provide a rigorous mathematical foundation for formally specifying safety and security properties and for analyzing system behavior under attacks. In this thesis, we refer to this DES-based representation of CPS as a cyber-physical discrete event system (CPDES) (Tong; Wang; Giua, 2022; Fritz; Zhang, 2023; Oliveira et al., 2024a; Oliveira et al., 2025).

A general architecture of a CPDES is depicted in Figure 1. In this architecture, control activities ($C1, C2, C3$) may be centralized in a single controller or distributed among multiple controllers. The communication network is used to collect data from the physical processes via sensors ($s1, s2, s3$) and to issue control commands to actuators ($a1, a2, a3$). Based on this information exchange, the desired closed-loop behavior is maintained over the network.

Figure 1 – General architecture of a CPDES.




Source: (Oliveira et al., 2023a).

## 1.1 MOTIVATION

The architecture shown in Figure 1 highlights that CPDES rely on communication networks for feedback and control. While this integration enables high levels of automation, it also introduces critical vulnerabilities.

The motivation for this thesis arises from two fundamental challenges in the cybersecurity of CPDES. The first challenge concerns the particular class of covert cyberattacks. Traditional attack mitigation strategies rely on detection mechanisms to initiate mitigation procedures. However, covert attacks are specifically designed to remain undetectable by controllers or, in the context of SCT, by the supervisor. By strategically manipulating sensor and actuator signals, an attacker can drive the physical plant toward unsafe behaviors while ensuring that the observations received by the supervisor remain consistent with normal operation. Because the attack remains covert within the expected behavior of the system, detection-based mechanisms are never activated, leaving the system vulnerable to physical damage.

Beyond the preservation of system integrity, a second and equally critical challenge concerns the recovery of an attacked system. In real-world scenarios, once an attack occurs, it usually disrupts the behavior expected for the system. After such an attack is mitigated or blocked, it is essential to drive the system back to a functional post-attack mode of operation. Developing mechanisms that can steer the system toward a designated target region, even in the presence of persistent interference from attackers, is therefore crucial. By addressing the recovery problem, it becomes possible to significantly enhance system resilience and guarantee availability, ensuring that the system remains operational despite the presence of adversarial actions.

Several real-world cyberattacks targeting both system integrity and availability are reported by Dibaji et al. (2019). Among them, we highlight the Stuxnet attack, which compromised an Iranian uranium enrichment facility in 2009. In this event, attackers manipulated the physical devices, specifically gas centrifuges, to cause physical damage while simultaneously sending false sensor data to the controllers to mask the malicious activity. This attack serves as a landmark example of how a covert attack can bypass traditional detection to violate the integrity of critical systems.

In contrast, the cyberattack on Ukrainian power distribution networks in 2015 provides a significant example of a threat to system availability and the challenges related to recovery. During this incident, attackers remotely manipulated circuit breakers, interrupting power supply to hundreds of thousands of customers and rendering the system unable to autonomously recover, thereby prolonging the outage. This event illustrates that, beyond preserving system integrity, the ability to restore an attacked system to a functional operational mode is essential for ensuring resilience. Taken together, these historical cases reveal fundamental limitations of detection-based and recovery approaches, motivating the

development of control-theoretic frameworks that ensure protection against covert attacks and robust recovery under persistent adversarial interference, as developed in this doctoral thesis.

## 1.2 RELATED WORKS

Cybersecurity challenges have attracted considerable attention in the literature. When seen as a DES problem, the cybersecurity of CPDES usually falls into the following categories: *attack modeling* (Meira-Góes et al., 2020; Lin; Zhu; Su, 2020; Zhang et al., 2022; Tai et al., 2023; Chen; Su; Li, 2025; Ma; Giua; Seatzu, 2025a), *attack detection and mitigation* (Thorsley; Teneketzis, 2006; Carvalho et al., 2018; Wang et al., 2020; Yao; Yin; Li, 2020; Lima et al., 2022; Oliveira et al., 2024a; Fahim; Meira-Goes, 2024; Tong; Cai; Seatzu, 2024; Fahim; Oliveira; Meira-Góes, 2025), *synthesis of resilient supervisors* (Su, 2018; Meira-Góes; Lafortune; Marchand, 2021; Alves; Pena; Rudie, 2022; Ma; Cai, 2022a; Meira-Góes; Marchand; Lafortune, 2023; Wang et al., 2023; Xie et al., 2025), and *system recovery* (Alves; Pena, 2022; Sakata et al., 2023; Sakata et al., 2024; Cavalcanti et al., 2025). Comprehensive reviews can also be found in the literature (Rashidinejad et al., 2019; Cao et al., 2020; Hadjicostis et al., 2022; Oliveira et al., 2023a).

In *attack modeling*, the perspective of the intruder is considered to formulate successful attacks on the system integrity, generally assuming a non-risky profile until executing a covert attack. A covert attack, also called stealthy attack, involves stealthy maneuvers within communication channels to compromise the underlying physical processes by forcing the system to execute unsafe behaviors while avoiding detection of such malicious activities (Alves; Rudie; Pena, 2024; Mainhardt et al., 2024).

The *attack detection and mitigation* strategy focuses on providing methods to detect the occurrence of active attacks and mitigate their effect (Fritz; Zhang, 2023). This strategy is also in line with the principles of fault-tolerant supervisory control (Paoli; Sartini; Lafortune, 2011; Moor, 2016; Moreira; Leal, 2020; Watanabe et al., 2022), where the system must continue to operate safely after faults occur and also avoid entering unsafe states. Nevertheless, the emphasis here is on detecting attacks before an unsafe state is reached and applying protective measures to avert damage. This goal is typically achieved by identifying events that should be disabled or enabled after attack detection (Carvalho et al., 2018; Yao; Yin; Li, 2020; Wang et al., 2020; Lima et al., 2022; Oliveira et al., 2024a). However, while these methods preserve system integrity, they do not guarantee continued system operation after an attack. In fact, they may admit behaviors whose sole objective is integrity preservation, including intentional system shutdowns, which violate system availability.

To ensure safety guarantees on system behavior, some works investigate the synthesis of resilient supervisors. These methods usually impose restrictions on the closed-loop system

behavior to ensure safety properties (Ma; Cai, 2022a; Meira-Góes; Marchand; Lafortune, 2023). Such restrictions remain active regardless of whether attacks are actually occurring, representing an overly conservative approach in the absence of attacks. In this context, Zhu, Lin and Su (2019) proposed an algorithm that aims to obfuscate a supervisor to make it resilient against covert actuator attacks, while preserving the behavior of the original closed-loop system. However, validating non-attackability for each possible system behavior can significantly degrade algorithm performance, requiring efficiency improvements.

System recovery methods focus on restoring system operation after deviations in the closed-loop behavior induced by attacks (Alves; Pena, 2022; Cavalcanti et al., 2025). Alves and Pena (2022) introduced a secure recovery mechanism using synchronizing automata. The method addresses desynchronization between the supervisor and the plant caused by malicious attacks. Recovery is achieved by guiding the system back to its initial state. This approach, however, does not enhance the system's resilience to future attacks, as the original vulnerabilities remain unchanged. Addressing the same challenge, Cavalcanti et al. (2025) propose a method for synthesizing a non-blocking supervisor that leads an attacked system back to its nominal closed-loop behavior while avoiding unsafe states. However, this approach relies on the assumption that the attacker is immediately isolated upon detection, meaning no further attacks can occur.

An alternative recovery strategy explores *fallback control systems*. In the works of Sakata et al. (2023), Sakata et al. (2024), post-attack operation is achieved by transitioning to a backup controller that remains connected to the network and is assumed to be immune to attacks. This method successfully restores nominal operation after attacks, similarly to the method proposed by Cavalcanti et al. (2025). However, it fundamentally depends on the existence of such an attack-proof fallback controller, which is a considerably restrictive assumption.

Despite significant progress in the field of cybersecurity of CPDES, two key gaps remain in the literature. First, existing methods either fail to provide integrity guarantees for CPDES subject to covert attacks, or achieve such guarantees at the cost of significant computational complexity, limiting their practical applicability. Second, current recovery strategies typically assume attacker isolation, depend on attack-proof fallback controllers, or do not address recovery under persistent attacks, highlighting the need for new methods that enable robust recovery without relying on such strong and restrictive assumptions.

## 1.3 PROBLEM STATEMENT

Motivated by the challenges discussed in the previous sections, this thesis addresses two fundamental problems in vulnerable CPDES:

**Problem 1: Integrity preservation under covert attacks.** The first problem concerns the protection of CPDES against covert attacks or, more generally, CPDES that are unable

to avoid unsafe behavior once an attack is detected. In this setting, malicious attackers may alter the system behavior without being detected by the supervisor, rendering detection-based mitigation strategies ineffective. The fundamental challenge is to design control mechanisms that guarantee the prevention of unsafe behavior even when the occurrence of an attack cannot be formally inferred by the control layer.

**Problem 2: Robust recovery of attacked CPDES.** The second problem concerns the recovery of CPDES after attack detection. Beyond avoiding unsafe states, recovery requires guaranteeing that an attacked system can be driven to a safe and functional post-attack mode of operation. The challenge is to achieve recovery under persistent attacks, meaning that the attacker continues to interfere with the system during the recovery phase. The problem is therefore to formally characterize the conditions under which such recovery is achievable and to synthesize control strategies that ensure system availability and operational continuity despite continued attacks.

## 1.4 OBJECTIVES

The primary objective of this work is twofold: (i) to develop protection mechanisms against covert attacks in CPDES; and (ii) to develop robust recovery strategies that guarantee safe post-attack operation of CPDES, even under persistent attacks.

To achieve this overall objective, the following specific objectives are pursued:

- Develop cybersecurity techniques for:
  1. integrity preservation in CPDES under covert attacks, and
  2. robust recovery of attacked CPDES;
- Formalize necessary and sufficient conditions associated with both protection and recovery techniques;
- Design and implement algorithms and conduct simulations using DES modeling software to validate the proposed frameworks;
- Develop formal theorems and provide rigorous proofs supporting the proposed frameworks;
- Apply the proposed techniques to practical applications in order to bridge the gap between theoretical developments and real-world systems.

## 1.5 MAIN CONTRIBUTIONS

The main contributions of this doctoral thesis are summarized as follows:

1. A systematic literature review that classifies existing cybersecurity strategies in the context of DES, highlighting areas where further investigation is needed. This work was originally published in *Annual Reviews in Control*, v. 56, p. 100907, 2023 (Oliveira et al., 2023a).

2. The formalization of actuator-side protection mechanisms based on event substitution to preserve system integrity in vulnerable CPDES;

3. The definition of the property of *live protectability*, which characterizes necessary and sufficient conditions under which a CPDES can be protected against covert attacks while preserving liveness through the use of actuator edit functions;

4. The development of a synthesis algorithm for computing global live-protecting actuator edit functions;

5. The formalization of robust recovery strategies that guide attacked CPDES to a designated target region while remaining resilient to further attacks, including actuator disablement attacks during the recovery process;

6. A theoretical characterization of necessary and sufficient conditions for robust recoverability in CPDES;

7. The formulation of the supervisor synthesis problem for robust recovery, establishing how robust recovery strategies can be enforced through supervisory control;

8. The development of a synthesis algorithm for maximally permissive, nonblocking, and robustly recoverable supervisors within the Ramadge–Wonham supervisory control framework (Ramadge; Wonham, 1987);

9. A case study based on a manufacturing testbed using the Fischertechnik factory simulation to illustrate and validate the proposed recovery framework.

## 1.6 TEXT ORGANIZATION

This doctoral thesis is organized as follows. Chapter 2 introduces the basic concepts of automata and languages used in the supervisory control of discrete event systems, as well as fundamental notions of cyberattacks against DES. Chapter 3 presents a literature review of the main topics addressed in this thesis. Chapter 4 presents the main framework for integrity and liveness preservation in CPDES under covert sensor and actuator attacks. Chapter 5 presents the main framework for robust recovery of attacked CPDES. Finally, Chapter 6 concludes the thesis and outlines directions for future research.

# 2 PRELIMINARY CONCEPTS

Formal methods are widely employed to model DES, as they effectively capture the event-triggered transitions between the discrete states of a system. This chapter provides the theoretical foundations of automata and languages, which are commonly used for modeling DES, and presents main concepts of Supervisory Control Theory (SCT). Furthermore, we introduce the main concepts on the cybersecurity of DES with respect to the characteristics of cyberattacks found in the literature.

## 2.1 AUTOMATA & LANGUAGES

**Definition 1** (Deterministic Finite-State Automaton)**.** *(Cassandras; Lafortune, 2021) A deterministic finite-state automaton may be represented by a six-tuple*

$$G = (X, \Sigma, f, \Gamma, x_0, X_m)$$

*where*

$X$ *denotes the finite set of states;*

$\Sigma$ *is the set of events;*

$f : X \times \Sigma \to X$ *denotes the possibly partial transition function;*

$\Gamma : X \to 2^{\Sigma}$ *is the active event function, as* $\Gamma(x)$ *represents all events* $\sigma$ *for which* $f(x, \sigma)$ *is defined;*

$x_0$ *is the initial state;*

$X_m$ *is the set of marked states, such that* $X_m \subseteq X$*.*

If an automaton $G$ includes a state that is unreachable through any sequence of events from the initial state, that particular state is termed non-accessible. The operation $Ac(G)$ eliminates all non-accessible states from $G$, along with their corresponding transitions. When the result of this operation is equivalent to $G$, the automaton is said to be accessible.

When modeling systems composed of interacting components, the event set of each component usually includes both private events related to its internal behavior and shared events with other automata reflecting the connection between the system components. The conventional approach to obtain a unified system model is the so-called parallel composition, in which the models of the individual components are combined. In this method, the parallel composition of the automata $G_1$ and $G_2$ leads to the formation of the automaton

$$G_1 \| G_2 = Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f, \Gamma_{1\|2}, (x_{0,1}, x_{0,2}), X_{m,1} \times X_{m,2})$$

where

$$f((x_1,x_2),\sigma) := \begin{cases} (f_1(x_1,\sigma), f_2(x_2,\sigma)) & \text{if } \sigma \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (f_1(x_1,\sigma), x_2) & \text{if } \sigma \in \Gamma_1(x_1) \setminus \Sigma_2 \\ (x_1, f_2(x_2,\sigma)) & \text{if } \sigma \in \Gamma_2(x_2) \setminus \Sigma_1 \\ \text{undefined} & \text{otherwise.} \end{cases}$$

A finite-length sequence of events in $\Sigma$ is named a string, and the set of all strings formed by events in $\Sigma$ is denoted by $\Sigma^*$, including the empty string $\varepsilon$ (string with no events). Thus, the transition function $f$ can be naturally extended to sequences of events. The extended transition function $f : X \times \Sigma^* \to X$ is defined recursively by $f(x,\varepsilon) = x$ and $f(x,s\sigma) = f(f(x,s),\sigma)$ whenever $f(x,s)$ and $f(f(x,s),\sigma)$ are defined, for all $x \in X$, $s \in \Sigma^*$, and $\sigma \in \Sigma$. In the sequel, we present the definition of languages.

**Definition 2** (Language)**.** *(Cassandras; Lafortune, 2021) A language defined over an event set $\Sigma$ is a set of finite-length strings formed from any subset of $\Sigma^*$.*

A string $u$ is considered a prefix of a string $s$ if $u$ forms the initial part of $s$. Then, if $r$ and $s$ are strings in $\Sigma^*$, $u$ is a prefix of $s$ if $ur = s$. For a language $\mathcal{L}$, the notation $\overline{\mathcal{L}}$, the so-called prefix closure of $\mathcal{L}$, represents the set of all the prefixes of all the strings in $\mathcal{L}$. $\mathcal{L}$ is said to be prefix closed if $\mathcal{L} = \overline{\mathcal{L}}$.

A DES may be modeled by an automaton $G$ used to represent and manipulate two languages: the language generated by $G$ and the language marked by $G$. The language generated by $G$ is defined as $\mathcal{L}(G) := \{s \in \Sigma^* \mid f(x_0,s) \text{ is defined}\}$. On the other hand, the language marked by $G$ is defined as $\mathcal{L}_m(G) := \{s \in L(G) \mid f(x_0,s) \in X_m\}$. In other words, the language $\mathcal{L}_m(G)$ is the subset of the language $\mathcal{L}(G)$ that contains all strings whose transition function leads to a marked state, representing the behavior of G in which tasks are completed. The automaton $G$ is said to be nonblocking if $\overline{\mathcal{L}_m(G)} = \mathcal{L}(G)$. That is, $G$ is nonblocking if there is always a sequence of events which takes the plant from any reachable state to a marked state. Notation $\mathcal{L}(G)/s$ is adopted herein to mean the post-language of $\mathcal{L}(G)$ after the sequence $s$, i.e., $\mathcal{L}(G)/s := \{t \in \Sigma^* : st \in \mathcal{L}(G)\}$.

Given two automata $G_1$ and $G_2$, $G_2$ is a *strict subautomaton* of $G_1$, denoted $G_2 \sqsubset G_1$, if $G_2$ is a copy of $G_1$ minus a set of states $Q$, i.e., $X_{G_2} = X_{G_1} \setminus Q$. Formally, $G_2 \sqsubset G_1$ if (i) $G_2$ is a subgraph of $G_1$, and (ii) for every $x, y \in X_{G_2}$ such that $f_{G_1}(x,s) = y$ for some $s \in \mathcal{L}(G_1)$ then $f_{G_2}(x,s) = y$. As a consequence, $G_2 \sqsubset G_1$ implies $\mathcal{L}(G_2) \subset \mathcal{L}(G_1)$, see, e.g., (Cho; Marcus, 1989; Cassandras; Lafortune, 2021) for more details.

## 2.2 SUPERVISORY CONTROL THEORY

Supervisory Control Theory (SCT) focuses on the synthesis of controllers for DES (Ramadge; Wonham, 1989). In this sense, the behavior of an uncontrolled system (modeled

by automaton $G$) is restricted by an external agent, generally called the supervisor, in order to achieve a given set of specifications.

In SCT, $\Sigma$ is partitioned into two disjoint subsets $\Sigma_c \dot{\cup} \Sigma_{uc}$ where

- $\Sigma_c$ is the set of controllable events: these are the events that can be prevented (or disabled) from happening by the supervisor;
- $\Sigma_{uc}$ is the set of uncontrollable events: these events cannot be directly prevented from happening by the supervisor.

In partially observed DES (where the supervisor does not observe all the events that $G$ executes), the set of events is also partitioned as $\Sigma_o \dot{\cup} \Sigma_{uo}$, where

- $\Sigma_o$ is the set of observable events: these are the events that can be observed by the supervisor;
- $\Sigma_{uo}$ is the set of unobservable events: these are the events that cannot be observed by the supervisor.

The natural projection $P_o : \Sigma^* \to \Sigma_o^*$ is used to map any string executed in the plant $G$ to the sequence of observations associated with it. The natural projection is recursively defined as $P_o(s\sigma) = P_o(s)P_o(\sigma)$, for $s \in \Sigma^*$, and $\sigma \in \Sigma$, according to the following properties:

$$P_o(\varepsilon) = \varepsilon,$$

$$P_o(\sigma) = \begin{cases} \sigma, \text{ if } \sigma \in \Sigma_o, \\ \varepsilon, \text{ if } \sigma \in \Sigma_{uo}. \end{cases}$$

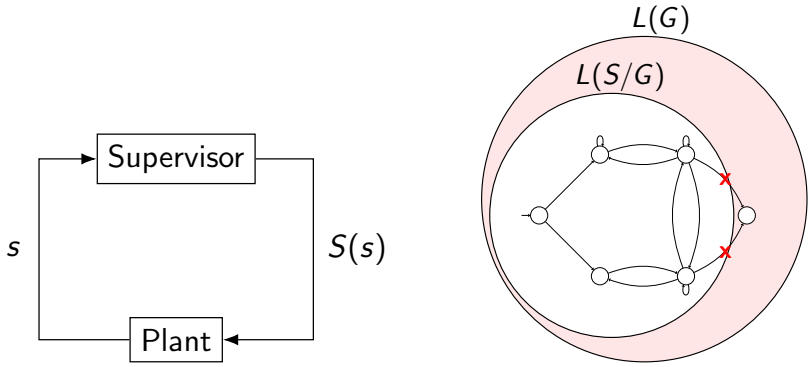
The corresponding inverse projection $P_o^{-1} : \Sigma_o \to 2^{\Sigma^*}$ is defined as

$$P_o^{-1}(t) = \{s \in \Sigma^* : P_o(s) = t\} \ (\forall t \in \Sigma_o^*).$$

Assuming that $G$ represents an uncontrolled plant, a supervisor denoted as $\mathcal{S}$ observes the events generated on the plant and subsequently disables events in $\Sigma_c$ to constrain the system behavior and enforce a predetermined specification. Formally, a supervisor $\mathcal{S}$ is a map from $\mathcal{L}(G)$ to a subset of events to be enabled $\mathcal{S} : \mathcal{L}(G) \to 2^{\Sigma}$. The representation of the supervisor $\mathcal{S}$ can be expressed by an automaton $S = (X_S, \Sigma_S, f_S, \Gamma_S, x_{0,S}, X_{m,S})$, which is usually called a supervisor realization. Thus, the plant and supervisor form a closed-loop system denoted by $S/G$ ($S$ controlling $G$) that is modeled by the automaton $S \| G$. In this context, $\mathcal{L}(S/G)$ is a subset of $\mathcal{L}(G)$ while $\mathcal{L}_m(S/G)$ is a subset of $\mathcal{L}_m(G)$, due to the disablement actions imposed by $S$ on $G$.

For any string $s \in \mathcal{L}(S/G)$, the set $\mathcal{S}(s)$ represents the enabled events that $G$ can execute at its current state $f(x_0, s)$, as depicted in Figure 2a. In this thesis, we use notation

Figure 2 – Supervisory control system.




(a) Feedback loop of supervisory control.

(b) Closed-loop behavior.

Source: Designed by the author (2024).

$\neg\mathcal{S}(s)$ to denote the set of events disabled by the supervisor after $s$. The disabled events at a specific supervisor state $x_S \in X_S$ are represented by $\neg\mathcal{S}(x_s)$. In this manner, the plant $G$ can only execute a controllable event from its current active event set $\Gamma(f(x_0, s))$ if such an event is also contained in $\mathcal{S}(s)$. This ensures the enforcement of the control specification, as illustrated in Figure 2b. Likewise, the plant $G$ cannot execute an event from its current active event set if such an event is also contained in $\neg\mathcal{S}(s)$. It is important to note that $\mathcal{S}(s) \cap \neg\mathcal{S}(s) = \emptyset$.

The supervisor $\mathcal{S}$ is said to be nonblocking if $\mathcal{L}(S/G) = \overline{\mathcal{L}_m(S/G)}$, which means that any string generated by the controlled plant can be expanded to a marked string representing a completed task. A blocking supervisor, on the other hand, results in a controlled system that cannot complete the execution of a given task.

In the framework of SCT (Ramadge; Wonham, 1989), the monolithic approach lies in designing a single supervisor responsible for coordinating the behavior of the entire plant. In composed systems, this involves the composition of all subsystem models $G_i$ (where $i$ is related to the number of subsystems) to form an automaton $G$, which represents the uncontrolled behavior of the plant.

The process of synthesizing a monolithic supervisor also involves the definition of control specifications on the behavior of $G$ that are typically given as natural language statements (e.g., avoid a list of illegal states of $G$, alternate specific events, execute a set of events in a certain priority order).

Consider that a language $K \subseteq \mathcal{L}_m(G)$ represents a control specification over the plant $G$. Formally, $K$ is said to be controllable with respect to $G$ if $\overline{K}\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}$, i.e., if its prefix closure $\overline{K}$ does not change under the occurrence of uncontrollable events in $G$. According to Ramadge and Wonham (1989), a nonblocking supervisor $\mathcal{S}$ such that $\mathcal{L}_m(S/G) = K$ exists if and only if $K$ is controllable with respect to $G$ and $\overline{K} \cap \mathcal{L}_m(G) = K$. Nevertheless, if $K$ is not controllable, the *supremal controllable sublanguage of* $K$ must be

computed, denoted by $SupC(K, L(G))$. Therefore, $\mathcal{L}_m(S/G) = SupC(K, \mathcal{L}(G))$.

## 2.3 CYBERATTACKS AGAINST CPDES

In CPDES, some system components communicate with controllers through vulnerable communication networks that are susceptible to cyberattacks. If an unauthorized person (an intruder) gains access to such a network, they can eavesdrop on system information and manipulate transmitted data by injecting false information or suppressing legitimate messages. These actions can cause the system to behave unexpectedly, resulting in physical or operational damage and violating fundamental security properties such as privacy, integrity, and availability.

The cyberattacks conducted in this scenario are divided into two main categories: passive attacks and active attacks.

### 2.3.1 Passive attacks

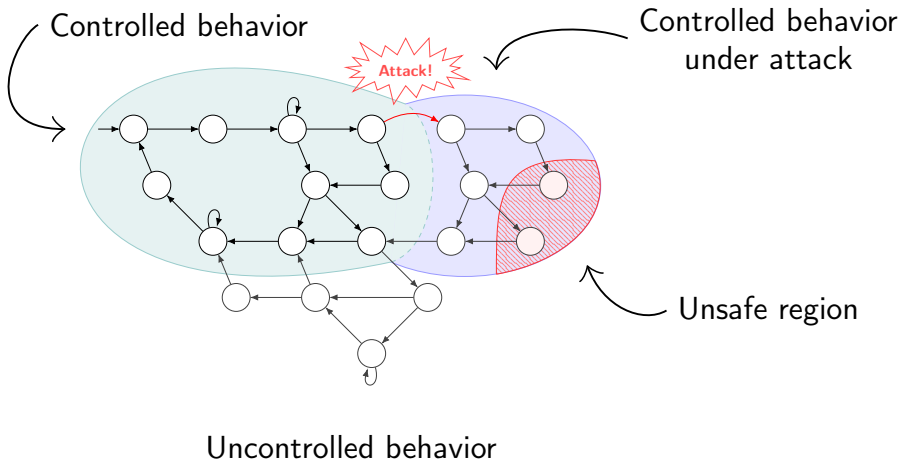
In passive attacks, the intruder does not interact with the system. Instead, it observes the evolution of the system aiming to capture some secret information of interest. This secret information is generally mentioned as a "predicate" or "secret" and may be related to various aspects of the system, such as the initial state or current state of the system or even a certain sequence of events executed in the system. Even though there is no potential damage to the system components due to these attacks, they represent a significant threat to system privacy and confidentiality.

Investigations on passive attacks in the context of DES started with the seminal works of Hadj-Alouane et al. (2005a) and Hadj-Alouane et al. (2005b) on non-interference and of Saboori and Hadjicostis (2007) on opacity. These methods aim to characterize if secret information about the system's behavior can be hidden from intruders. Among them, opacity has attracted significant attention in recent years. Comprehensive surveys on opacity can be found in the literature (Jacob; Lesage; Faure, 2016; Lafortune; Lin; Hadjicostis, 2018; Guo et al., 2020; Balun; Masopust, 2021). Other important approaches to deal with passive attacks in the context of DES can be found in the literature, such as *secret securing with minimum costs problem (SSMCP)* (Matsui; Cai, 2019; Ma; Cai, 2022b), and *encryption methods* (Fritz; Fauser; Zhang, 2019; Zhou et al., 2020; Lima; Carvalho; Moreira, 2020; Lima, 2021; Lima et al., 2022; Lima; Carvalho; Moreira, 2023; Moreira et al., 2024).

### 2.3.2 Active attacks

Active attacks occur when an attacker directly interacts with the targeted system by injecting false information or suppressing legitimate information, with the goal of compromising system integrity or availability. Attacks against the system integrity are

Figure 3 – Controlled system under active attacks.




Source: (Oliveira et al., 2023a).

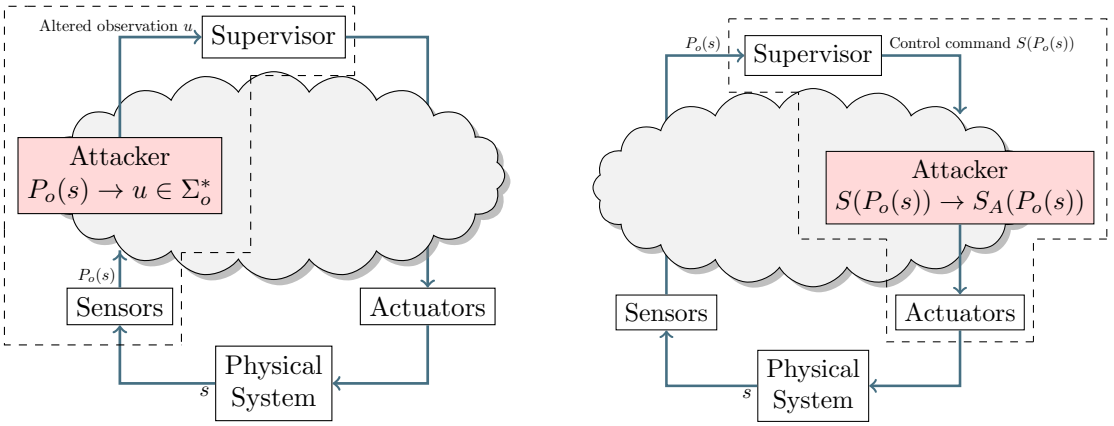
performed by manipulating the data transmitted over the network, aiming to use fake information to deceive system devices as well as the supervisor. Thus, active attackers explore interaction with the system with the goal of driving it outside of the secure, controlled behavior, where undesirable actions are executed. In addition, when the system reaches an unsafe region, it performs actions that compromise its integrity, resulting in damage to its equipment. Figure 3 depicts a controlled system under active attacks, which may drive the system into unsafe regions.

As for the terms used in the literature regarding the characteristics of attacks on the integrity of the system, an attack is called *covert* (Tai; Lin; Su, 2023b) or *stealthy* (Meira-Góes et al., 2017; Fritz; Zhang, 2023) if the intruder remains undetected by the supervisor and causes damage to the system. In this scenario, the attacker assumes a *non-risky* profile. Conversely, an attacker that does not prioritize being covert is called a *risky attacker*. The term *deception attacks*, on the other hand, is commonly used to refer to attacks that are conducted in such a way that the supervisor is fed with false information by the intruder in order to bring the system to a critical state where damage occurs (Meira-Góes; Marchand; Lafortune, 2023). Other general terms used to describe attacks on the integrity of the system are *malicious attacks* and *harmful attacks*.

In the context of DES, active attacks against the integrity of the system can be carried out in three possible manners: (i) attacks in the vulnerable sensor channels, also called "*sensor attacks*", which affects the plant data observed by the supervisor; (ii) attacks in the vulnerable control command channels, named "*actuator attacks*", which affects the data sent to the plant by the supervisor; and (iii) attacks on both sensor and control command channels, known as *"sensor-actuator attacks"*.

Regarding sensor attacks, it is important to highlight that the observations reported by sensors in a DES are modeled as strings of events represented by a sequence of symbols.

Figure 4 – Cyberattacks in DES.




(a) Sensor attack (b) Actuator attack

Source: Adapted from Basilio, Hadjicostis and Su (2021).

Once a string $s$ subject to a natural projection $P_o$ has been captured by an attacker, it can be manipulated using one of the following operations: symbol erasure, symbol insertion, and symbol modification, which can be considered a combination of the first two operations. That is, $P_o(s) \rightarrow u \in \Sigma_o^*$, where $u$ denotes the altered observation. Then, the manipulated string will serve as the input to the supervisor and will possibly drive the system to execute actions outside of the secure controlled behavior (Su, 2017; Meira-Góes et al., 2020). A general architecture of a controlled DES under a sensor attack is depicted in Figure 4a.

It is worth noting that sensor attacks in which the attacker erases sensor observations can be considered as attacks on system availability. In this case, the supervisor cannot issue new control commands because it is unable to observe the events executed in the plant. In this scenario, the system availability is violated. However, the works that address the problem of sensor attacks in DES generally assume that the intruder is not only interested in deleting sensor observations, but also in inserting fake events to deceive the supervisor. Thus, the intruder expects that the control actions issued by the supervisor, enabled by the altered observations, will result in a behavior that implies a violation of system integrity. This type of attack is commonly referred to in the literature as *sensor deception attacks* (Meira-Góes et al., 2020; Meira-Góes; Marchand; Lafortune, 2023; Labed et al., 2023).

Actuator attacks, in turn, are performed in the vulnerable control command channels aiming to intercept and alter control commands that a supervisor issues based on the observed events $S(P_o(s))$. These commands are characterized by enablement and disablement actions (Carvalho et al., 2016; Li; Tong; Giua, 2020; Ma; Cai, 2022a). In this case, the attacker may alter control commands sent to a given actuator $S(P_o(s)) \rightarrow S_A(P_o(s))$ and enable it when it should be disabled, or the opposite, as illustrated in Figure 4b.

When an attacker explores both the vulnerable sensor and actuator communication channels, a sensor-actuator attack is performed, which is a combination of the attacks

shown in Figures 4a and 4b (Carvalho et al., 2018; Lin; Su, 2021; Wang; Tong, 2022). In this scenario, the attacker is able to eavesdrop a subset of events in the sensor channels, as well as in the actuator channels, and manipulate the captured data in an attempt to drive the system to an unsafe region and to inflict damage in the system components.

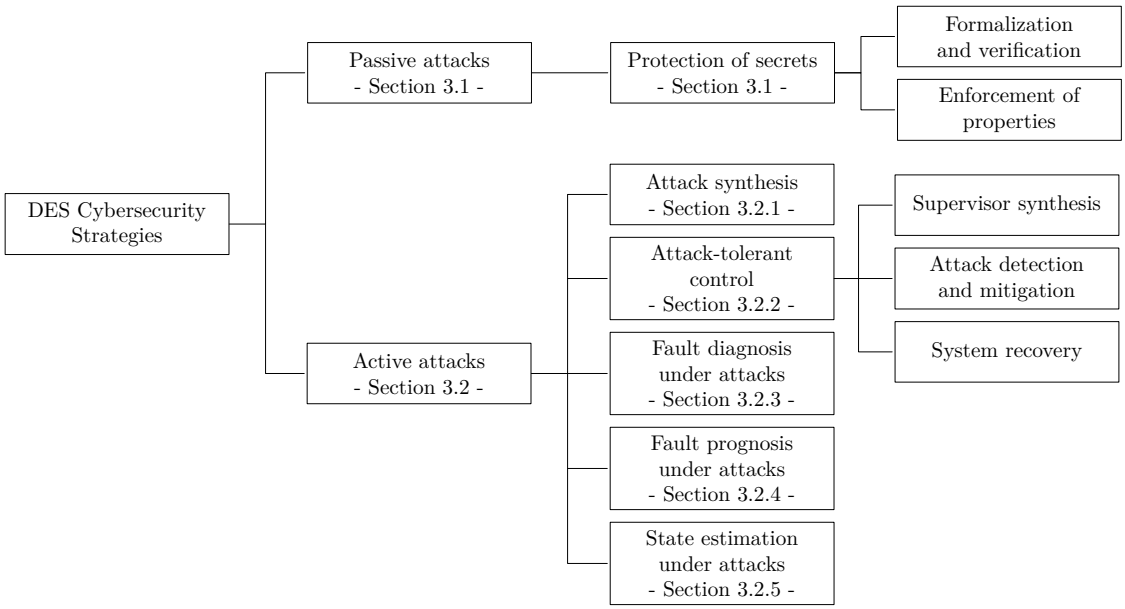
Basilio, Hadjicostis and Su (2021) presented a comprehensive overview of papers investigating attack and defense mechanisms in the context of DES, considering sensor, actuator, and sensor-actuator attacks before 2021.

In the literature, some papers, such as Lima et al. (2017), and Lima et al. (2022), mention a specific term called “Man-in-the-middle” referring to a type of network attack where the attacker is able to eavesdrop on communication channels as well as creating, hiding, and changing information transmitted over the network without being detected. In the topic of DES, this type of attack is considered a similar case of sensor-actuator attack.

# 3 LITERATURE REVIEW

In this chapter, we provide a classification and analysis of the cybersecurity strategies in the context of DES that were identified in the literature, considering the main goal set by each strategy (e.g., maintaining privacy, integrity, or availability of a system), the types of attacks considered, and also the formalism used to model a DES. In Figure 5, an overview of the cybersecurity strategies in the context of DES is provided.

Figure 5 – Classification of cybersecurity strategies.




Source: (Oliveira et al., 2023a)

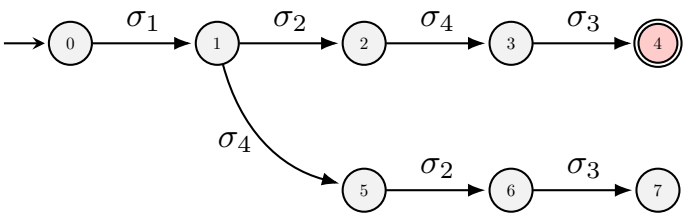
## 3.1 PROTECTION OF SECRETS AS A CYBERSECURITY STRATEGY AGAINST PASSIVE ATTACKS

Numerous methods for protecting secrets in the context of DES have been proposed in the literature. These methods mostly deal with passive attacks (eavesdropping attacks) to protect the privacy of the system. The secrets of the system can be modeled both as a subset of states (the states are divided into secret and non-secret states) and as a sequence of events of interest. Thus, the main goal of a passive intruder is to determine whether the system has reached a secret state or a secret sequence has been executed.

**Example 1.** *Consider the automaton $G$ presented in Figure 6, with the sets of secret and non-secret states $X_S = \{4\}$ and $X_{NS} = X \backslash X_S$, respectively, and the secret string $s = \sigma_1\sigma_2\sigma_4\sigma_3$.*

Let us first consider the set of observable events $\Sigma_o = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$. When $s$ is observed by intruders, they are certain that the secret sequence has occurred and the system has reached state 4, which is a secret state. Consequently, the secret is revealed.

Figure 6 – Automaton $G$ for Example 1.




Source: (Oliveira et al., 2023a)

Now, consider that $\Sigma_o = \{\sigma_1, \sigma_2, \sigma_3\}$ and the set of unobservable events $\Sigma_{uo} = \{\sigma_4\}$. When $s$ is executed in the system, the intruder cannot accurately determine whether the secret sequence has occurred or the system has reached a secret state, once another sequence $s' = \sigma_1\sigma_4\sigma_2\sigma_3$ with the same projection exists. That is, $P_o(s) = P_o(s') = \sigma_1\sigma_2\sigma_3$.

In this sense, papers that focus on passive attacks follow one of the following directions: (i) formalization and verification of properties and (ii) enforcement of properties. In the first direction, properties aimed at protecting secrets are formalized, including the characteristics of the secret itself (Saboori; Hadjicostis, 2009; Yang; Deng, 2025). In this sense, it is possible to verify whether the system is able to protect the secret or not. In the second direction, the focus is on systems that lack such inherent protection. In these cases, protection is ensured through enforcement strategies, which can involve the synthesis of supervisory control (Tong; Cai; Giua, 2018) or the implementation of external methods coupled to the system (Wu; Lafortune, 2016).

The following approaches aimed at protecting secrets in the context of DES were identified in the literature: *opacity* (Saboori; Hadjicostis, 2007; Dubreil; Darondeau; Marchand, 2008; Lin, 2011), *non-interference* (Basile; Tommasi; Sterle, 2021; Basile; Tommasi, 2022), *secret securing with minimum costs problem (SSMCP)* (Matsui; Cai, 2019; Ma; Cai, 2022b), and *encryption* (Fritz; Fauser; Zhang, 2019; Zhou et al., 2020; Lima; Carvalho; Moreira, 2020; Lima, 2021; Lima et al., 2022; Lima; Carvalho; Moreira, 2023; Moreira et al., 2024). For further details on the above approaches, the reader is referred to a comprehensive literature review provided by Oliveira et al. (2023a).

Beyond protection of secrets, a complementary line of work has investigated the trade-off between privacy and utility in DES subject to passive attacks. In this context, utility refers to the ability of legitimate observers to obtain sufficient information from system observations to support decision making or achieve operational goals, while still preventing sensitive information from being disclosed to malicious intruders. Unlike opacity and non-interference, which focus exclusively on restricting information available to intruders, utility-aware approaches explicitly regulate the information flow so as to balance privacy and usability. This perspective has motivated the development of methods that jointly enforce privacy and utility (Wu et al., 2017; Wintenberg et al., 2022; Wintenberg; Ozay; Lafortune, 2025; Cardoso; Moreira; Carvalho, 2023; Barcelos; Basilio, 2023; Mayer

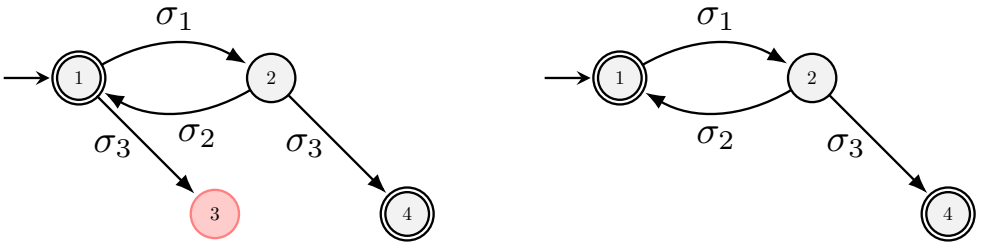
et al., 2024; Reis; Carvalho; Moreira, 2026).

## 3.2 CYBERSECURITY STRATEGIES ADDRESSING ACTIVE ATTACKS

The strategies related to active attacks found in the literature deal mainly with preserving the integrity of the targeted systems. In general, it is considered that the system has one or more unsafe states, and the supervisor disables some controllable events in order to avoid entering such states.

**Example 2.** *Consider the plant $G$ shown in Figure 7a, where $\Sigma_{uc} = \{\sigma_2\}, \Sigma_c = \{\sigma_1, \sigma_3\}$. In a normal operation, the supervisor (Fig. 7b) disables the controllable event $\sigma_3$ from state 1 in order to prevent the system from reaching state 3, which is an unsafe state.*

Figure 7 – Plant and Supervisor for Example 2.




(a) Plant G with an unsafe state (b) Supervisor

Source: (Oliveira et al., 2023a)

However, in an active attack scenario, an attacker could overwrite the supervisor's disablement of $\sigma_3$ with an enablement action. As a result, the system may be driven to an unsafe state, while synchronism between the plant and the supervisor is lost. Moreover, in state 2, the attacker could corrupt sensor observations by hiding the occurrence of $\sigma_2$ from the supervisor. After $\sigma_2$ occurs, the plant transitions to state 1, whereas the supervisor remains in state 2. Consequently, $\sigma_3$ is enabled by the supervisor, driving the system to the unsafe state.

In the literature, papers dealing with active attacks rely on the following strategies: (i) *attack synthesis*, which provides formulations for successful active attacks from the attacker's point of view; (ii) *attack-tolerant control*, which presents methods to detect the occurrence of active attacks as well as methods to ensure the achievement of the system specification in such situations; (iii) *fault diagnosis under attacks*, which provides methods to ensure a system's ability to diagnose faults despite the occurrence of active attacks; (iv) *fault prognosis under attacks*, which provides methods to predict future occurrences of faults in a system under attacks; and (v) *state estimation under attacks*, which explore the ability to perform state estimation in the presence of malicious attackers.

### 3.2.1 Attack Synthesis

An important cybersecurity strategy that has emerged in the literature is to provide active attack synthesis, i.e., formulations for successful active attacks from the attacker's perspective. Su (2017) points out that an intelligent, active attacker must have some familiarity with the targeted system, as well as skills to disclose information of interest and disrupt resources in order to carry out a successful active attack. In this sense, papers based on this strategy generally assume that the attacker is fully aware of the system model and is able to perform intelligent actions to inflict damage on the system's components. Although these assumptions may seem too strong, this approach is helpful in effectively supporting the construction of methods to protect systems against active attacks, given the proposed attack synthesis itself.

The problem of synthesizing a successful stealthy deception attack was investigated by Meira-Góes et al. (2017) and Meira-Góes et al. (2020). In these papers, the system is modeled by finite-state automata, and the attacker is assumed to be aware of both the plant and the supervisor models. The authors also assume that only the sensor channels are vulnerable to attacks. A game-like discrete transition structure is used to capture the interaction between two players (system and attacker). Based on this, attacks are formulated in such a way that the supervisor is induced to allow the system to reach unsafe states. This approach was also considered for stochastic DES modeled as probabilistic automata in Meira-Góes, Kwong and Lafortune (2019), Meira-Góes, Kwong and Lafortune (2022). Besides the topic of automata-modeled DES, the problem of synthesizing successful sensor attacks was also considered in the topic of Petri nets (Zhang et al., 2020; Habbachi et al., 2022).

Lin et al. (2019) and Lin, Zhu and Su (2020) considered the actuator attack synthesis problem in automata-modeled DES. The authors assumed that the attacker knows the models of the system (plant and supervisor) and is able to modify each control command issued by the supervisor, considering a specified subset of vulnerable controllable events. For the attack to succeed, the system under attack must eventually reach an unsafe state. The problem of synthesizing a successful actuator attack is solved by a reduction of the well-known Ramadge-Wonham supervisor synthesis.

Some of the existing papers in the literature consider the problem of synthesizing both sensor and actuator successful attacks. Khoumsi (2019) proposed a method for detecting sensor-actuator attacks in DES modeled by automata. Afterwards, the author proceeds to consider the attacker's point of view by synthesizing a successful sensor-actuator attack which is able to remain undetectable to the proposed attack detection module. On the other hand, some papers rely upon different assumptions by considering that the attacker knows the model of the plant, but the model of the supervisor is unknown (Lin et al., 2022; Tai et al., 2023; Ma; Giua; Seatzu, 2025a; Ma; Giua; Seatzu, 2025b). In

this sense, methods are provided to solve the sensor-actuator attack synthesis problem based on the plant model and a finite set of observations.

Recent works have further extended attack synthesis to scenarios involving distributed architectures and attackers operating under explicit resource constraints (Tai et al., 2024; He et al., 2025).

Once a successful attack synthesis is designed, it is possible to make use of such synthesis to formulate an attack-aware robust supervisor in order to achieve an attack-tolerant control in the system.

### 3.2.2 Attack-tolerant control

Due to the threat of active attacks in DES, a relevant number of papers have explored methods for attack-tolerant control as a cybersecurity strategy. According to Yao, Yin and Li (2020), an attack-tolerant control system must be able to strictly prevent the system from entering unsafe states while maximizing the desired behavior, i.e., closed-loop behavior without attacks. In particular, this strategy is divided into three main strands: (i) *supervisor synthesis*; (ii) *attack detection and mitigation*, and (iii) *system recovery*.

#### *3.2.2.1 Supervisor synthesis*

Considering that an active attacker has the ability to conduct intelligent and malicious actions based on a specific attack formulation, namely, an active attack synthesis, some papers in the literature have presented methods for synthesizing a supervisor that strives to be robust and annul the effects of such an attack. These papers usually describe details of the active attack synthesis under consideration.

Su (2018) presented a sensor attack synthesis called *attack-with-bounded-sensor-reading-alterations* (ABSRA). When using ABSRA, the attacker intercepts the observations reported by the sensors and modifies them arbitrarily, with an upper bound on the length of each modified observation sequence. The attacker's goal is to deceive the supervisor while using the supervisor's control commands to cause damage. Upon the attack synthesis, the author presented a supervisor synthesis method in order to obtain a robust supervisor against the ABSRA. On the other hand, Su (2022) investigates the decision problem regarding the existence of resilient, non-blocking supervisors against all possible smart sensor attacks.

Meira-Góes, Lafortune and Marchand (2021) deals with supervisor synthesis for sensor attacks as well. In this work, the problem of synthesizing a supervisor that is robust against this class of sensor deception attacks is formulated. The proposed approach combines game-theoretic techniques on automata with imperfect information and results from the supervisory control of partially observed DES, leading to necessary and sufficient conditions for the existence of such a robust supervisor.

The synthesis of a robust supervisor against indefinite actuator attacks was considered in the work of Ma and Cai (2022a). In this case, the authors assume that there is no a priori knowledge about which events are susceptible to attackers or not. Thus, the events targeted by attackers are indefinite, and no attack synthesis is considered. The proposed method considers that each unsafe state in the plant is associated with a required safety level. Consequently, the supervisor must avoid entering in a given unsafe state if the number of attacks is no greater than the safety level of such a state. This is achieved by restricting system behavior and disabling actions if they would lead the system into unsafe states after an actuator attack. This approach, however, is unsuitable for scenarios where the system is required to execute its complete behavior.

Recent works on supervisor synthesis against actuator attacks have been reported in the literature (Xie; Tong, 2025; Cui; Giua; Yin, 2025). Notably, Xie and Tong (2025) explicitly considers actuator disablement (AD) attacks in addition to actuator enablement (AE) attacks. This is significant because most papers on actuator attacks in the context of DES focus strictly on AE-attacks under the assumption that AD-attacks only reduce the controlled behavior and therefore cannot drive the system to execute a critical behavior. Nevertheless, AD-attacks may induce blocking, which compromises availability and is a critical cybersecurity concern.

In this context, Zhu, Lin and Su (2019) proposed a supervisor synthesis scheme based on the covert actuator attack formulation of Lin et al. (2019). The main goal is to synthesize an obfuscated supervisor that is resilient to covert actuator attacks while preserving the original closed-loop behavior. The proposed approach relies on enumerating behavior-preserving supervisors and verifying their non-attackability using a Boolean satisfiability–based procedure. However, the enumeration of all candidate supervisors may lead to severe computational complexity, which may cause massive degradation in algorithmic performance, as acknowledged by the authors. In this sense, ensuring protection against covert actuator attacks while maintaining the original closed-loop behavior remains an ongoing challenge in the literature, warranting further investigation in this direction.

More recently, a sequence of works has investigated the decidability of existence of resilient or fortified supervisors against covert actuator attacks. In particular, Tai, Lin and Su (2023a) and Tai, Lin and Su (2024a) formalized the supervisor fortification problem and established procedures to decide whether a behavior-preserving resilient supervisor exists. This line of research was further extended in the work of Tai, Lin and Su (2024b), which addressed the existence of resilient supervisors against covert sensor–actuator attackers using a Resilient Ensemble Bipartite Supervisor representation. These works characterize when resilient, behavior-preserving supervisors exist and when such fortification is not achievable.

Synthesis of resilient supervisors against sensor-actuator attacks was also investigated using Petri nets (Wang et al., 2021), finite-state transducers (FST) (Wang; Pajic,

2019a; Wang; Pajic, 2019b; Wang et al., 2023), and automata (Lin; Zhu; Su, 2019; Zheng; Shu; Lin, 2021; Lima et al., 2022; Meira-Góes; Marchand; Lafortune, 2023; Zheng; Shu; Lin, 2024).

#### *3.2.2.2 Attack detection and mitigation.*

In this strand of attack-tolerant control, the main idea is to provide methods aimed at recognizing the occurrence of active attacks and mitigate their effects. This strategy has some similarities to the problem of active fault-tolerant supervisory control, where the system must safely continue its operation after faults occur and must additionally avoid entering unsafe states. Such capability is represented by a property called *safe controllability* (Paoli; Sartini; Lafortune, 2011).

Attack detection and mitigation were first studied by Thorsley and Teneketzis (2006). In this work, the authors considered the threat of actuator attacks and proposed a method for detecting such attacks in time to prevent the execution of unsafe behavior. In the work of Carvalho et al. (2018), both sensor and actuator attacks were considered, and the identification of such attacks is performed by a detection module coupled to the supervisor. Furthermore, a property termed *General Form of safe controllability* (*GF-safe controllability* for short) was introduced. This property characterizes the ability to detect any attack occurrence and then disable a controllable event before the plant reaches an unsafe state. Variations of *safe-controllability* property were explored by Carvalho et al. (2016), Lima et al. (2017), Lima, Carvalho and Moreira (2018), Lima et al. (2019), Li, Tong and Giua (2020), Yao, Yin and Li (2020), Yu et al. (2023) and Yu, Jia and Cong (2026).

Another approach that aligns with the problem of fault-tolerant control is the one proposed by Tong, Cai and Seatzu (2024). In the system considered by the authors, there is a nominal supervisor which achieves a specification without considering attacks, and a set of security supervisors that can be switched based on attack detection. The switching operations are issued by a decision module to ensure system integrity.

A different approach to detect the presence of sensor attackers was proposed by Tong, Wang and Giua (2022), in which the detection mechanism only needs to keep track of the last observable event received. If such an event is not enabled at the current state of the supervisor, the attack is detected, and all controllable events are disabled thereafter. In addition, the authors proposed a method to verify if there exists a possible formulation for a sensor attack (considering attacks on insertion and erasure in sensor observations) for a given plant and supervisor such that the supervisor fails to prevent the system from entering unsafe states under the sensor attack. This is referred to as *attacker existence problem*, which is solved with polynomial complexity.

A complementary line of work addresses active attacks by examining what an attacker can infer from partial observation before launching an attack. In particular,

He, Wu and Li (2024) introduces strong and weak actuator-enablement estimability (AE-estimability) to characterize whether an attacker can determine that an attacked closed-loop system can reach an undesirable state. To prevent such attacks without restricting the original closed-loop behavior, the authors propose a prevention module based on reverse sensor functions that modify sensor readings to mislead the attacker's estimation.

Some papers have also explored the detection and mitigation of DoS attacks that compromise the availability of DES (Delaval et al., 2020; Sakata; Fujita; Sawada, 2021). In these papers, it is assumed that DoS attacks can be detected by monitoring the delay between the occurrence of observable events. When the measured delay exceeds a predefined time threshold, an interrupt signal is raised, allowing the system to switch from a nominal control to a fallback control. Thus, the system is able to maintain its functionality, be it in a degraded mode. He, Ma and Tang (2021), on the other hand, considered an active attacker with the ability to increase the delay of specific events in the system. In this work, the system is modeled by Petri nets, and the firing frequency of each transition between the states of the system is assumed to be identical under normal conditions. The authors introduce the *Performance Safety Enforcing Problem* (PSEP), whose solution is based on finding a protection policy that ensures that the firing frequency of each transition of the system is not lower than a certain threshold when this type of attack occurs.

More recently, artificial neural networks have been employed to detect cyber-attacks in timed probabilistic CPDES (Amri et al., 2025). Detection is performed either over observed event sequences or over time, while attack mitigation is not addressed.

#### *3.2.2.3 System recovery.*

Unlike attack detection and mitigation strategies, which focus on preventing unsafe behavior after attack detection, system recovery approaches consider scenarios in which an attack has already caused disruption or desynchronization. In this case, the objective is to restore nominal or otherwise acceptable system operation. As such, recovery mechanisms are closely related to availability and liveness objectives rather than integrity preservation alone. Consequently, system recoverability becomes a key requirement for achieving resilience in CPDES.

In the work of Alves and Pena (2022), a recovery mechanism for active attack scenarios is proposed based on the theory of *synchronizing automata*. The main idea is to augment the models of the plant and the specifications with recovery events, so that a synchronizing word exists. This word can then be executed to drive both the plant and the supervisor back to their initial states, thereby restoring synchronism and recovering the closed-loop system. While effective for correcting desynchronization, this approach does not modify the system structure or control logic, and thus the original vulnerabilities to future attacks remain unchanged.

In the recovery framework proposed by Cavalcanti et al. (2025), recovery is for-

mulated through a recoverability property defined over the supervisor's state estimate after attack detection and isolation. Based on this property, a nonblocking supervisor is synthesized to drive the system from an attacked state estimate back to its nominal closed-loop behavior in a finite number of observations while avoiding unsafe states. The recovery process operates under partial observation and assumes that the attacker has been isolated prior to recovery, which eliminates the possibility of further attacks during the recovery phase.

A different recovery paradigm is investigated by Sakata et al. (2023), Sakata et al. (2024) through the design of fallback control systems. These works consider industrial control systems in which a backup controller takes over operation after attack detection. The fallback logic is synthesized using supervisory control theory, integrating normal and fallback control modes. While this strategy enables continued operation after attacks, it relies on the availability of a fallback controller that is assumed to remain secure even though connected to the network.

### 3.2.3 Fault diagnosis under attacks

Fault diagnosis is related to the ability of a system to identify the occurrence of faults in its components. In fact, a fault is represented by an unobservable event, whose identification must occur with certainty in a bounded number of observable events after its occurrence. Li, Hadjicostis and Wu (2022) consider the problem of fault diagnosis when the observed sensor data can be tampered by a malicious attacker. The authors assume that each operation executed by the attacker (symbol insertion, deletion, or modification) bears a favorable cost. Then, a method for verifying tamper-tolerant fault diagnosis under cost constraints in a centralized observation setting is presented.

Stealthy sensor attacks in the context of fault diagnosis have also been investigated. In Kang et al. (2023), the authors introduce a *stealthy joint diagnoser* to characterize all possible stealthy attacks that can prevent fault detection under corrupted observations. This structure allows one to determine whether a stealthy harmful attack exists and to perform fault diagnosis despite the presence of such attacks.

More recent works have addressed diagnosability and detectability of DES under sensor attacks. In Lin, Lafortune and Wang (2023), a formal sensor attack model is introduced, and diagnosability conditions are derived under adversarially corrupted observations. Building on this framework, Lin, Lafortune and Wang (2024) jointly address fault diagnosability and attack detectability, characterizing when faults and sensor attacks can be distinguished based solely on compromised sensor information.

Decentralized fault diagnosis under attacks has also been studied in the literature. In this setting, multiple modules are used for diagnosing faults. These modules, also called *local diagnosers*, can observe the events of specific areas of the plant through communication channels that may be vulnerable to active attacks. In this sense, the presence of an attacker

in a system may involve not only problems in supervisory control but also in fault diagnosis activities. Methods for ensuring decentralized fault diagnosis despite the actions of active attackers are proposed by Li, Hadjicostis and Wu (2021), Alves et al. (2022).

Recent papers have also considered fault diagnosis in DES under attacks, indicating that investigation in this line of work remains an open and active research direction (Hu et al., 2025; Hu; Li; Liu, 2025; Shamloo; Santis; Benedetto, 2025).

### 3.2.4 Fault prognosis under attacks

Fault prognosis in the context of DES refers to the ability to infer with certainty future occurrences of faults or failures in these systems by observing their current and past behavior. As pointed out by Watanabe et al. (2021), the terms prognosability and predictability and their derivatives are equivalently used in the literature on fault prognosis of DES.

Zhang (2023) investigated the problem of robust predictability against sensor attacks using diagnosers. The author assumes that a malicious attacker is able to corrupt sensor observations by inserting fake events or deleting real events, so that the diagnoser cannot predict the occurrence of future critical events. To address this problem, a structure called a *hybrid diagnoser* is proposed. Such a structure allows the diagnoser to predict the occurrence of critical events, even if an attacker may tamper with its observation. An approach to test robust predictability under sensor attacks is also provided.

Building on this line of work, Kang, Liu and Li (2026) introduces the notion of robust prognosability against stealthy sensor attacks, which captures the ability to predict system faults prior to their occurrence despite corrupted observations. In contrast to Zhang (2023), which relies on a hybrid diagnoser construction with exponential complexity, the authors propose a deterministic verifier under attack that yields necessary and sufficient conditions for robust prognosability verification. The proposed approach enables polynomial-time verification with respect to the number of states and events of the system.

### 3.2.5 State Estimation under attacks

The problem of state estimation is related to the setting of partially-observed DES, where an operator observes the events of a plant through a natural projection in an attempt to estimate the current state of the system. Considering that the communication channels may be vulnerable, an attacker can perform active attacks on sensor observations in order to alter the state estimation of the operator.

Based on this, Zhang et al. (2021) proposed a method based on an automaton called *joint estimator*, which contains all possible attacks that can be performed on the system. The aim of the proposed solution is to find the joint state estimation for each possible corrupted observation, as well as to determine whether the attack is harmful or not. Zhang

et al. (2022) extended this work and showed how to trim the *joint estimator* aiming to obtain the *supremal stealthy joint subestimator* that contains all the stealthy attacks. Chen, Su and Li (2022), on the other hand, investigate a property called *detectability*, which is used to evaluate whether the observed events are sufficient to determine the current state for a given system considering the presence of sensor attackers.

Other work along this line is Zheng, Shu and Lin (2024), which develops state estimation under joint sensor-actuator attacks and uses the resulting state estimates to synthesize a supervisor that enforces safety specifications.

Hadjicostis et al. (2022) analyzed, in their tutorial paper, the problem of active attacks in the context of DES considering the theories of supervisory control (synthesis of robust supervisors against covert attacks), diagnosability, and state estimation under attacks. The authors model the attacker by using input/output automata combined with augmented models of the plant and supervisor in order to achieve attack-tolerant diagnosability, estimation, and resilient-supervisor synthesis.

State estimation under attacks has also been studied through the notion of detectability. In Chen, Su and Li (2025), the authors investigate whether strong and weak detectability of partially observed DES can be compromised by covert sensor attacks. The notions of attackable strong detectability and attackable weak detectability are introduced, and necessary and sufficient conditions are derived using observer constructions. This work shows that sensor attacks may fundamentally impair the ability to estimate the system state, even when detectability holds in the attack-free case.

Along this line, Ritsuka et al. (2025) further investigates detectability under sensor attacks by shifting the focus from fault diagnosis to state estimation, and derives conditions under which system states remain detectable despite adversarially corrupted observations.

## 3.3 DISCUSSION

Over the years, a considerable number of papers on the cybersecurity of DES have been published in the literature. Passive attacks (eavesdropping) represent the security threat that researchers have addressed most intensively since 2005. It started with the seminal work of Hadj-Alouane et al. (2005a), Hadj-Alouane et al. (2005b) on non-interference and of Saboori and Hadjicostis (2007) on opacity, which became more numerous in the following years. On the other hand, although the first study addressing active attacks in the context of DES was published in 2006 with the work of Thorsley and Teneketzis (2006) on actuator attacks, active attacks have gained more attention only from 2016 with several papers, including: Carvalho et al. (2016), Meira-Góes et al. (2017), Lima et al. (2017), Su (2018), Carvalho et al. (2018).

Another important fact is the increasing number of contributions to active attacks over the years. In 2022, for the first time, it exceeded the number of contributions to

passive attacks, as pointed out by Oliveira et al. (2023a).

It is worth mentioning that some of the cybersecurity techniques based on supervisory control apply some level of restriction in the controlled behavior of the system by disabling the part that allows the security violation. It is the case of opacity enforcement by supervisory control (Falcone; Marchand, 2014; Tong et al., 2018) and also techniques aimed at dealing with active attacks based on the strategies of attack detection and supervisor synthesis (Yao; Yin; Li, 2020; Ma; Cai, 2022a). While the approach of constraining the behavior of the system preserves its integrity and confidentiality, it does not apply to situations where the system must execute its full behavior.

Consequently, this issue has received attention in the literature through alternative methods that modify the information exchanged between the plant and the supervisor. In the context of passive attacks, such modifications aim to prevent the attacker from inferring system secrets (Wu; Lafortune, 2014; Ji; Yin; Lafortune, 2019; Wintenberg et al., 2022; Wintenberg; Ozay; Lafortune, 2025). In the context of active attacks, similar information-modification strategies are employed to prevent attackers from carrying out attacks without restricting the nominal closed-loop behavior (Zhu; Lin; Su, 2019; Tai; Lin; Su, 2023a; Tai; Lin; Su, 2024a; Tai; Lin; Su, 2024b; Oliveira et al., 2023b; Oliveira et al., 2024a). In this sense, further research is needed on attack-tolerant control methods that allow the system to perform its complete behavior despite the presence of malicious attackers.

Beyond that, we also highlight that exploring recent advances around the property of safe controllability in terms of fault diagnosis (Leitão et al., 2020; El-Mahdy; Maged; Awad, 2022) and fault prognosis (Watanabe et al., 2021; Watanabe et al., 2022) can help to enhance the resilience of systems to cyberattacks.

Some of the cybersecurity strategies in the context of DES have recently appeared in the literature. They may represent opportunities for future research, which is the case of *system recovery* (Cavalcanti et al., 2025; Oliveira et al., 2025). Other DES cybersecurity strategies that have recently appeared in the literature are based on the development of methods to ensure fault diagnosis, fault prognosis, and state estimation under attacks (Shamloo; Santis; Benedetto, 2025; Ritsuka et al., 2025; Kang; Liu; Li, 2026). As they emerged recently, only a few works are found in this area, which demonstrates the need for further investigations.

Two recent papers investigated the possibility of performing active attacks with the intent to reveal a secret from a DES (Yao; Yin; Li, 2022; Habbachi et al., 2022). In this context, active attacks are also considered a potential threat to system privacy. That is, beyond eavesdropping the vulnerable communication channels, the attacker interacts with the system by inserting false information as well as hiding the actual observations of the system in an attempt to make the behavior of the system under attack lead to the disclosure of a certain secret. This also represents a possible direction for future research

since only passive attacks are generally considered by articles aimed at protecting secrets in the context of DES.

Another relevant issue on the topic of cybersecurity of DES is addressing the reality gap. To formulate cybersecurity methods, it is common for researchers to make assumptions about the attackers' knowledge about the system structure and their capability for observing the system. Nevertheless, these assumptions may not always be applicable in real scenarios. For example, most works that address the problem of active attacks in DES usually assume that the attacker is fully aware of both the plant and supervisor models. This assumption can be overly restrictive and may not align with the complexities of real-world situations. To overcome this limitation, Lin et al. (2022) and Tai et al. (2023) proposed methods for attack synthesis, considering that the attacker is aware of the model of the plant, but the model of the supervisor is unknown. Further research towards this direction could support bridging the reality gap. Furthermore, some papers consider that only the communication channels from the plant to the controller (sensor channels) are vulnerable to attackers, while the communication channels from the controller to the plant (actuator channels) are secure, and vice versa. Depending on the network topology used, there may not exist a practical division between the communication channels. Other papers adopt an assumption wherein certain events are not attackable, while a subset of events remains susceptible and potentially open to infinite attacks. It is important to note, however, that this assumption might not always hold true in real-world scenarios. In this sense, Ma and Cai (2022a) address the problem of indefinite actuator attacks, in which any event can be attacked, relaxing the assumptions usually made. Continued research in this direction also holds the potential to contribute significantly to closing the reality gap. It is worth mentioning that a testbed for cybersecurity techniques in the context of DES is provided by Alves, Rudie and Pena (2022). Although the authors do not present any cybersecurity strategy, the aim of their work is to present an implementation scheme using low-cost microcontrolled devices to be used as a testbed for new techniques based on attack-tolerant control and attack synthesis, thus reducing the reality gap.

# 4 PROTECTING CPDES FROM COVERT SENSOR AND ACTUATOR ATTACKS VIA ACTUATOR EDIT FUNCTIONS

In this chapter, we present the main protection mechanism for integrity preservation developed in this doctoral thesis. We adopt a unified perspective that addresses both sensor and actuator attacks. This unified approach is motivated by the observation that, in practical CPDES, malicious manipulation of sensor information can indirectly cause the execution of unsafe controllable events, just as direct actuator enablement attacks can. By intervening at the actuator interface, the proposed framework prevents the physical execution of unsafe controllable events, whether they are triggered by sensor deception attacks or actuator enablement attacks.

Furthermore, the protection mechanism introduced in this chapter goes beyond integrity preservation and explicitly enforces system liveness. While integrity-focused approaches aim to prevent the system from reaching unsafe states, they may do so by eliminating all the feasible continuations of the system behavior, potentially halting system operation. Such outcomes are particularly problematic in safety-critical applications, where maintaining operation, possibly in a degraded but controlled manner, is essential. Moreover, the ability to preserve liveness under attack is a fundamental prerequisite for any subsequent recovery strategy, as recovery inherently relies on the existence of feasible system continuity (Alves; Pena, 2022; Cavalcanti et al., 2025; Oliveira et al., 2025).

These considerations motivate the development of a protection framework that jointly addresses integrity and liveness in the presence of both sensor and actuator attacks. The remainder of this chapter introduces such a framework by formalizing actuator edit functions, which modify control commands at the plant level to prevent damage while preserving the ability of the system to continue operating under attack.

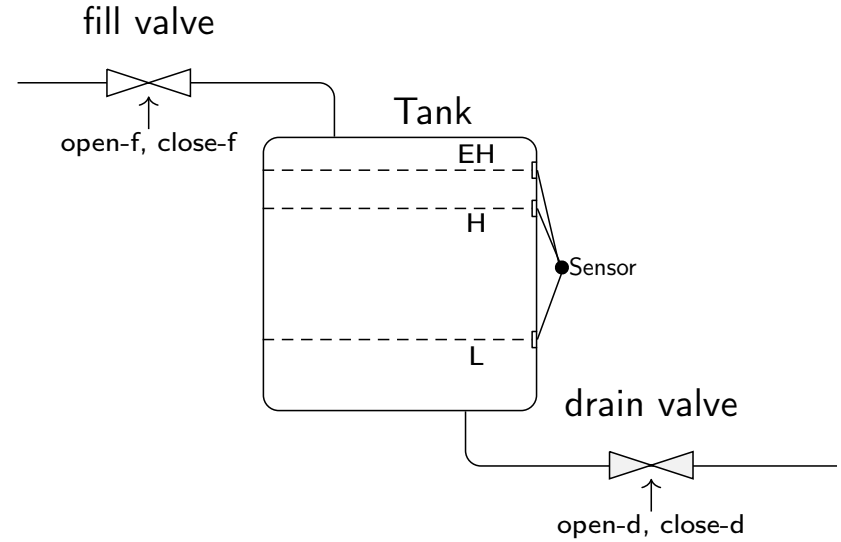
## 4.1 SYSTEM UNDER ATTACK

We assume that the supervisory control system is fully observable. Additionally, we assume the system is subject to both sensor and actuator attacks. In sensor attacks, an attacker hijacks a subset of the sensor observations and alters them to deceive the supervisor. This alteration may occur either through *sensor insertion* (SI), where the attacker inserts observations of events that did not occur in the plant, or through *sensor deletion* (SD), where the attacker removes (hides) observations of events that actually occurred in the plant. By doing so, the attacker seeks to deceive the supervisor into issuing a control command that leads the plant into an unsafe state. Moreover, in AE-attacks, the attacker arbitrarily enables a subset of the control commands sent to the actuators, with the goal of driving the plant to an unsafe state.

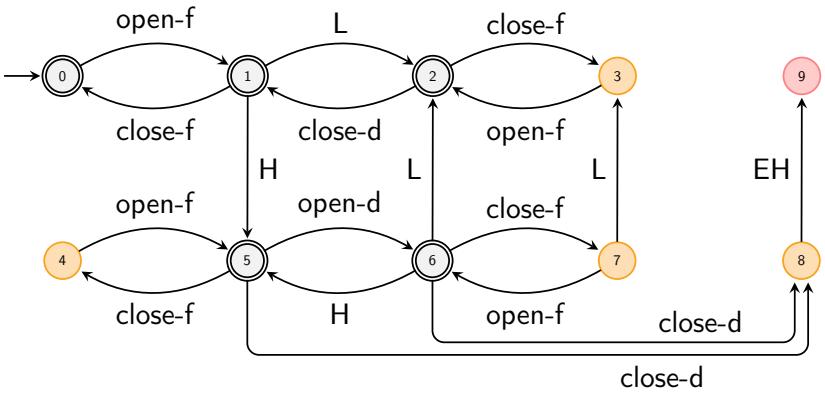
We apply these concepts to the water tank example shown next.

**Example 3.** *We adapt the water tank example from Su (2018) and Lin and Su (2021) as our running example. Within this system, a monitored water level generates sensor observations* L, H, *and* EH *for low, high, and extremely high levels, respectively. The plant includes two valves: an* inflow (supply) *valve with events* open-f *and* close-f, *and an* outflow (drain) *valve with events* open-d *and* close-d, *as shown in Figure 8a. The plant model $G$, with $\Sigma_{uc} = \{\mathsf{L}, \mathsf{H}, \mathsf{EH}\}$, $\Sigma_c = \{\text{open-f}, \text{close-f}, \text{open-d}, \text{close-d}\}$, $X_m = \{0, 1, 2, 5, 6\}$ and $x_{us} = 9$ is shown in Figure 8b.*

Figure 8 – Water tank system and its modeling.




(a) Water tank system




(b) Plant model $G$

Source: Designed by the author (2026).

*The safety specification requires the water level to remain between* L *and* H *by opening and closing the drain valve. More importantly, it requires that* EH *is never reached. Specifically, when the supervisor observes* H, *it decreases the level by issuing* open-d, *thereby draining water and avoiding* EH; *when the supervisor observes* L, *it closes the drain valve* close-d *so the tank can refill through the inflow supply.*

*A nominal supervisor $S$ that enforces this specification can be obtained from the plant automaton by removing states 3, 4, 7, 8, and 9. The resulting supervisor disables any control choices that could drive the plant to the* EH *level while allowing actions that keep the level within* L *and* H.

We now introduce useful notation to represent attacker actions, characterize the closed-loop behavior of the system under attacks, and show attack traces that illustrate how a sensor/actuator attacker can drive the plant to damage while remaining covert.

**Remark 1.** *While the nominal system is assumed to be fully observable, sensor deletion and insertion attacks effectively make the attacked closed-loop system to become partially observable from the supervisor's perspective.*

### 4.1.1 Notation

For simplicity, we assume that the plant $G$ has one unsafe (critical) state, denoted $x_{us} \in X$, as shown in Figure 8 (highlighted in red). Supervisor $S$ ensures that this state is not reachable in the closed-loop system $S/G$.

The subsets of vulnerable events are defined as follows:

- $\Sigma_{c,v} \subseteq \Sigma_c$: vulnerable controllable events (subject to AE-attacks)
- $\Sigma_{o,v} \subseteq \Sigma_o$: vulnerable observable events (subject to SI/SD attacks)

To distinguish the attacker's actions from legitimate events issued by $S$ and generated by $G$, the attacker's actions are captured by a new set of events, $\Sigma_A = \Sigma_{en} \cup \Sigma_{ins} \cup \Sigma_{del}$, with

- $\Sigma_{en} = \{\sigma_{en} \mid \sigma \in \Sigma_{c,v}\}$ models actuator enablement;
- $\Sigma_{ins} = \{\sigma_{ins} \mid \sigma \in \Sigma_{o,v}\}$ models sensor insertion; and
- $\Sigma_{del} = \{\sigma_{del} \mid \sigma \in \Sigma_{o,v}\}$ models sensor deletion.

Events in $\Sigma_{ins} \cup \Sigma_{del}$ represent *attacker actions* and are therefore assumed to be unobservable by the supervisor. In the case of sensor insertion, when an attacker executes $\sigma_{ins}$, the supervisor does not observe the insertion action itself, but rather perceives the corresponding original event $\sigma$. As a result, the supervisor updates its state according to its nominal dynamics as if $\sigma$ had occurred, while the plant state remains unchanged. In the case of sensor deletion, the attacker hides the observation of an event that actually occurred in the plant, and thus no observation is received by the supervisor.

Similarly, events in $\Sigma_{en}$ are also unobservable by the supervisor. When an attacker enables $\sigma_{en} \in \Sigma_{en}$ and the event is executed in the plant, the supervisor does not observe the enablement action itself, but rather perceives the occurrence of the corresponding nominal

event $\sigma$. Since $\sigma$ occurs despite not being enabled by the supervisor, this observation constitutes an unexpected behavior, allowing the supervisor to infer that an attack has occurred.

The total alphabet of the attacked system is $\Sigma_{all} = \Sigma \cup \Sigma_A$.

We use a masking function $\mathcal{M} : \Sigma_{all} \to \Sigma$ that retrieves the original event associated with an attacked event, as in the work of Meira-Góes, Marchand and Lafortune (2023). Specifically, $\mathcal{M}(\sigma_{en}) = \sigma$, $\mathcal{M}(\sigma_{del}) = \sigma$, and $\mathcal{M}(\sigma_{ins}) = \sigma$. Additionally, $\mathcal{M}(\sigma) := \sigma$, $\forall \sigma \in \Sigma \setminus \Sigma_A$. This function highlights that each attacked event represents the corrupted version of a legitimate event in $\Sigma$. We extend $\mathcal{M}$ to strings in the usual way: for any $s = \sigma_1\sigma_2 \ldots \sigma_n \in \Sigma_{all}^*$, we define $\mathcal{M}(s) = \mathcal{M}(\sigma_1)\mathcal{M}(\sigma_2) \ldots \mathcal{M}(\sigma_n)$.

Next, we characterize the closed-loop system in the presence of AE, SI, and SD attacks.

### 4.1.2 Attacked closed-loop system model

In this section, we construct a DES model of the closed-loop system that explicitly captures the possible actions of a sensor and actuator attacker. The modeling procedure augments both the plant and the supervisor models with additional events and transitions that represent attack actions, allowing the interaction between the attacker, plant, and supervisor to be analyzed within the supervisory control framework. The construction follows modeling approaches commonly used in the literature for representing cyberattacks in CPDES, particularly those proposed by Carvalho et al. (2018), Lima et al. (2022) and Meira-Góes, Marchand and Lafortune (2023).

#### *4.1.2.1 Attacked plant*

The attacked plant $G^a$ is derived from the plant $G$ by incorporating the effects of AE, SI, and SD attacks.

The state space of the attacked plant, $X_{G^a}$, is augmented to account for the sensor insertion attack mechanism. A state $x \in X$ is extended with a new parallel state $(x, \sigma_{ins})$ for every $\sigma \in \Sigma_{o,v}$. This augmented state represents an intermediate condition where the plant physically remains in state $x$, but the attacker has successfully injected the fake event $\mathcal{M}(\sigma_{ins})$ into the supervisor's observation channel.

**Definition 3** (Attacked plant)**.** *Given the plant $G$ and $\Sigma_A$, we define the attacked plant model $G^a := (X_{G^a}, \Sigma_{all}, f_{G^a}, \Gamma_{G^a}, x_{0,G^a})$, where $X_{G^a} := X \cup (X \times \Sigma_{ins})$, $x_{0,G^a} := x_0$, and*

$$f_{G^a}(x,\sigma) \coloneqq \begin{cases} f(x,\mathcal{M}(\sigma)), & \textit{if } \sigma \notin \Sigma_{ins},\ x \in X, \textit{and } f(x,\mathcal{M}(\sigma))! \\ (x,\sigma), & \textit{if } \sigma \in \Sigma_{ins},\ \textit{and } x \in X \\ x_1, & \textit{if } x = (x_1,\sigma_1), \textit{and } \sigma = \mathcal{M}(\sigma_1) \\ \textit{undefined}, & \textit{otherwise.} \end{cases} \tag{1}$$

The transition function $f_{G^a}$ captures how the plant evolves under both legitimate and attacked events. When legitimate events occur, i.e., $\sigma \in \Sigma$, the plant behaves according to its nominal dynamics through the mapping $\mathcal{M}(\sigma) = \sigma$. For sensor insertion attacks ($\sigma_{ins} \in \Sigma_{ins}$), the mechanism is represented by two transitions: first, a transition from $x$ to an auxiliary state $(x, \sigma_{ins})$ labeled by $\sigma_{ins}$, and then a transition from $(x, \sigma_{ins})$ back to $x$ labeled by the corresponding legitimate event $\sigma = \mathcal{M}(\sigma_{ins})$. This construction mimics the situation in which the attacker injects a fictitious observation of $\sigma$ while the physical plant remains in the same state. The sequence $\sigma_{ins}\sigma$ thus uniquely encodes the insertion of event $\sigma$ at state $x$: the first event represents the attacker's action, and the second represents the falsified observation perceived by the supervisor.

The remaining attack types, i.e., actuator enablement ($\sigma_{en} \in \Sigma_{en}$) and sensor deletion ($\sigma_{del} \in \Sigma_{del}$), are modeled as follows. For each nominal transition labeled by $\sigma \in \Sigma_{c,v}$, a parallel transition labeled by $\sigma_{en}$ is added to represent the attacker's action of allowing the execution of a vulnerable controllable event even when it is disabled by the supervisor. Similarly, for each transition labeled by $\sigma \in \Sigma_{o,v}$, a parallel transition labeled by $\sigma_{del}$ is added to capture the deletion of that observable event from the supervisor's perception. In both cases, the resulting transition leads to the same successor state as the nominal one.

#### *4.1.2.2 Attacked Supervisor*

Let $S = (X_S, \Sigma, f_S, \Gamma_S, x_{0,S})$ be the realization of the admissible supervisor for $G$, and $\Sigma_A$ the set of events that model the attacker actions. The attacked supervisor is defined as follows:

**Definition 4** (Attacked supervisor $S^a$)**.** *Given the supervisor $S$, and $\Sigma_A$, we define $S^a \coloneqq (X_{S^a}, \Sigma_{all}, f_{S^a}, \Gamma_{S^a}, x_{0,S^a})$, where $X_{S^a} \coloneqq X_S$, $x_{0,S^a} \coloneqq x_{0,S}$, and*

$$f_{S^a}(x,e) \coloneqq \begin{cases} f_S(x,e), & \textit{if } e \in \Sigma \ \textit{ and } \ f_S(x,e)!, \\ x, & \textit{if } \Big(e \in (\Sigma_{en} \cup \Sigma_{uc}) \wedge \\ & \quad f_S(x,\mathcal{M}(e))\neg! \Big) \vee \ \Big(e \in \Sigma_{ins}\Big) \\ & \quad \vee \ \Big(e \in \Sigma_{del} \ \wedge \ f_S(x,\mathcal{M}(e))!\Big), \\ \textit{undefined}, & \textit{otherwise.} \end{cases} \tag{2}$$

The attacked supervisor $S^a$ extends the nominal supervisor $S$ by adding transitions that model the attacker's possible actions. For each event $e \in (\Sigma_{en} \cup \Sigma_{uc})$, a self-loop labeled by $e$ is added to every state $x \in X_{S^a}$ such that $\mathcal{M}(e) \notin \Gamma_S(x)$. These self-loops capture the enablement of controllable events through AE-attacks, as well as unexpected occurrences of uncontrollable events that may happen when the supervisor loses synchronization with the plant. Similarly, for each insertion event $\sigma_{ins} \in \Sigma_{ins}$, a self-loop labeled by $\sigma_{ins}$ is added to every state to model sensor readings injected by the attacker. Finally, for each deletion event $\sigma_{del} \in \Sigma_{del}$, a self-loop labeled by $\sigma_{del}$ is added to every state $x$ for which the corresponding observable event is defined, i.e., $\mathcal{M}(\sigma_{del}) \in \Gamma_S(x)$. This construction allows $S^a$ to preserve the nominal control logic of $S$ while accounting for potential desynchronization due to both sensor and actuator attacks.

#### *4.1.2.3 Attacked closed-loop system*

The attacked closed-loop system $S^a/G^a$ is obtained by computing $S^a \parallel G^a$. We define the set of unsafe states in $S^a/G^a$ as $X_{us} \coloneqq \{x \in X_{S^a/G^a} | \exists s \in \mathcal{L}(S^a/G^a) \text{ s.t. } x_{us} = f_{G^a}(x_{0,G^a}, s)\}$.

With respect to the system presented in Example 3, we assume that the vulnerable controllable events are given by $\Sigma_{c,v} = \{\mathsf{close\text{-}d}\}$ and that the vulnerable observable events are given by $\Sigma_{o,v} = \{\mathsf{L}, \mathsf{H}\}$. The attacked closed-loop system for Example 3 is shown in Figure 9.

### 4.1.3 Damaging and Covert Attacks

Not all attacks on a CPDES are equally critical. Some attacks may be easily detectable by the supervisor, while others may not lead the system to reach an unsafe state. In this work, we focus on attacks that are both *damaging* and *covert*, as we define in the following.

First, we define the unsafe region $\widehat{X}_{us}$ as the set of states of the attacked closed-loop system from which unsafe states can be reached through uncontrollable events, i.e., $\widehat{X}_{us} \coloneqq \{x \in X_{S^a/G^a} \mid \exists s \in \Sigma_{uc}^* \text{ such that } f_{S^a/G^a}(x, s) \in X_{us}\}$, including the trivial case in which the state itself is unsafe.

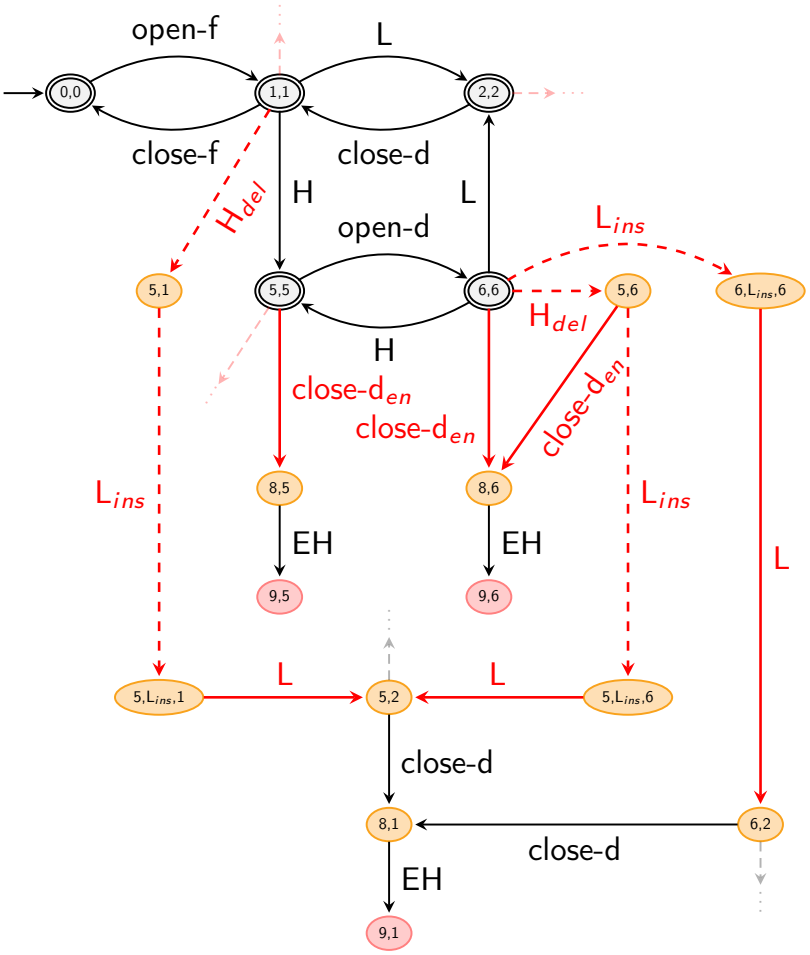
We refer to as an *attack trace* any sequence $t \in \mathcal{L}(S^a/G^a)$ that contains at least one event from $\Sigma_A$.

**Definition 5** (Damaging attack)**.** *An attack trace $t \in \mathcal{L}(S^a/G^a)$ is said to be* damaging *if there exists a continuation $s \in \Sigma_{all}^*$ such that $ts \in \mathcal{L}(S^a/G^a)$ and $f_{S^a/G^a}(x_0, ts) \in \widehat{X}_{us}$.*

**Definition 6** (Covert attack)**.** *An attack trace $t \in \mathcal{L}(S^a/G^a)$ is* covert *if one of the following conditions hold:*

*(i) $\mathcal{M}(t) \in \mathcal{L}(S/G)$, i.e., the attacked trace is indistinguishable from a nominal trace; or*

Figure 9 – Part of the attacked closed-loop system model $S^a/G^a$ for Example 3.




Source: Designed by the author (2026).

*(ii)* $t = t'e$ *for some* $t' \in \mathcal{L}(S/G)$ *and* $e \in \Sigma_A$ *such that* $\mathcal{M}(t) \notin \mathcal{L}(S/G)$ *and* $f_{S^a/G^a}(x_{0,S^a/G^a}, t) \in \widehat{X}_{us}$*, i.e., the attack trace is only distinguishable from nominal traces once the system has entered the unsafe region.*

Examples of damaging and covert attack traces include:

(i) **Sensor attacks:** $t_1 =$ open-f $\mathsf{H}_{del}$ $\mathsf{L}_{ins}$ L close-d EH, and $t_2 =$ open-f H open-d $\mathsf{L}_{ins}$ L close-d EH. In both cases, the attacker injects L to cause the supervisor to close the drain while the level is high.

(ii) **Actuator attacks:** $t_3 =$ open-f H $\mathsf{close\text{-}d}_{en}$ EH, and $t_4 =$ open-f H open-d $\mathsf{close\text{-}d}_{en}$ EH. In both cases, close-d is enabled by the attacker against the supervisor's control decision.

(iii) **Joint sensor–actuator attack:** $t_5 =$ open-f H open-d $\mathsf{H}_{del}$ $\mathsf{close\text{-}d}_{en}$ EH, which combines sensor deletion and actuator enablement.

Note that the attacked closed-loop system shown in Figure 9 does not satisfy the notion of *GF-safe controllability* introduced by Carvalho et al. (2018). Even though attacks may eventually become detectable, detection may occur only after the system has already entered the unsafe region. As a result, no control action exists that can prevent damage once the attack is detected.

This work explicitly targets such systems. That is, we consider CPDES in which attack detection may be inherently late, and where classical supervisory control mechanisms are insufficient to guarantee safety once attacks are detected. Our goal is therefore to provide a protection mechanism that prevents the system from reaching unsafe states even when attacks remain *covert*, i.e., indistinguishable from nominal behavior.

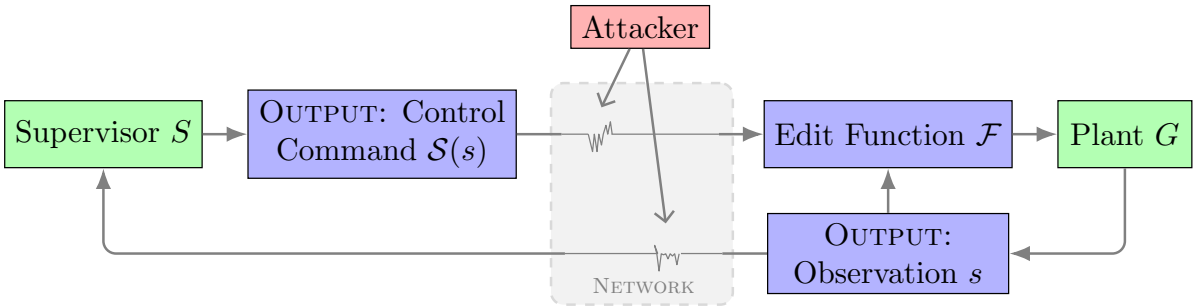
## 4.2 ACTUATOR EDIT FUNCTIONS AND LIVE PROTECTABILITY

Motivated by the limitations discussed above, we pose the main problem addressed in this chapter: *How can integrity (i.e., never reaching unsafe states) be enforced in non-GF-safe controllable CPDES while preserving liveness (non-blocking behavior)?*

In such systems, attacks may eventually become detectable. However, detection can occur only after the system has already entered the unsafe region. Thus, classical supervisory control mechanisms, which rely on disabling controllable events after attack detection, are insufficient to preserve integrity.

In this work, we investigate the use of *edit functions* to preserve integrity in CPDES that are not GF-safe controllable. Edit functions have been widely studied in the context of opacity enforcement to modify sensor observations. Here we adopt a different perspective. We employ *actuator edit functions* at the actuator side, modifying control commands before they are executed in the plant. Figure 10 illustrates the proposed system framework using actuator edit functions.

Figure 10 – System framework with actuator edit functions.




Source: Designed by the author (2026).

This approach follows from the system dynamics under attack: unsafe states can only be reached when a controllable event that deviates from the nominal controlled behavior is physically executed by the plant. Sensor attacks and actuator enablement

attacks differ in their mechanisms, but both can cause the execution of such unsafe controllable events. Therefore, if unsafe controllable events can be prevented from being executed at the plant level (e.g., through the use of an edit function), unsafe states can be avoided regardless of whether they are triggered by actuator attacks or by sensor deception attacks. This suggests that integrity preservation may still be achievable even in systems that do not satisfy GF-safe controllability by employing actuator edit functions.

It is important to emphasize that the objective of the proposed framework is not to redesign the control architecture by relocating the supervisor closer to the physical plant or eliminating the communication network. In many practical cyber-physical systems, the use of communication networks is inherent to the system design, enabling scalability, modularity, remote monitoring, and integration with higher-level management layers. Removing the network or replicating the entire supervisor locally may be infeasible due to hardware limitations, cost constraints, or architectural dependencies. This issue becomes even more evident in distributed control architectures, where multiple supervisors coordinate the operation of different subsystems through shared communication networks. Therefore, our goal is to protect CPDES that already rely on networked supervisory control, without altering their architectural assumptions. The proposed actuator edit function acts as a protective layer that enforces safety and liveness properties at the actuator interface, complementing (rather than replacing) the existing supervisory control structure.

### 4.2.1 Actuator Edit functions

The edit function $\mathcal{F}$ is implemented at the actuator side (plant level), between the (possibly corrupted) control commands channel and the plant $G$. Consequently, it has access to uncompromised sensor observations, allowing it to determine the true plant state independently of the supervisor's potentially corrupted state estimation. The edit function's role is to mediate the execution of controllable events based on the actual plant state, independently of the supervisor's state estimation. We formally define it as follows:

**Definition 7** (Actuator Edit Function)**.** *Let $X_{safe} = X_{S^a/G^a} \setminus \widehat{X}_{us}$ be the set of safe states in the attacked closed-loop system. The actuator edit function is a mapping $\mathcal{F} : X_{safe} \times \Sigma_c \to \Sigma_c \cup \{\epsilon\}$ defined for any state $x = (x_{S^a}, x_{G^a}) \in X_{safe}$ and issued controllable event $\sigma \in \Sigma_c$ as:*

$$\mathcal{F}(x,\sigma) \coloneqq \begin{cases} \sigma & \text{if } f_{S^a/G^a}(x,\sigma) \in X_{safe} \\ \sigma' & \text{if } f_{S^a/G^a}(x,\sigma) \in \widehat{X_{us}} \text{ and } \exists \sigma' \in \mathit{Cand}(x,\sigma) \\ \varepsilon & \text{otherwise} \end{cases} \tag{3}$$

*where the set of candidate substitutions $\mathit{Cand}(x,\sigma)$ is defined as*

$$\mathit{Cand}(x,\sigma) \coloneqq \{\sigma' \in \Gamma_{G^a}(x_G) \cap \Sigma_c \mid f_{S^a/G^a}(x,\sigma') \in X_{safe}\}. \tag{4}$$

In words, when a controllable event $\sigma$ issued by the supervisor is safe at the current plant state (i.e., its execution does not lead to the unsafe region), the edit function allows it to be executed unchanged. However, when the issued event is unsafe at the current plant state, the edit function intervenes by substituting the unsafe event with another feasible controllable event $\sigma'$ whose execution does not drive the system toward the unsafe region.

### 4.2.2 Live Protectability against covert attacks

We now formalize the conditions under which an actuator edit function guarantees integrity and liveness under covert sensor and actuator attacks. We start by defining the set of potentially damaging controllable events.

**Definition 8** (Critical events)**.** *Let $X_{safe}$ denote the safe region of the attacked closed-loop system $S^a/G^a$. A controllable event $\sigma \in \Sigma_c$ is said to be* critical *at a state $x \in X_{safe}$ if $f_{S^a/G^a}(x, \sigma) \in \widehat{X}_{us}$.*

*That is, executing $\sigma$ at state $x$ places the system on a trajectory that inevitably leads to the unsafe state.*

Critical events capture the physical mechanism through which damage may occur, independently of how the execution of such events is triggered. In particular, this definition applies equally to actuator enablement attacks and to sensor attacks that deceive the supervisor into issuing unsafe control commands.

**Definition 9** (Live Protectability)**.** *Let $S^a/G^a$ be the attacked closed-loop system. Each state of $S^a/G^a$ can be written as $x = (x_S, x_G) \in X_{S^a} \times X_{G^a}$. $S^a/G^a$ is said to be* live protectable *if, for every state $x = (x_S, x_G) \in X_{safe}$ and every critical event $\sigma \in \Sigma_c$ at $x$, there exists an alternative controllable event $\sigma' \in \Sigma_c \cap \Gamma_{G^a}(x_{G^a})$ such that $f_{S^a/G^a}(x, \sigma') \notin \widehat{X}_{us}$ and there exists a string $t \in \Sigma^*_{all}$ such that:*

1. $f_{S^a/G^a}(x, \sigma' t) \in X_{S^a/G^a,m}$;
2. $f_{S^a/G^a}(x, \sigma' u) \notin \widehat{X}_{us},\ \forall u \in \bar{t}$.

Intuitively, live protectability captures the necessary and sufficient conditions that a system must hold so that actuator edit functions can be applied to prevent unsafe behavior while preserving system liveness. Whenever a controllable event that could drive the system toward the unsafe region is about to be executed, live protectability requires the existence of an alternative controllable event that is feasible at the plant and whose execution keeps the system outside the unsafe region. This alternative event must preserve liveness in the sense that the system remains coaccessible: after its execution, there must exist a continuation that reaches a marked state without ever visiting the unsafe region.

While an actuator edit function is defined locally at each state $x \in X_{safe}$, a global edit function is obtained by collecting the local mappings over all safe states. A local edit

function that guarantees integrity while preserving liveness in a given state is said to be a *live-protecting edit function*. If all edit functions are live-protecting, then the global edit function is said to be a *global live-protecting edit function*.

**Theorem 1.** *A* global live-protecting edit function *exists if and only if the attacked closed-loop system $S^a/G^a$ is live protectable.*

*Proof:* ($\Longrightarrow$) Assume a *global live-protecting edit function* exists. By definition, this implies that for every state $x \in X_{safe}$, there exists a local live-protecting edit function. For any critical event $\sigma$ at $x$, this function substitutes such critical event by a non critical one $\sigma'$ that ensures $f_{S^a/G^a}(x, \sigma') \notin \widehat{X}_{us}$ and preserves coaccessibility. This satisfies all conditions of Definition 9, meaning the system is live protectable.

($\Longleftarrow$) Assume the system is live protectable. For every state $x \in X_{safe}$, we can define a local live-protecting edit function using the substitution $\sigma'$ guaranteed by Definition 9. The collection of these local mappings for all states forms a *global live-protecting edit function*, proving its existence. ■

## 4.3 SYNTHESIZING GLOBAL LIVE-PROTECTING EDIT FUNCTIONS

The notion of live protectability introduced in the previous section naturally leads to both a verification and a synthesis problem. Given an attacked closed-loop system $S^a/G^a$, one may wonder whether the conditions of Definition 9 are satisfied. Similarly, whenever these conditions hold, it is desirable to explicitly construct a global live-protecting edit function that enforces integrity while enforcing system liveness.

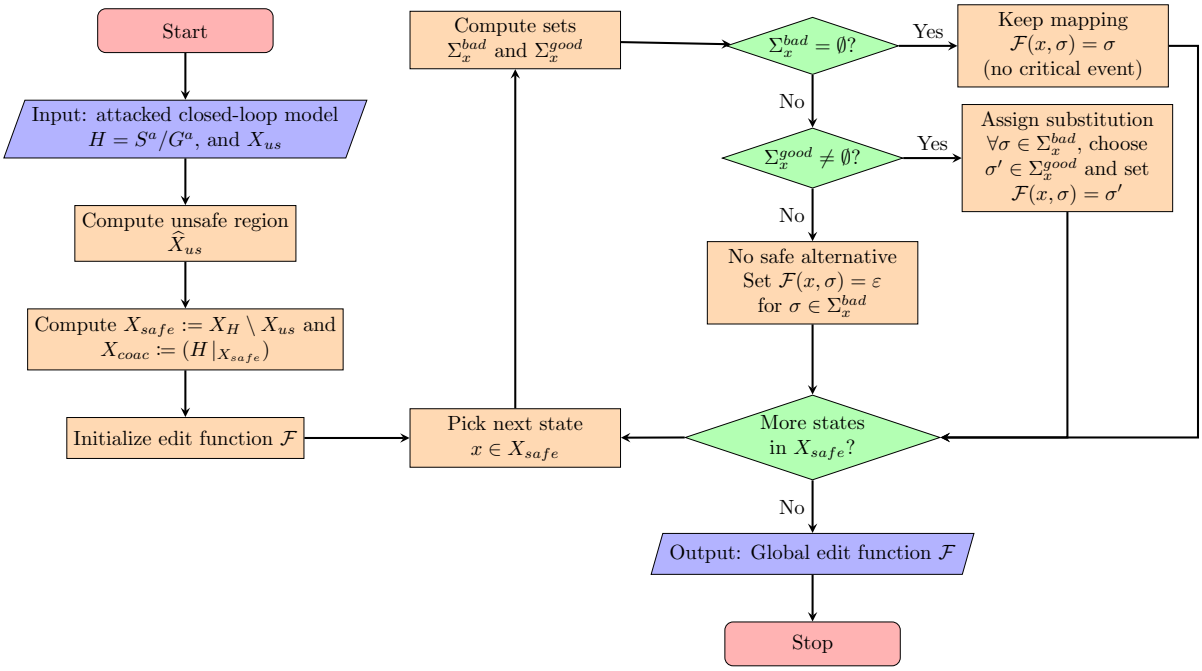
To address these problems, we propose a solution framework that synthesizes a global live-protecting edit function for the attacked system $S^a/G^a$. This unified approach simultaneously solves both the verification and synthesis problems, as shown in the sequel.

### 4.3.1 Synthesis algorithm

The synthesis procedure is illustrated in the flowchart shown in Figure 11, which summarizes the main steps required to construct a global live-protecting edit function for the attacked closed-loop system $S^a/G^a$. The corresponding formal algorithm is presented in Algorithm 1.

First, for each state $x \in X_{safe}$, the algorithm identifies the set of critical controllable events $\Sigma^x_{bad}$, i.e., those whose execution would drive the system into the unsafe region $\widehat{X}_{us}$. Whenever such events exist, the algorithm checks whether there also exists at least one alternative controllable event whose execution remains within $X_{safe}$ and preserves coaccessibility, captured by the set $\Sigma^x_{good}$. Thus, if $\Sigma^x_{bad} = \emptyset$ at given state $x$, the event enablements remain unchanged. If $\Sigma^x_{bad} \neq \emptyset$ and $\Sigma^x_{good} \neq \emptyset$, then a suitable substitution for the critical event does exist. If $\Sigma^x_{bad} \neq \emptyset$ and $\Sigma^x_{good} = \emptyset$ for some $x \in X_{safe}$, then

Figure 11 – Flowchart representation of Algorithm 1.




Source: Designed by the author (2026).

no substitution satisfying the conditions of Definition 9 exists at that state. Hence, $\varepsilon$ is assigned to that local edit function, representing a supression of such critical event, which prevents the system from reaching the unsafe region, but continuity is not guaranteed.

**Remark 2** (Complexity analysis of Algorithm 1)**.** *The time complexity of the actuator edit function synthesis is $O(|X_H| + |f|)$, where $|X_H|$ denotes the number of states and $|f|$ represents the total number of transitions in the automaton $H$. This linear complexity follows from the structure of the algorithm:*

1. *The pre-computation of the sets $\widehat{X}_{us}$, $X_{safe}$ and $X_{coac}$ can be performed via standard graph search procedures, each requiring $O(|X_H| + |f|)$ time.*

2. *The synthesis phase consists of a traversal of the safe state space. For each state $x \in X_{safe}$, the algorithm inspects the outgoing controllable transitions in $\Gamma_H(x) \cap \Sigma_c$. Since each transition is processed once in the inner loop, this step is linear in the number of states $|X_{safe}|$ and transitions $|f|$.*

3. *The construction of the local sets $\Sigma_{bad}^x$ and $\Sigma_{good}^x$ involves only event-level membership tests and transition lookups, whose cost is constant.*

*Therefore, the overall computational complexity is linear in the size of the attacked closed-loop system.*

**Algorithm 1:** Synthesis of an actuator edit function

**Input** : $H := S^a/G^a = (X_H, \Sigma_{all}, f_H, x_{0,H}, X_{m,H})$, $\Sigma_c$, $X_{us}$
**Output** : $\mathcal{F}$
Compute $\widehat{X}_{us} := \{x \in X_H \mid \exists t \in \Sigma_{uc}^* \text{ s.t. } f_H(x,t) \in X_{us}\}$
Compute $X_{safe} := X_H \setminus \widehat{X}_{us}$
Compute $X_{coac} := \texttt{CoAc}(H|_{X_{safe}})$
**foreach** $x \in X_{safe}$ **do**
  $\Sigma^x_{bad} \leftarrow \emptyset$
  $\Sigma^x_{good} \leftarrow \emptyset$
  **foreach** $\sigma \in \Gamma_H(x) \cap \Sigma_c$ **do**
    **if** $f_H(x,\sigma) \in \widehat{X}_{us}$ **then**
      $\Sigma^x_{bad} \leftarrow \Sigma^x_{bad} \cup \{\sigma\}$
    **else if** $f_H(x,\sigma) \in X_{coac}$ **then**
      $\Sigma^x_{good} \leftarrow \Sigma^x_{good} \cup \{\sigma\}$
  **if** $\Sigma^x_{bad} = \emptyset$ **then**
    **foreach** $\sigma \in \Gamma_H(x) \cap \Sigma_c$ **do**
      $\mathcal{F}(x,\sigma) \leftarrow \sigma$
  **else if** $\Sigma^x_{good} = \emptyset$ **then**
    **foreach** $\sigma \in \Gamma_H(x) \cap \Sigma_c$ **do**
      **if** $\sigma \in \Sigma^x_{bad}$ **then**
        $\mathcal{F}(x,\sigma) \leftarrow \varepsilon$
      **else**
        $\mathcal{F}(x,\sigma) \leftarrow \sigma$
  **else**
    **foreach** $\sigma \in \Sigma^x_{bad}$ **do**
      $\mathcal{F}(x,\sigma) \leftarrow \sigma' \in \Sigma^x_{good}$
    **foreach** $\sigma \in (\Gamma_H(x) \cap \Sigma_c) \setminus \Sigma^x_{bad}$ **do**
      $\mathcal{F}(x,\sigma) \leftarrow \sigma$
**return** $\mathcal{F}$

**Theorem 2.** *Let $\mathcal{F}$ be the global actuator edit function synthesized by Algorithm 1. Then, $\mathcal{F}$ is a* global live-protecting edit function *if and only if* $\mathcal{F}(x,\sigma) \neq \varepsilon, \quad \forall x \in X_{safe}, \forall \sigma \in \Gamma_H(x) \cap \Sigma_c$.

*Proof:* We prove both directions by contradiction.

($\Longrightarrow$) Assume that $\mathcal{F}$ is a *global live-protecting edit function*, but that there exist $x \in X_{safe}$ and $\sigma \in \Gamma_H(x) \cap \Sigma_c$ such that $\mathcal{F}(x,\sigma) = \varepsilon$. By Algorithm 1, the assignment $\mathcal{F}(x,\sigma) = \varepsilon$ can only occur when $\Sigma^x_{bad} \neq \emptyset$ and $\Sigma^x_{good} = \emptyset$, and only for events $\sigma \in \Sigma^x_{bad}$. Hence, $\sigma$ is a critical event at $x$, and there exists no controllable event $\sigma' \in \Gamma_H(x) \cap \Sigma_c$ such that $f_H(x,\sigma') \in X_{coac}$, where $X_{coac} = \mathrm{CoAc}(H|_{X_{safe}})$. Since $X_{coac}$ consists of states that are coaccessible to $X_{m,H}$ in the restricted system $H|_{X_{safe}}$, the absence of such a $\sigma'$ means that no controllable substitute exists whose execution both avoids $\widehat{X}_{us}$ and

preserves liveness (i.e., admits a continuation to a marked state without leaving $X_{safe}$). This contradicts the assumption that $\mathcal{F}$ is live-protecting.

($\Longleftarrow$) Assume that $\mathcal{F}(x,\sigma) \neq \varepsilon$ for all $x \in X_{safe}$ and all $\sigma \in \Gamma_H(x) \cap \Sigma_c$, but that $\mathcal{F}$ is *not* a global live-protecting edit function. Then there exists some state $x \in X_{safe}$ such that the local mapping of $\mathcal{F}$ at $x$ is not live-protecting. In particular, there exists a critical event $\sigma \in \Sigma^x_{bad}$ such that no controllable event $\sigma' \in \Gamma_H(x) \cap \Sigma_c$ satisfies the conditions of Definition 9; equivalently, there is no $\sigma'$ with $f_H(x,\sigma') \in X_{coac}$, and thus $\Sigma^x_{good} = \emptyset$. Since $\sigma$ is critical, $\Sigma^x_{bad} \neq \emptyset$. Therefore, Algorithm 1 executes the branch $\Sigma^x_{bad} \neq \emptyset$ and $\Sigma^x_{good} = \emptyset$ at state $x$ and assigns $\mathcal{F}(x,\sigma) = \varepsilon$ for every $\sigma \in \Sigma^x_{bad}$, in particular for the critical event $\sigma$ identified above. This contradicts the assumption that $\mathcal{F}(x,\sigma) \neq \varepsilon$ for all $(x,\sigma)$. Hence, $\mathcal{F}$ must be a global live-protecting edit function. ■

### 4.3.2 Synthesized Edit Functions for the Water Tank Example

We now revisit Example 3 to illustrate how Algorithm 1 constructs a global live-protecting actuator edit function. Such an edit function thwarts the damaging and covert attack traces identified in Section 4.1 while preserving system liveness.

Recall that the unsafe plant state corresponds to the water level reaching EH. In the attacked closed-loop system $H = S^a/G^a$, Algorithm 1 first computes the unsafe region $\widehat{X}_{us}$. The synthesis is then performed over the safe region $X_{safe} \coloneqq X_H \setminus \widehat{X}_{us}$.

#### *4.3.2.1 Critical event and its substitution*

For the water-tank system, $\Sigma_c = \{\text{open-f}, \text{close-f}, \text{open-d}, \text{close-d}\}$. Applying Algorithm 1 to the attacked closed-loop model in Fig. 9 reveals that the only critical controllable event in $X_{safe}$ is close-d when executed while the physical water level is high. Intuitively, closing the drain valve at level H leads to the uncontrollable evolution to EH. Formally, for any $x \in X_{safe}$ whose water level corresponds to H, we have $f_H(x, \text{close-d}) \in \widehat{X}_{us}$, and thus close-d $\in \Sigma^x_{bad}$. At these same states, the candidate event close-f belongs to $\Sigma^x_{good}$, since closing the fill valve maintains the evolution within $X_{safe}$ and preserves coaccessibility to $X_{m,H}$ in the restricted system $H|_{X_{safe}}$.

#### *4.3.2.2 Synthesized edit function*

Algorithm 1 synthesizes the following local mapping at every state $x \in X_{safe}$ with water level H: $\mathcal{F}(x, \text{close-d}) = \text{close-f}$. As an example, consider the attack trace $t_1 = \text{open-f}\ \mathsf{H}_{del}\ \mathsf{L}_{ins}\ \mathsf{L}\ \text{close-d}\ \mathsf{EH}$. The edit function substitutes close-d by close-f, so that the unsafe region is avoided and coaccessibility to $X_{m,H}$ is preserved. All other controllable events remain unchanged: $\mathcal{F}(x,\sigma) = \sigma, \quad \forall \sigma \in \Sigma_c \setminus \{\text{close-d}\}$.

Since $\Sigma^x_{good} \neq \emptyset$ at all such states, the algorithm never assigns $\varepsilon$, and the resulting mapping is a global live-protecting edit function (Theorem 2). This implies that the

attacked closed-loop system $S^a/G^a$ is *live protectable*, according to Definition 9.

### 4.3.3 Discussion

The edit function synthesized in this example guarantees that unsafe states remain unreachable in the plant, even under sensor deception or actuator enablement attacks. This safety property holds provided the system satisfies the necessary and sufficient conditions for live-protectability. Moreover, since $\mathcal{F}(x, \sigma) \neq \varepsilon$ for all $x \in X_{safe}$ and all $\sigma \in \Gamma_H(x) \cap \Sigma_c$, integrity is preserved *without sacrificing liveness*, meaning that the closed-loop system remains nonblocking.

Preserving liveness is essential when recovery strategies are considered. While recent works have proposed methods to recover attacked systems (Alves; Pena, 2022; Cavalcanti et al., 2025; Oliveira et al., 2025), these procedures are ineffective if the protection mechanism itself drives the system into a deadlock. By preventing such scenarios, the proposed approach maintains the structural conditions necessary for post-attack recovery, even though the recovery process itself is outside the scope of this framework.

Although the protection mechanism may lead to temporary desynchronization between the supervisor and the plant, the system's integrity and liveness are maintained. Since the edit function operates at the actuator level and relies on the original (unchanged) sensor observations, integrity and liveness are preserved even during periods of desynchronization. Consequently, the approach creates favorable conditions for higher-level recovery procedures to be applied.

The method proposed in this chapter builds upon and extends the conceptual foundations established in our preliminary works (Oliveira et al., 2023b; Oliveira et al., 2023c; Oliveira et al., 2024a; Oliveira et al., 2024b), where cipher-based protection mechanisms were employed to mitigate actuator attacks. By adopting the edit function formalism in this framework, we provide a more generalized and mathematically rigorous characterization of the event substitution logic previously explored through cryptography.

Finally, we emphasize that the proposed approach does *not* address attack detection. The edit function operates transparently at the actuator level based on the actual plant state, without attempting to inform the supervisor about the presence of an attacker. Integrating detection and resynchronization mechanisms with actuator-level edit functions remains a direction for future research.

In the next chapter, we introduce a robust recovery framework designed to drive attacked systems toward a specific subset of states for post-attack operation, ensuring operational continuity for CPDES under attacks.

# 5 ROBUST RECOVERY OF CPDES UNDER ACTUATOR ATTACKS

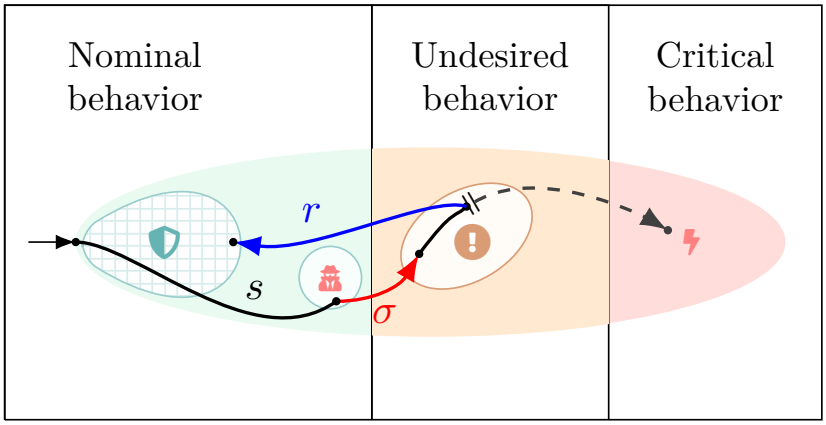
This chapter addresses the second perspective explored in this thesis, which concerns *robust recovery*, that is, the ability to recover from attacks by driving the system to a subset of states that represent essential behavior, while remaining robust to further attacks.

In contrast to the protection mechanism developed in the previous chapter, the robust recovery problem considered in this chapter adopts the traditional supervisory control architecture. In the edit-function approach, a plant-side device modifies actuator commands before they reach the plant to enforce integrity and liveness. In the robust recovery approach, recovery is achieved by synthesizing a supervisor that restricts the behavior of the attacked system after attack detection so that safe operation can be restored. This formulation allows the robust recovery problem to be addressed within the standard supervisory control framework.

The recovery problem studied in this chapter considers only actuator enablement attacks. This restriction is adopted as an initial step toward the development of a more general recovery framework. Recovery under sensor attacks is left as a direction for future work. For simplicity, the system is assumed to be AE-safe controllable (Carvalho et al., 2018), which ensures the system is able to avoid unsafe states after attack detection.

Figure 12 illustrates the method proposed in this chapter. After nominal behavior $s$, the attacker enables event $\sigma$ to disrupt the nominal operation. Upon attack detection, our *robust-recovery strategy* achieves three objectives: it (i) prevents the system from reaching unsafe states, (ii) recovers the system from this undesired behavior to a subset of states defined for post-attack operation (sequence of events $r$), and (iii) enforces a robust controller against attacks to keep the system within this subset of states (hatched area within the controlled behavior). By doing so, our approach ensures operational continuity while preserving both integrity and availability, which are critical cybersecurity aspects.

Figure 12 – Robust recovery strategy.




Source: Adapted from (Oliveira et al., 2025)

The results presented in this chapter extend our preliminary work on robust recovery

presented in the 64th IEEE Conference on Decision and Control (Oliveira et al., 2025), which provides an initial characterization of how post-attack operation can be enforced in attacked CPDES. However, its formulation imposes structural restrictions on both the target region and the recovery paths that limit its applicability in more general settings.

To motivate the need for a more flexible formulation, this chapter begins with a motivating example based on a manufacturing system composed of a robotic arm and two machines that process different types of parts. This example illustrates scenarios in which recovery necessarily involves vulnerable states and target regions where attacks remain feasible, yet do not compromise the intended post-attack operation.

Building on this motivating example, we introduce a novel robust recovery framework. Rather than requiring the target region to be attack-free, the proposed approach characterizes regions in which attacks, even if feasible, do not interfere with the system's core functionality. Moreover, vulnerable states may be visited during recovery provided that reachability to the target region can be guaranteed regardless of the attacker's actions. While the recovery formulation continues to focus primarily on AE-attacks, we also account for the impact of actuator disablement (AD) attacks during the recovery process, which were not considered in the preliminary framework. Finally, while the preliminary results characterize robust recoverability in terms of the existence of recovery paths, the framework developed in this chapter formalizes robust recovery at the language level, thereby providing a more rigorous mathematical characterization of recovery guarantees.

## 5.1 MOTIVATION

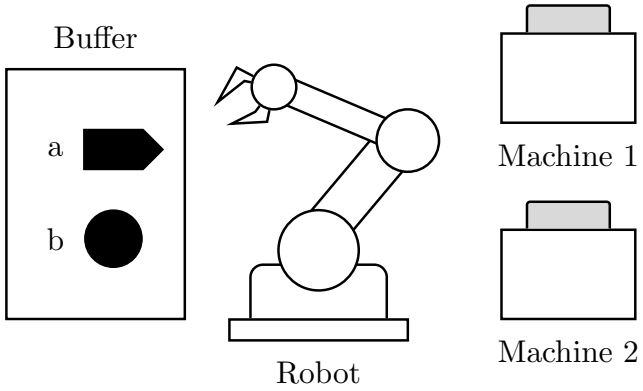
As a motivating example, we consider a manufacturing system comprised of two machines and a robot, as depicted in Figure 13. In this system, type-$a$ and type-$b$ parts are processed by machines 1 and 2, respectively. Once processing is complete, the finished parts are automatically placed in an exit buffer by the machines.

### 5.1.1 Controlled system

Under normal operation, the robot follows a specific sequence: it takes one part ($a$ or $b$) from the entry buffer and places it on the corresponding machine (machine 1 for type-$a$, machine 2 for type-$b$). Then, the machine is allowed to begin processing. Once processing is finished, the robot can grab another part from the entry buffer and then continue the operation cycle. For simplicity and without loss of generality, we assume that the system operates sequentially, with only one part processed at a time.

The system operates under a critical safety specification: the robot must never deposit a part on a machine that is already working. Violating this specification represents an unsafe behavior that damages both the part and the machine.

Figure 13 – A simple manufacturing system example.




Source: Designed by the author (2026).

### 5.1.2 Attacked system

Consider that machine 2 has been compromised by an actuator attacker. Thus, such an attacker can enable it to start working even when no part has been deposited on it. This leads to an imminent unsafe scenario: if the robot places a type-$b$ part on machine 2 while it is already working (due to the attack), the safety specification is violated and damage occurs.

This example illustrates a fundamental challenge in CPDES: while traditional control systems assume that actuators behave as commanded, cyberattacks can break this assumption and lead to unsafe behavior.

### 5.1.3 Existing approaches and their limitations

Existing mitigation approaches primarily focus on preventing unsafe behaviors after attacks, typically through one of the following strategies:

(i) Complete system shutdown (Carvalho et al., 2018): upon attack detection, M1 and M2 are shutdown;

(ii) Selective event disabling (Lima et al., 2019): a security supervisor only disables specific non-vulnerable events when there is imminent risk of unsafe behavior. For instance, in our running example, this approach would specifically prevent the robot from placing the type-$b$ part in the attacked machine, as the robot is not vulnerable to attacks;

(iii) Selective event enabling (Wang et al., 2020; Oliveira et al., 2024a): when disabling events is insufficient to avoid unsafe states, this strategy enables alternative safe behaviors. This is achieved either probabilistically, by enabling events that are likely to lead to safe states after attacks (Wang et al., 2020), or deterministically,

by substituting, in the plant, an attacked event that would lead to an unsafe state with a safe one (Oliveira et al., 2024a).

While these approaches preserve system integrity by preventing unsafe behaviors, they cannot ensure *operational continuity*.

Methods for synthesizing *robust supervisors* (Ma; Cai, 2022a; Meira-Góes; Marchand; Lafortune, 2023) guarantee robustness against attacks by permanently restricting the system's behavior to avoid all vulnerable system components. In our running example, a robust supervisor would disable all operations involving machine 2 from the beginning, producing only type-$a$ parts on machine 1. This ensures safety but sacrifices valid behavior even when no attacks occur.

Recent recovery methods (Alves; Pena, 2022; Cavalcanti et al., 2025; Sakata et al., 2023; Sakata et al., 2024) investigate strategies to restore the system to its nominal behavior after attacks. While some approaches assume immediate attacker isolation or the availability of attack-proof backup controllers, these assumptions may not hold in practice. In standard supervisory control systems without such guarantees, restoring nominal behavior leaves the system vulnerable to repeated disruptions from persistent attackers. In our running example, once the attacked machine returns to normal operation and the system resumes its nominal behavior, a new enablement attack can occur, forcing repeated detection-recovery cycles.

### 5.1.4 Robust Recovery: Our Contribution

To address the aforementioned limitations, we propose an approach that enables the system to: (i) detect attack occurrences and prevent unsafe behaviors, (ii) recover from such attacks to a subset of states defined for post-attack operation that preserves essential functionality, and (iii) continue operating under a post-attack behavior that, while possibly more restrictive than nominal operation, is resilient to persistent attacks.

In our running example, after attack detection, our method:

(i) avoids unsafe behaviors by preventing the robot from placing the part on the running machine 2;

(ii) executes recovery by guiding the robot to return the type-$b$ part to the entry buffer;

(iii) enforces an attack-resilient post-attack behavior by guiding the system to process only type-$a$ parts on machine 1, since machine 2 remains vulnerable to attacks.

This post-attack behavior is said to be attack-resilient because any future attacks on machine 2 can no longer affect the new operation of the system. This is because it no longer interacts with that machine. In this sense, the system maintains both safety and functionality under persistent attacks.

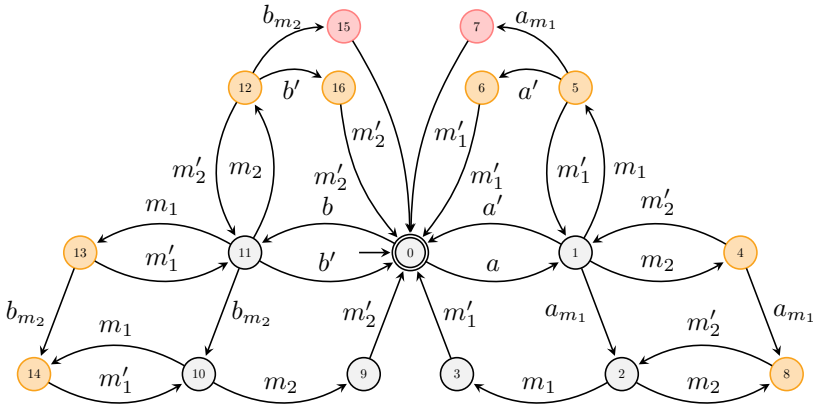
Table 1 – Event descriptions for the running example

| Event | Description |
|---|---|
| $a$ | Robot grabs a type-$a$ part |
| $b$ | Robot grabs a type-$b$ part |
| $a'$ | Robot puts back a type-$a$ part |
| $b'$ | Robot puts back a type-$b$ part |
| $a_{m_1}$ | Robot places the type-$a$ part on Machine 1 |
| $b_{m_2}$ | Robot places the type-$b$ part on Machine 2 |
| $m_1$ | Machine 1 starts |
| $m_1'$ | Machine 1 finishes processing |
| $m_2$ | Machine 2 starts |
| $m_2'$ | Machine 2 finishes processing |

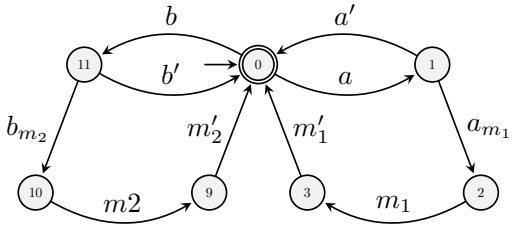
Source: Designed by the author (2026).

**Example 4.** *We model the manufacturing system illustrated in Figure 13, which operates sequentially, with only one part processed at a time. The plant model $G$ is shown in Figure 14a, where the set of unsafe states is $X_{us} = \{7, 15\}$ and the set of vulnerable controllable events is $\Sigma_{c,v} = \{m_2\}$. The supervisor model $S$ is depicted in Figure 14b, enforcing the desired behavior and ensuring that unsafe states are not reached. A description of each event used in the model is provided in Table 1.*

Figure 14 – Plant and Supervisor models for Example 1.




(a) Plant $G$




(b) Supervisor $S$

Source: Designed by the author (2026).

## 5.2 CPDES UNDER ACTUATOR ATTACKS

### 5.2.1 Actuator enablement attacks (AE-attacks)

In this work, we consider an attacker who has the ability to modify a subset of controllable events $\Sigma_{c,v} \subseteq \Sigma_c$, representing vulnerable actuators from the plant $G = (X, \Sigma, f, \Gamma, x_0, X_m)$. In the case of AE-attacks, the attacker enables an event that was originally disabled by the supervisor. By doing so, the attacker can lead the system to unsafe states.

To model the system under attack, two new models are constructed: the attacked plant model $G^a$ and the attacked supervisor model $S^a$, both derived from the original models $G$ and $S$, respectively. Let $\Sigma^a_{c,v} \coloneqq \{\sigma^a : \sigma \in \Sigma_{c,v}\}$ be the set of vulnerable events that can be enabled by an attacker. Then $\sigma^a$ represents the occurrence of $\sigma$ in the plant when it is originally disabled by the supervisor, but enabled by an attacker.

The attacked plant $G^a = (X, \Sigma \cup \Sigma^a_{c,v}, f_{G^a}, \Gamma, x_0, X_m)$ is a copy of $G$ with new events representing enablement attacks.

The transition function $f_{G^a}$ extends the transition function of $G$ as follows:

$$\begin{aligned} &\forall x \in X, \forall \sigma \in \Sigma : f(x,\sigma)! \Rightarrow f_{G^a}(x,\sigma) = f(x,\sigma), \\ &\forall x \in X, \forall \sigma \in \Sigma_{c,v} : f(x,\sigma)! \Rightarrow f_{G^a}(x,\sigma^a) = f(x,\sigma). \end{aligned} \tag{5}$$

In words, parallel transitions labeled with $\sigma^a$ are added for each original transition labeled with $\sigma \in \Sigma_{c,v}$, expressing the attacker's ability to enable these events.

The attacked supervisor model $S^a = (X_S \cup \{x_A\}, \Sigma_S \cup \Sigma^a_{c,v}, f_{S^a}, \Gamma_S, x_{0,S}, X_{S,m})$ includes a new state $x_A$ that defines the supervisor's control actions after an attack is detected. Specifically, from any state $x \in X_S$, any attacked event $\sigma \in \Sigma^a_{c,v}$ triggers a transition to $x_A$. As we assume a full observation setting, these attacks are immediately identified upon occurrence. Once in $x_A$, the supervisor enables all events $\sigma \in \Sigma \cup \Sigma^a_{c,v}$, establishing a maximally permissive baseline from which we aim to synthesize robust-recovery strategies later on.

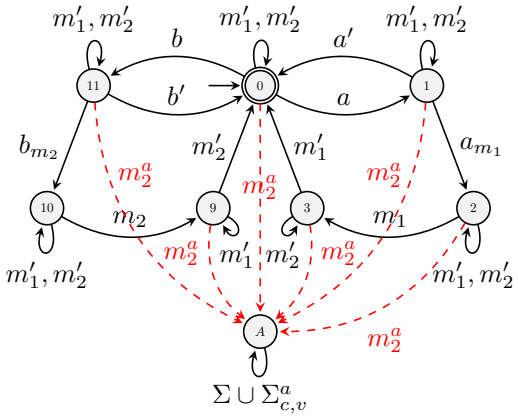
The transition function $f_{S^a}$ is defined as

$$f_{S^a}(x,\sigma) \coloneqq \begin{cases} f_S(x,\sigma) & \text{if } f_S(x,\sigma)! \\ x_A & \text{if } f_S(x,\sigma)\neg! \wedge \sigma \in \Sigma^a_{c,v} \\ x & \text{if } (f_S(x,\sigma)\neg! \wedge \sigma \in \Sigma_{uc}) \vee \\ & (x = x_A \wedge \sigma \in \Sigma \cup \Sigma^a_{c,v}). \end{cases} \tag{6}$$

The model of the attacked supervisor $S^a$ is shown in Figure 15. The model of the closed-loop system under attack $S^a/G^a$ is obtained by computing the parallel composition $S^a \| G^a$as illustrated in Figure 16.

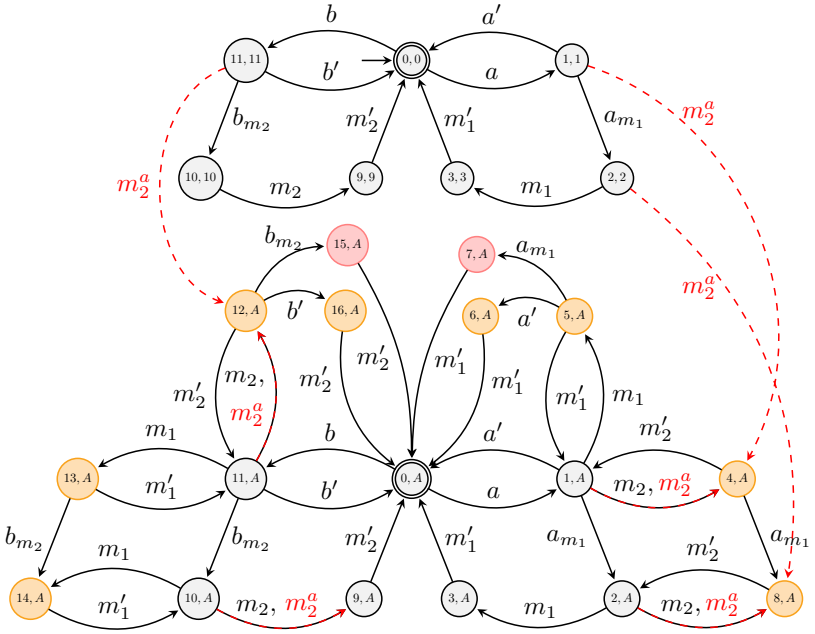
It is worth noting that this attacked system satisfies the property of AE-safe controllability introduced by Carvalho et al. (2018). This property holds when it is possible

Figure 15 – The model of the Supervisor under attack $S^a$.




Source: Designed by the author (2026).

Figure 16 – The model of the attacked closed-loop system $S^a/G^a$.




Source: Designed by the author (2026).

to detect any attack occurrence and subsequently disable non-vulnerable controllable events to prevent the plant from reaching unsafe states. In our example, if an attacker enables event $m_2$ in state 11, the supervisor can disable the non-vulnerable event $b_{m_2} \in \Sigma_c \setminus \Sigma_{c,v}$ to prevent transition to the unsafe state 15.

Hereinafter, we assume: (i) the supervisor $S$ is *safe*, i.e., it prevents the system from reaching unsafe states $x \in X_{us}$, and (ii) the attacked closed-loop system $S^a/G^a$ is AE-safe controllable, i.e., we can detect and prevent/block the attacker from reaching unsafe states.

### 5.2.2 Actuator disablement attacks (AD-attacks)

In AD-attacks, the attacker disables events that were originally enabled by the supervisor. A fundamental challenge with such attacks is their inherent difficulty of detection. Unlike AE-attacks, where the supervisor can identify an attack when an unexpected event occurs (since the event was supposed to be disabled), AD-attacks are characterized by the absence of an expected action. When the supervisor enables an event that is subsequently disabled by an attacker, the supervisor cannot immediately distinguish between: (i) the plant naturally choosing not to execute the enabled event (which is feasible within the supervisory control framework); or (ii) an actual attack that prevents the execution of such an event.

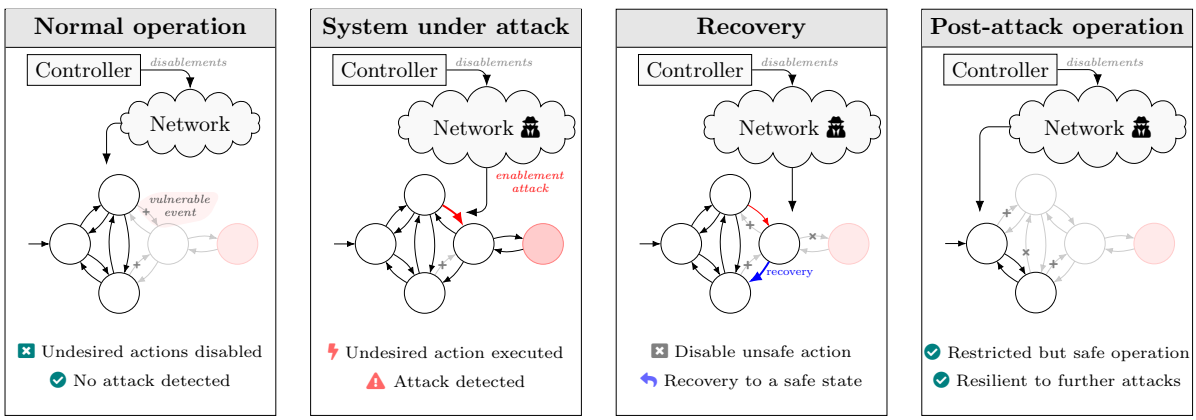
Despite this difficulty in detection, previous works in the literature have noted that AD-attacks cannot lead to a violation of safety, as described by $X_{us}$. Instead, they can only reduce the overall controlled behavior. Thus, for enforcing safety, it is sufficient to consider AE-attacks (Carvalho et al., 2018; Meira-Góes; Marchand; Lafortune, 2023). Nevertheless, we point out that these attacks are crucial in the context of recovery. This is because they can lead to blocking, where the system is no longer able to progress or complete critical operations such as recovery procedures.

Based on the above, rather than explicitly modeling AD-attacks, we address this vulnerability in our formulation of robust-recovery strategies in the next section.

## 5.3 ROBUST-RECOVERY OF CPDES UNDER AE-ATTACKS

In this section, we pose the problem of recovering a CPDES from AE-attacks and then enforcing a robust post-attack control strategy. The fundamental steps for this approach are depicted in Figure 17. In this scenario, the system is allowed to execute its full valid behavior in the absence of attacks. Once an attack is detected, the system should

Figure 17 – Robust recovery strategy




Source: Designed by the author (2026).

not only avoid unsafe states, but also transition to a *target region* (a properly selected subset of states of $G$) defined for post-attack operation. Considering that the plant $G$ contains such a region, this problem raises a fundamental question: *How can we transition the system from attack detection to the target region even under continued attacks?*

To address this question, we formalize: (i) the target region to be reached after attacks, (ii) the definition of robust-recovery strategies and (iii) the problem of synthesizing such strategies to transition the attacked system to this target region.

### 5.3.1 Target region - Our recovery goal

The target region, denoted by $G_{target}$, is a subautomaton of $G$ that specifies the intended behavior to be enforced after attack detection. In our previous work (Oliveira et al., 2025), we assumed $G_{target}$ to be attack-free, which guaranteed safety in the post-attack operation, but limited applicability. Here, we consider that attacks can occur within $G_{target}$, provided they *do not compromise the post-attack behavior*. This relaxation increases the applicability of our method to scenarios in which attack-free regions cannot be guaranteed.

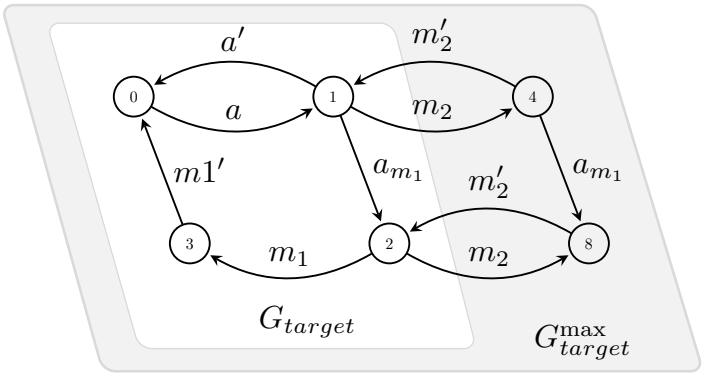
We assume $G_{target}$ exhibits two key characteristics: (i) it captures a post-attack behavior where the system maintains essential functionality, and (ii) within this region, attacks may still occur but do not disrupt the post-attack behavior. The key insight is that although attacks on compromised components cannot be prevented, a post-attack control strategy can be designed to rely solely on non-vulnerable components.

For instance, in our running example in Figure 13, machine 2 is assumed to be vulnerable to attacks. In this case, $G_{target}$ specifies a behavior that depends solely on machine 1, which is not vulnerable. However, since compromised components remain part of the system model, attackers can still enable vulnerable events that would drive the system outside $G_{target}$.

To address this characteristic, we consider a larger region $G^{\max}_{target} \sqsupset G_{target}$ that satisfies an *invariance property* (Baier; Katoen, 2008). This property ensures that the system always remains within $G^{\max}_{target}$ under any feasible event, including those enabled by an attacker. Additionally, from any state $x \in X_{G^{\max}_{target}} \setminus X_{G_{target}}$ the system can always return to $G_{target}$.

**Example 5.** *Consider the plant $G$ presented in Figure 14a with $\Sigma_{c,v} = \{m_2\}$. $G_{target}$ is formed by states $\{0, 1, 2, 3\}$, while $G^{\max}_{target}$ includes states $\{0, 1, 2, 3, 4, 8\}$, as shown in Figure 18.*

Figure 18 – $G_{target}$ and $G^{\max}_{target}$




Source: Designed by the author (2026).

*$G_{target}$ models the system behavior with only the robot and machine 1: robot grabs a type-a part ($a$), places it on machine 1 ($a_{m_1}$), machine 1 processes it ($m_1$), and then finishes ($m_1'$).*

*Suppose an attacker enables $m_2$ in state 1, causing the system to transition to state 4, leaving $G_{target}$. Since our control strategy ignores machine 2, the system continues with its intended behavior, which is to process type-a parts on machine 1, eventually returning to $G_{target}$. In this sense, machine 2 remains part of the system model but becomes* ***virtually disconnected*** *from the intended post-attack behavior.*

**Remark 3.** *In this work, we assume that $G_{target}$ and $G^{\max}_{target}$ are given. Therefore, we do not impose additional constraints on their definition. Additionally, determining the existence of such regions is beyond the scope of this work, as our focus is on verifying and synthesizing robust-recovery strategies from attack detection to the given target region.*

### 5.3.2 Robust recovery from attacks

We formalize *robust recovery* in such a way that, after an attack, the system transitions to the target region $G_{target}$ despite possible subsequent attacks.

We start by formalizing concepts related to attack detection.

**Definition 10** (Detection language)**.** *Given the attacked closed-loop system $S^a/G^a$, the detection language is defined as*

$$\mathcal{L}_{\text{det}} \coloneqq \left\{ s\sigma \in \mathcal{L}(G) \;\middle|\; s \in \mathcal{L}(S/G),\ \sigma \in \Sigma_{c,v}, s\sigma^a \in \mathcal{L}(S^a/G^a) \right\}. \tag{7}$$

In words, the detection language $\mathcal{L}_{det}$ consists of all strings $s$ from the nominal behavior followed by an attacked event $\sigma^a \in \Sigma^a_{c,v}$. Based on $\mathcal{L}_{det}$, we now define the set of states from the attacked closed-loop system $S^a/G^a$ where attack detection is achieved.

**Definition 11** (Detection states)**.** *Given an attacked system $S^a/G^a$ and its detection language $\mathcal{L}_{det}$, the set of detection states is defined as*

$$X_{det} \coloneqq \left\{ x \in X_{S^a/G^a} \;\middle|\; \exists s\sigma \in \mathcal{L}_{det}, f_{S^a/G^a}(x_{0,S^a/G^a}, s\sigma^a) = x \right\}. \tag{8}$$

The set of detection states is composed of those immediately reached as a direct consequence of AE-attacks. Since these attacks are observable by the attacked supervisor $S^a$, we assume that once such states are reached, the attack is detected. Our objective is then, for each detection state $x \in X_{det}$, to design recovery strategies that drive the system to $G_{target}$.

To this end, we now formalize the set of shortest strings from a detection state to $G_{target}$.

**Definition 12** (Shortest strings reaching $G_{target}$)**.** *Let $\mathcal{L}(G, x_d)$ denote the language generated by the plant $G$ starting from a detection state $x_d \in X_{det}$. For $x_d$, the set of all shortest strings to reach $G_{target}$ is*

$$\mathcal{L}_{min}(G, x_d) \coloneqq \left\{ s \in \mathcal{L}(G, x_d) \;\middle|\; \begin{array}{l} f(x_d, s) \in X_{G_{target}} \text{ and } \forall t \in \overline{s} \setminus \{s\}, \\ f(x_d, t) \notin X_{G_{target}} \end{array} \right\}. \tag{9}$$

Thus, $\mathcal{L}_{min}(G, x_d)$ is the language that contain all strings that reach the target region from $x_d$ for the first time. By *shortest*, we imply that no proper prefix of $s$ reaches $X_{G_{target}}$.

While multiple paths may lead the system from a given detection state to the target region $G_{target}$, we are interested in those that guarantee successful recovery even under persistent attacks, thereby making the recovery robust. The set of such valid recovery paths is called a *robust-recovery language*.

**Definition 13** (Robust-recovery language)**.** *A non-empty, prefix-closed language $K \subseteq \overline{\mathcal{L}_{min}(G, x_d)}$ is said to be a robust-recovery language if the following conditions hold:*

1) $\forall s \in K, \exists t \in K/s$ *such that* $f(x_d, st) \in X_{G_{target}}$.

2) $\forall s \in K,\ \forall e \in \Sigma_{c,v},\ se \in K \implies \exists e' \in \Sigma \setminus \Sigma_{c,v}$ *such that* $se' \in K$.

3) $\forall s \in K, f(x_d, s) \notin X_{us}$.

4) $\forall s \in K, \forall e \in (\Sigma_{uc} \cup \Sigma_{c,v}), se \in \mathcal{L}(G, x_d) \implies se \in K$.

*Condition 1* expresses that from every state reached by a string in $K$, there exists a continuation within $K$ that leads the system to $G_{target}$. Thus, $K$ is non-blocking with respect to $X_{G_{target}}$.

*Condition 2* enforces robustness of the recovery language by requiring that, for every vulnerable event along a recovery sequence, there exists a continuation that starts with a non-vulnerable event and remains within $K$. As a result, if a vulnerable event is disabled by an attacker during recovery, the system still has an alternative sequence leading to $G_{target}$. This guarantees robustness against AD-attacks.

*Condition 3* establishes a safety invariance property: no sequence in $K$ reaches an unsafe state.

*Condition 4* requires $K$ to be controllable with respect to $\mathcal{L}(G, x_d)$ and the event set $\Sigma_{uc} \cup \Sigma_{c,v}$. This means that any uncontrollable or vulnerable event that is feasible in the plant must keep the system within $K$, ensuring that recovery remains possible after such events. Combined with Condition 3, this implies that even if uncontrollable or vulnerable events occur during recovery, they cannot drive the system into unsafe states $x \in X_{us}$. This property satisfies the necessary and sufficient condition for a supervisor to enforce $K$ (Cassandras; Lafortune, 2021).

Collectively, these four conditions ensure that a robust-recovery language (i) is non-blocking with respect to $X_{G_{target}}$, (ii) contains a non-vulnerable alternative continuation whenever a vulnerable event appears in it, (iii) ensures safety invariance, and (iv) is closed under uncontrollable and vulnerable events that may occur in the plant. This characterization provides the conditions for robust-recoverability of a given detection state.

With *robust-recovery languages* defined, we can formally present the definition of an *AE-robustly recoverable system.*

**Definition 14** (AE-robustly recoverable system)**.** *An attacked system $S^a/G^a$ is said to be AE-robustly recoverable if, for every detection state $x_d \in X_{det}$, there exists a non-empty, prefix-closed, robust-recovery language $K_{x_d} \subseteq \overline{\mathcal{L}_{min}(G, x_d)}$.*

**Example 6.** *Consider the attacked closed-loop system $S^a/G^a$ shown in Figure 16, where $X_{G_{target}} = \{0, 1, 2, 3\}$. From each detection state $x_d \in X_{det}$, a distinct robust-recovery language $K_{x_d} \subseteq \overline{\mathcal{L}_{min}(G, x_d)}$ exists.*

*From $(4, A)$, the recovery language is $K_{(4,A)} = \overline{\{\, m'_2,\ a_{m_1} m'_2 \,\}}$, with two paths that lead the system to $X_{G_{target}}$. Both paths rely only on non-vulnerable events and satisfy all conditions for robust-recovery languages. From $(8, A)$, $K_{(8,A)} = \overline{\{\, m'_2 \,\}}$. Thus, $m'_2$ is the only event available for recovery, which directly reaches a target state. From $(12, A)$, the recovery language is*

$$K_{(12,A)} = \overline{\{\, (b'\, m'_2) \ +\ (m'_2\, ((m_1 m'_1) \ +\ (m_2 m'_2))^*\, b') \,\}}. \tag{10}$$

*In this case, recovery can be achieved either by executing $b'\, m'_2$ or, after the initial $m'_2$, by performing any combination of the loops $((m_1 m'_1) + (m_2 m'_2))^*$ before completing recovery through $b'$. Note that $K_{(12,A)}$ satisfies Condition 2 of Definition 13: although $m_2$ is a vulnerable event, an alternative continuation starting with the non-vulnerable event $b'$ exists from state $(11, A)$. This ensures that recovery remains possible even if $m_2$ is disabled by an attacker*[1]*.*

*Note that from $(12, A)$, the string $s = b_{m_2}\, m'_2$ reaches $X_{G_{target}}$ but its prefix $b_{m_2}$ passes through the unsafe state $(15, A) \in X_{us}$. This violates Condition 3 of Definition 13. Thus, $b_{m_2} m'_2$ is excluded from $K_{(12,A)}$.*

[1] The notation $(b'\, m'_2) \ +\ (m'_2\, ((m_1 m'_1) \ +\ (m_2 m'_2))^*\, b')$ follows standard regular-expression syntax.

*Since a robust-recovery language exists for each detection state, the system $S^a/G^a$ is AE-robustly recoverable.*

Building upon robust-recovery languages, we now introduce the notion of robust-recovery strategies. For each string observed from a detection state, a robust-recovery strategy specifies the events that must be enabled to ensure recovery. While such strategies define post-attack behavior towards $G_{target}$, they do not yet constitute the full supervisory solution, as they only take place after the system reaches a detection state.

**Definition 15** (Robust-recovery strategy)**.** *Let $S^a/G^a$ be the attacked closed-loop system, $x_d \in X_{det}$ a detection state, and $K_{x_d} \subseteq \overline{\mathcal{L}_{min}(G, x_d)}$ a robust-recovery language. A robust-recovery strategy is the function $\mathcal{R}_{x_d} : \overline{\mathcal{L}_{min}(G, x_d)} \to 2^{\Sigma}$ defined for each $s \in \overline{\mathcal{L}_{min}(G, x_d)}$ by*

$$\mathcal{R}_{x_d}(s) \coloneqq \{\, \sigma \in \Gamma(f(x_d, s)) \mid s\sigma \in K_{x_d} \,\}. \tag{11}$$

In words, Definition 15 describes how the robust-recovery strategy $\mathcal{R}_{x_d}$ makes control decisions from a detection state: it enables exactly those feasible events that keep the closed-loop system within the robust-recovery language $K_{x_d}$.

**Lemma 1.** *Let $x_d \in X_{det}$ be a detection state, and let $\mathcal{R}_{x_d}$ be the robust-recovery strategy constructed from the robust-recovery language $K_{x_d}$. From $x_d$, the closed-loop behavior of $G$ under $\mathcal{R}_{x_d}$ is characterized by the language $\mathcal{L}(\mathcal{R}_{x_d}/G, x_d)$ defined recursively by*

1. *$\varepsilon \in \mathcal{L}(\mathcal{R}_{x_d}/G, x_d)$;*
2. *$[(s \in \mathcal{L}(\mathcal{R}_{x_d}/G, x_d)) \wedge (se \in \mathcal{L}_{min}(G, x_d)) \wedge (e \in \mathcal{R}_{x_d}(s))] \Leftrightarrow [se \in \mathcal{L}(\mathcal{R}_{x_d}/G, x_d)]$.*

*Then, we have that $\mathcal{L}(\mathcal{R}_{x_d}/G, x_d) = K_{x_d}$.*

*Proof:* The proof follows from Definition 15. For every $s \in \overline{\mathcal{L}_{min}(G, x_d)}$ the strategy $\mathcal{R}_{x_d}$ enables exactly the events $\sigma \in \Gamma(f(x_d, s))$ such that $s\sigma \in K_{x_d}$. Hence, starting from $\varepsilon$, the recursive rules in items 1–2 generate exactly the prefix-closed language obtained by iteratively appending to any string $s$ an event $\sigma$ that remain in $K_{x_d}$. Since $K_{x_d}$ is nonempty and prefix-closed, this generated language coincides with $K_{x_d}$, i.e., $\mathcal{L}(\mathcal{R}_{x_d}/G, x_d) = K_{x_d}$. ■

**Remark 4.** *Since $\mathcal{R}_{x_d}$ inherits the reachability objective encoded in $K_{x_d}$, and $K_{x_d} \subseteq \overline{\mathcal{L}_{min}(G, x_d)}$, the strategy $\mathcal{R}_{x_d}$ stops issuing control decisions once a target state $x \in X_{G_{target}}$ is reached. Therefore, admissibility (i.e., never disabling uncontrollable events) applies until the first visit to $G_{target}$. The long-term behavior after recovery is determined by a supervisor introduced later in this chapter.*

The notion of AE-robust recoverability naturally raises two fundamental questions:

- *Verification:* Is the closed-loop system AE-robustly recoverable, i.e., do robust-recovery languages (and hence strategies) exist for every detection state $x_d \in X_{det}$?
- *Synthesis:* If so, how can we synthesize robust-recovery strategies for every detection state?

These questions correspond to the following formal problem.

**Problem 1** (AE-robust recoverability)**.** *Given an attacked system $S^a/G^a$, the set of detection states $X_{det}$, and a target region $G_{target}$, determine whether $S^a/G^a$ is AE-robustly recoverable with respect to $G_{target}$. Furthermore, synthesize, if it exists, a robust-recovery strategy for every detection state $x_d \in X_{det}$.*

To address these questions, we propose a solution framework that synthesizes a supervisor which preserves nominal behavior under normal conditions and enforces robust-recovery strategies upon detecting attacks. This unified approach simultaneously solves both the verification and synthesis problems.

## 5.4 SYNTHESIZING ROBUST-RECOVERY STRATEGIES

In the following, the problem of synthesizing robust-recovery strategies is formulated as a *supervisory control problem*. Specifically, we construct a control specification $H$ that captures both pre-attack and post-attack behaviors. The resulting supervisor enforces nominal behavior under normal conditions and ensures robust recovery after attacks by implementing the corresponding robust-recovery strategies.

### 5.4.1 Control specification $H$

The specification $H$ is modeled as a state-based specification, meaning the plant is restricted to avoid a subset of undesirable states. Specifically, $H$ is derived from the attacked system model $S^a/G^a$, where the plant follows the nominal controlled behavior before attacks. However, after an attack, the system's behavior expands to the entire behavior of $G$.

First, we define $G^R = S^a/G^a$ as our system of interest, i.e., $G^R$ represents the closed-loop behavior before and after the occurrence of AE-attacks. Let $X_{G_{target}} \subseteq X_{G^R}$ denote the set of states of $G^R$ whose plant component belongs to the target region $G_{target}$.

At this point, we aim to restrict $G^R$ in three ways:

(i) to ensure that the system can reach a state within the target region $G_{target}$,

(ii) to avoid reaching unsafe states, and

(iii) to disable controllable events that would cause the system to leave $G_{target}$ once the system has reached it. While our main goal is to guarantee reachability of

$G_{target}$, remaining there represents the benefit of maintaining the system's essential functionality after an attack.

To encode (i), we redefine the marked states of $G^R$ so that $X_{G^R,m} = X_{S^a/G^a,m} \cup \{(x, A) \mid x \in X_{G_{target}}\}$, meaning that reaching the target region corresponds to satisfying our recovery goal. Next, we construct the control specification $H$ as a modified copy of $G^R$ that enforces requirements (ii) and (iii). The construction of $H$ is summarized in the following definition.

**Definition 16** (Control specification for robust-recovery)**.** *Let $G^R$ be the model of the attacked closed-loop system. The control specification for robust-recovery $H$ is constructed as a copy of $G^R$, with the following additional constraints:*

1. $X_H \coloneqq X_{G^R} \setminus X_{us}$;
2. $f_H \coloneqq f_{G^R}|X_H$;
3. $\forall x \in X_H \cap X_{G_{target}}, \forall \sigma \in \Sigma_c$,

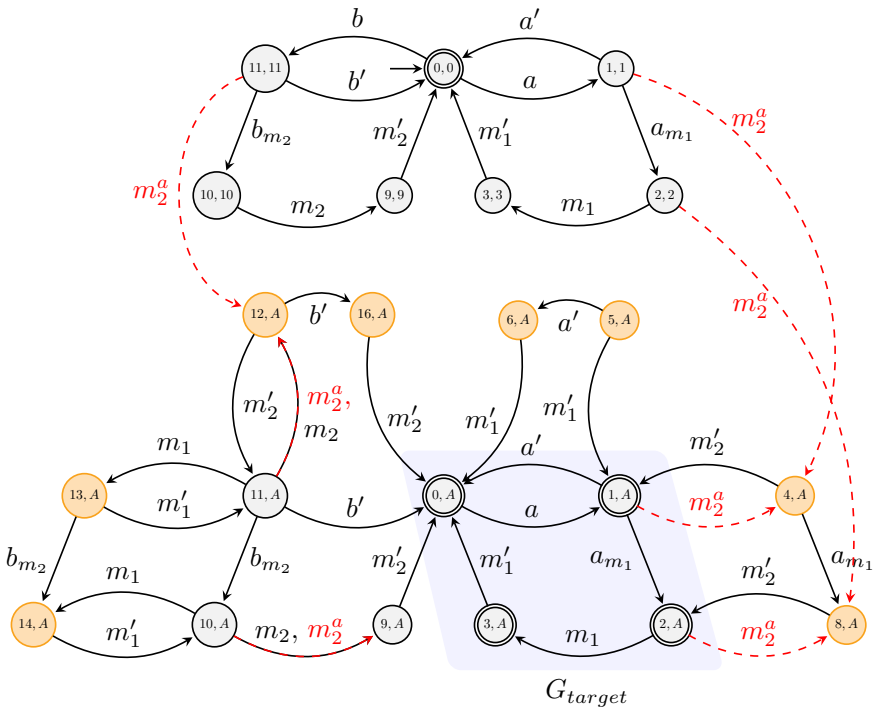
$$f_H(x,\sigma) \coloneqq \begin{cases} f_{G^R}(x,\sigma), & \text{if } f_{G^R}(x,\sigma)! \text{ and } f_{G^R}(x,\sigma) \in X_{G_{target}}, \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

This construction removes unsafe states from $X_H$ and refines the behavior within the target region by disabling controllable transitions leading outside $G_{target}$. The resulting specification $H$ for the running example is shown in Figure 19.

When $G_{target}$ is reachable after attack detection (as required by Condition 1 of Definition 13), the specification $H$ also accounts for the removal of recovery paths that would violate Conditions 3 and 4 (safety invariance and closure under uncontrollable and vulnerable events). In the standard supervisor synthesis procedure, this results in eliminating any path that (i) reaches an unsafe state or contains a sequence of uncontrollable or vulnerable transitions leading to one, or (ii) contains a sequence of uncontrollable or vulnerable transitions from which no continuation to $G_{target}$ exists, thus causing blocking. Because unsafe states are removed from $H$ and synthesis *trims* states that are not coaccessible to $G_{target}$, all paths of the types described in (i) and (ii) necessarily pass through a state removed during synthesis. Consequently, they do not appear in the resulting supervisor.

What $H$ does not account for, however, is the removal of recovery paths that involve a vulnerable event with no valid non-vulnerable alternative, as required by Condition 2 of Definition 13. For example, from detection state $(12, A)$, one possible continuation is $s_1 = m_2' \, b_{m_2} \, m_2 \, m_2'$. However, this sequence contains the vulnerable event $m_2$ and offers no alternative non-vulnerable event from which recovery could continue. If the supervisor enables $m_2$, the attacker could prevent its execution through an AD-attack, causing

Figure 19 – Specification $H$




Source: Designed by the author (2026).

blocking and preventing recovery. In this sense, relying on $H$ alone is not sufficient for our supervisory control solution.

To overcome this limitation, it is necessary to remove, during synthesis, recovery paths that reach a state from which the target region $G_{target}$ can be reached *only* by executing a vulnerable event. In this example, from state $(12, A)$, alternative continuations such as $s_2 = b'\, m_2'$ or $s_3 = m_2'\,(m_1\, m_1')^*\, b'$ belong to a robust-recovery language, as they avoid vulnerable events while still ensuring recovery. Restricting $G^R$ to permit only such sequences prevents the attacker from compromising recovery through AD-attacks and guarantees that the system can safely reach the target region.

A possible solution to achieve this goal is to remove all vulnerable events $\sigma \in \Sigma_{c,v}$ and $\sigma^a \in \Sigma^a_{c,v}$ from the attacked state space in $H$, as addressed by Oliveira et al. (2025). We refer to as *attacked state space* the subset of states of $G^R$ of the form $(x, A)$, corresponding to the system behavior after an AE-attack has been detected. Removing vulnerable events from these states guarantees that only recovery sequences without vulnerable events survive in the synthesized supervisor. However, this approach is more restrictive than our formulation of robust-recovery languages, as it excludes any use of vulnerable events even when safe alternatives exist.

To address this issue, we adopt a modified synthesis procedure to compute *robustly recoverable supervisors* that, from each detection state, enforce robust-recovery languages, thereby permitting vulnerable events whenever they do not compromise recovery.

### 5.4.2 Robustly Recoverable Supervisor

The term *robustly recoverable supervisor* denotes a supervisor that, after an AE-attack, enforces a robust-recovery language to drive the system to the target region $G_{target}$.

**Definition 17** (Robustly recoverable supervisor)**.** *Let $G^R$ be the attacked closed-loop system, and let $X_{det} \subseteq X_{G^R}$ be its set of detection states. For each detection state $x_d \in X_{det}$, let $K_{x_d} \subseteq \overline{\mathcal{L}_{min}(G, x_d)}$ be a robust-recovery language and let $\mathcal{R}_{x_d}$ denote the corresponding robust-recovery strategy as in Definition 15. A supervisor $S^R$ is said to be* robustly recoverable *if, for every detection state $x_d \in X_{det}$, it encodes a $\mathcal{R}_{x_d}$. That is, for all $x_d \in X_{det}$ and all $s \in \mathcal{L}(S^R, x_d)$ such that $f_{S^R}(x_d, s) \notin X_{G_{target}}$, $\Gamma_{S^R}(f_{S^R}(x_d, s)) = \mathcal{R}_{x_d}(s)$.*

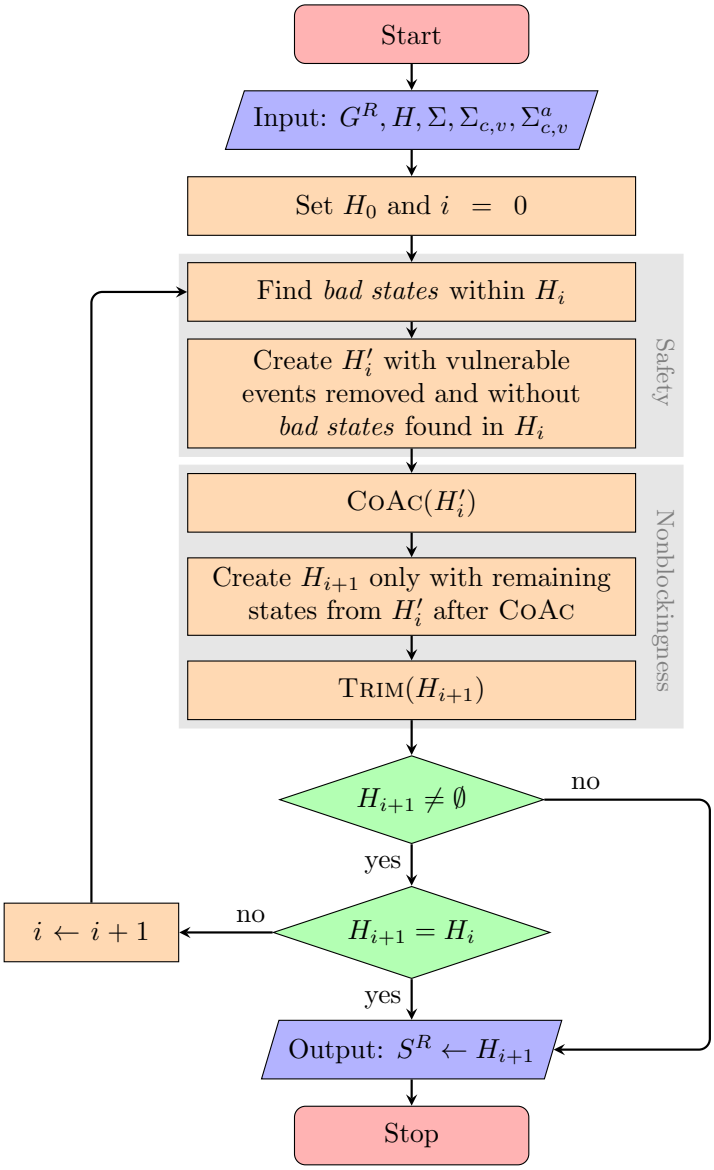
To compute the robustly recoverable supervisor $S^R$, we propose a modified version of the standard supervisory control synthesis algorithm (Ramadge; Wonham, 1987). The algorithm constructs an auxiliary model $H'$, obtained by removing vulnerable events $\Sigma_{c,v} \cup \Sigma^a_{c,v}$ from all states of the attacked state space. It then alternates between safety and non-blockingness procedures over $H$ and $H'$, as depicted in the flowchart in Figure 20.

At each iteration $i$, the algorithm first performs a safety check on $H_i$ to identify bad states, i.e., states where an uncontrollable event is disabled. Then, the auxiliary automaton $H'_i$ is created by removing all vulnerable events from *every state* of the attacked state space, as well as the bad states found in $H_i$. To ensure non-blockingness, the coaccessibility operation $\textsc{CoAc}(H'_i)$ is applied. $H_{i+1}$ is then created without the states that are not coaccessible in $H'_i$, thereby propagating non-blockingness. A trimming step $\textsc{Trim}(H_{i+1})$ is subsequently applied to eliminate any remaining states that are not both accessible and coaccessible. These steps are repeated until a fixpoint is reached, i.e., when $H_{i+1} = H_i$.

The introduction of the auxiliary model $H'_i$ is key to enforcing robust-recovery languages. By removing vulnerable events from all states in the attacked state space, $H'_i$ enables the coaccessibility operation to identify only those states that admit continuations to marked states through non-vulnerable events. Consequently, states that become non-coaccessible in $H'_i$ correspond to situations where recovery would rely solely on vulnerable events. By propagating their removal to $H_{i+1}$, the algorithm guarantees that the resulting supervisor enforces only recovery behaviors that satisfy all conditions of robust-recovery languages. In other words, once a detection state is reached, the synthesized supervisor enforces robust-recovery languages that guide the system safely to $G_{target}$ while remaining resilient to attacks.

**Example 7.** *Consider the specification $H$ shown in Figure 19. At a given iteration $i$, the auxiliary automaton $H'_i$ is constructed by removing all vulnerable events $\Sigma_{c,v} \cup \Sigma^a_{c,v}$ from attacked states. In particular, from state $(10, A)$, events $m_2$ and $m^a_2$ are removed, leaving no continuation from $(10, A)$ to $X_{G_{target}}$. As a consequence, states $(10, A)$*

Figure 20 – Flowchart representation of Algorithm 2.




Source: Designed by the author (2026).

*and* $(14, A)$ *become non-coaccessible in* $H'_i$*, whereas* $(9, A)$ *remains coaccessible. After the coaccessibility step, states* $(10, A)$ *and* $(14, A)$ *are removed when constructing* $H_{i+1}$*. Finally, the trimming operation* $\text{Trim}(H_{i+1})$ *removes state* $(9, A)$*. In the end, the only states of the attacked state space that survive synthesis are the states of* $G_{target}$*, namely* $\{(0, A), (1, A), (2, A), (3, A)\}$*, and those that admit a robust-recovery language leading to* $G_{target}$*, namely* $\{(8, A), (4, A), (11, A), (12, A), (13, A), (16, A)\}$*.*

The algorithm for synthesizing maximally-permissive, non-blocking and robustly recoverable supervisors is given in Algorithm 2. Since $H$ only deletes states and transitions from $G^R$ ($H \sqsubset G^R$), $S^R \sqsubset G^R$ follows by construction. Following the synthesis procedure, we now verify whether $S^R$ satisfies Definition 17. That is, whether it encodes a robust-recovery strategy $\mathcal{R}_{x_d}$ for each detection state $x_d \in X_{det}$. This can be checked by verifying

**Algorithm 2:** Synthesis of maximally-permissive, non-blocking and robustly recoverable supervisors.

**input** : $G^R = (X, \Sigma, f, \Gamma, (x_0, y_0), X_m)$,
$H = (Y, \Sigma, f_H, \Gamma_H, (x_0, y_0), Y_m), \Sigma_{c,v}, \Sigma^a_{c,v}$
**output** : Supervisor $S^R$ if it exists, and empty supervisor otherwise

1 Set $H_0 \coloneqq (Y_0, \Sigma, f_{H_0}, \Gamma_{H_0}, (x_0, y_0), Y_{0,m}) = H \times G^R$;
2 Set $i = 0$;
3 **while** *True* **do**
    `// Safety part`
4  $Y_i^{\mathsf{bad}} \coloneqq \{(x,y) \in Y_i : \Gamma(x) \cap (\Sigma_{uc} \cup \Sigma^a_{c,v}) \nsubseteq \Gamma_{H_i}((x,y))\}$;
5  $Y_i^{\mathsf{safe}} = Y_i \setminus Y_i^{\mathsf{bad}}$ ;
6  Set $H_i' \coloneqq (Y_i^{\mathsf{safe}}, \Sigma, f_{H_i'} \mid Y_i^{\mathsf{safe}}, \Gamma_{H_i'}, (x_0, y_0), Y_{i,m} \cap Y_i^{\mathsf{safe}})$, where $f_{H_i'}$ is defined as

$$f_{H_i'}((x,y),e) \coloneqq \begin{cases} f_{H_i}((x,y),e) & \text{if } f_{H_i}((x,y),e)! \ \wedge [e \notin \Sigma_{c,v} \cup \Sigma^a_{c,v} \vee y \neq A] \\ \text{undefined} & \text{otherwise} \end{cases}$$

    `// Non-blockingness part`
7  $H_i' \leftarrow \mathrm{CoAc}(H_i')$ ;
8  $Y_i^{\mathsf{CoAc}} \coloneqq \{(x,y) \in Y_i^{\mathsf{safe}} \mid (x,y) \text{ is CoAc in } H_i'\}$;
9  Set $H_{i+1} \coloneqq (Y_i^{\mathsf{CoAc}}, \Sigma, f_{H_i} \mid Y_i^{\mathsf{CoAc}}, (x_0, y_0), Y_{i,m} \cap Y_i^{\mathsf{CoAc}})$;
10  $H_{i+1} \leftarrow \mathrm{Trim}(H_{i+1})$;
11  $Y_i^{\mathsf{Trim}} \coloneqq \{(x,y) \in Y_i^{\mathsf{CoAc}} \mid (x,y) \text{ is Ac in } H_{i+1}\}$;
12  **if** $Y_i^{\mathsf{Trim}} = \emptyset$ **then**
13  return $S^R \leftarrow \emptyset$;
14  **else**
15  $H_{i+1} =: (Y_{i+1}, \Sigma, f_{H_{i+1}}, (x_0, y_0), Y_{i+1,m})$;
16  **end**
17  **if** $H_{i+1} = H_i$ **then**
18  return $S^R \leftarrow H_{i+1}$;
19  **else**
20  Set $i \leftarrow i + 1$;
21  **end**
22 **end**

whether all detection states are preserved by the synthesis, as shown next.

**Theorem 3.** *Supervisor $S^R$ (synthesized by Algorithm 2) is robustly recoverable if and only if $X_{det} \subseteq X_{S^R}$.*

*Proof:* We start by proving $\Rightarrow$: If $S^R$ is robustly recoverable, then $X_{det} \subseteq X_{S^R}$. First, assume that $S^R$ is robustly recoverable, which implies that, for every detection state $x_d \in X_{det}$, there exists a robust-recovery language $K_{x_d}$ satisfying all four conditions of Definition 13. Algorithm 2 removes a state only when no such robust-recovery language can be enforced from it. Therefore, since a valid $K_{x_d}$ exists for every detection state, none of the detection states are removed during synthesis, and thus $x_d \in X_{S^R}$ for all $x_d \in X_{det}$. Hence, $X_{det} \subseteq X_{S^R}$.

Next, we prove $\Leftarrow$: If $X_{det} \subseteq X_{S^R}$, then $S^R$ is robustly recoverable. We prove this statement by contradiction. Assume $X_{det} \subseteq X_{S^R}$ and that $S^R$ is not robustly recoverable. Then, for at least one detection state $x_d \in X_{det}$, $S^R$ fails to encode a robust-recovery strategy $\mathcal{R}_{x_d}$ in the sense of Definition 15. This implies that no robust-recovery language $K_{x_d}$ satisfying Definition 13 exists. Equivalently, every sequence $s \in \overline{\mathcal{L}_{min}(G, x_d)}$ is such that, along its feasible continuations, at least one of the conditions for robust recovery languages (Definition 13) is violated. However, Algorithm 2 is designed to iteratively eliminate all such paths. Specifically, violations of Conditions 1 and 2 are addressed by removing states that are not coaccessible to $X_{G_{target}}$ via the $\mathrm{CoAc}(H_i')$ operation. Violations of Conditions 3 and 4 are addressed by identifying unsafe states and states containing uncontrollable or vulnerable deviations with no continuation to $X_{G_{target}}$ as bad states, which are subsequently removed. Since $S^R$ is not robustly recoverable by assumption, there exists at least one detection state $x_d$ from which all paths will eventually trigger one of these removal criteria during the synthesis. Consequently, $x_d$ is pruned, implying $x_d \notin X_{S^R}$, which directly contradicts the assumption that $X_{det} \subseteq X_{S^R}$. ■

**Remark 5** (Complexity analysis of Algorithm 2)**.** *Let $n = |X|$ and $m = |Y|$ denote the numbers of states of $G^R$ and $H$, respectively, and let $|\Sigma|$ denote the number of events. The synchronous product constructed in Algorithm 2 contains at most $nm$ states.*

*Each iteration of the algorithm consists of two main steps. First, the algorithm identifies states that violate the safety condition by examining uncontrollable transitions. This step requires checking the outgoing transitions of each state and therefore has complexity $O(nm|\Sigma|)$. Second, nonblockingness is enforced by computing the set of coaccessible states of the auxiliary model $H_i'$ and subsequently applying the* Trim *operation to obtain $H_{i+1}$. Both operations correspond to graph searches on an automaton with at most $nm$ states and $nm|\Sigma|$ transitions, and thus each has complexity $O(nm + nm|\Sigma|)$.*

*Consequently, each iteration has complexity $O(nm+nm|\Sigma|)$. Since at least one state is removed at each iteration, the loop executes at most $nm$ times. Therefore, the overall worst-case time complexity of Algorithm 2 is $O(nm(nm + nm|\Sigma|))$, which simplifies to $O(n^2m^2|\Sigma|)$. Hence, the proposed synthesis algorithm has the same worst-case complexity as the standard supervisory control synthesis algorithm (Cassandras; Lafortune, 2021).*

We now show that the problem of determining AE-robust recoverability and synthesizing robust-recovery strategies, i.e., Problem 1, can be reduced to the problem of synthesizing robustly recoverable supervisors.

**Theorem 4.** *There exists a robustly recoverable supervisor $S^R$ for $G^R$ if and only if the attacked system $S^a/G^a$ is* AE-robustly recoverable *w.r.t. $G_{target}$ and $\Sigma_{c,v}$.*

*Proof:* We start with the *only if* part $\Rightarrow$: If there exists a robustly recoverable supervisor $S^R$ for $G^R$, then the attacked system $S^a/G^a$ (equivalently $G^R$) is AE-robustly

recoverable w.r.t. $G_{target}$ and $\Sigma_{c,v}$. Assume $S^R$ is robustly recoverable. By Definition 17, there exists, for every detection state $x_d \in X_{det}$, a nonempty prefix-closed robust-recovery language $K_{x_d} \subseteq \overline{\mathcal{L}_{min}(G, x_d)}$ and a corresponding robust-recovery strategy $\mathcal{R}_{x_d}$ encoded by $S^R$. The existence of a $K_{x_d}$ for all $x_d \in X_{det}$ is exactly the requirement in Definition 14 for AE-robust recoverability of an attacked system $S^a/G^a$.

We now show the *if* part $\Leftarrow$: If the attacked system $S^a/G^a$ is AE-robustly recoverable, then there exists a robustly recoverable supervisor $S^R$ for $G^R$ (equivalently $S^a/G^a$). Assume $S^a/G^a$ is AE-robustly recoverable. Definition 14 guarantees that, for every detection state $x_d \in X_{det}$, there exists at least one robust-recovery language $K_{x_d}$ that leads the system to $G_{target}$. Given this property, Algorithm 1 preserves every detection state $x_d \in X_{det}$ in the synthesized supervisor $S^R$. By Theorem 1, the resulting supervisor $S^R$ is therefore robustly recoverable. ■

Consequently, a robustly recoverable supervisor $S^R$ provides a solution to both the verification and synthesis problems (Problem 1). That is, the existence of a robustly recoverable supervisor $S^R$ implies that $G^R$ is AE-robustly recoverable. Conversely, if $G^R$ is AE-robustly recoverable, then a robustly recoverable $S^R$ can be synthesized to enforce all robust-recovery languages from detection states to $G_{target}$.

**Example 8.** *Consider $G$ and $S$ as given in Example 4. Based on these models, we construct the attacked system $G^R$, and then the control specification $H$. Finally, the robustly recoverable supervisor $S^R$ for plant $G^R$ is shown in Figure 21.*

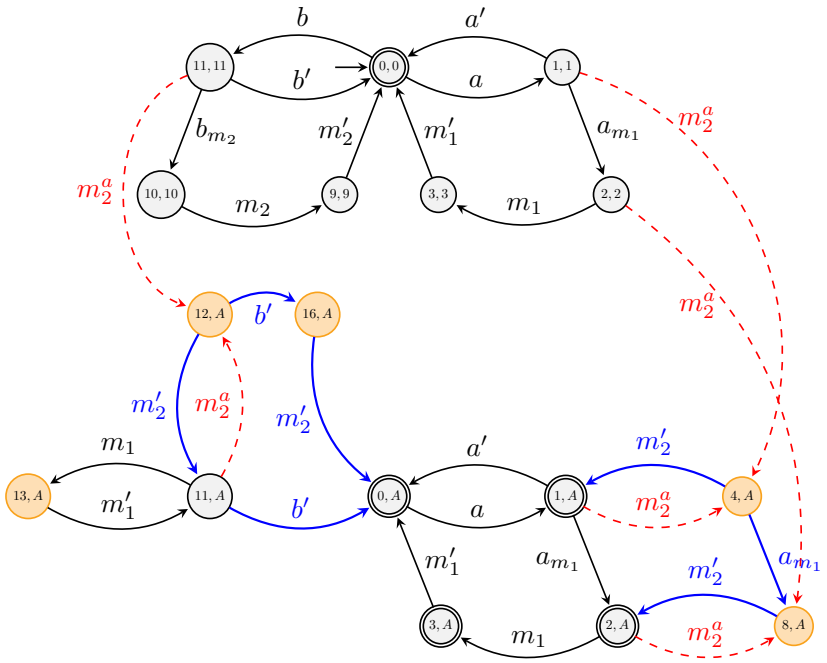



Figure 21 – Robustly recoverable supervisor $S^R$.

*Note that $S^R$ contains the desired behavior for the system both before and after AE-attacks. In detail, the transitions highlighted in red refer to the attacked events while*

*the transitions highlighted in blue refer to the recovery paths that lead to $G_{target}$.*

## 5.5 CASE STUDY: MANUFACTURING SYSTEM

In this section, we present a case study to illustrate our robust-recovery approach. We use the Fischertechnik factory simulation 24v™, a down-scale manufacturing testbed shown in Figure 22. The testbed consists of several stations, including an industrial oven, turntable, sorting station, conveyor belt, robotic gripper, and warehouse. In this case study, we model the behavior of the sorting station, conveyor belt, robotic gripper, and warehouse.

Two types of parts arrive on the conveyor belt: white or red. We assume that only one part is present in the system at any given time. Next, the conveyor moves the part to the sorting station, which has three different locations: initial, location 1, and location 2. In locations 1 and 2, pneumatic cylinders can push the parts out of the conveyor belt to buffers. Lastly, the gripper picks parts for storage. The gripper can pick parts from the initial location of the conveyor belt and the two buffers.

Figure 23 depicts the model of this system, whereas Table 2 explains each event. For simplicity, we model the gripper picking up a part and storing it with events $m$ (manual sort) and $s$ (automatic sort). For instance, if the gripper picks a part in the initial location, the part will be manually sorted, event $m$. When the gripper picks a white part in the buffer of location 1, the part is automatically stored sorted, event $s$. If the gripper picks a red part in the buffer of location 1, the part must be manually sorted before storing.

A supervisor is designed to automatically store the parts sorted by color. The supervisor directs white parts to location 1, while red parts go to location 2. In Figure 23, states in red are unsafe states. They represent that a part was wrongly stored as sorted, e.g., a red part picked in location 2 and stored as sorted. States in yellow are also removed by the supervisor since they are undesired states, e.g., manually sorting parts. The supervisor is defined as the subautomaton of the plant model obtained by removing all red and yellow states.

### 5.5.1 Actuator attacks

We assume three actuator attack scenarios for the pneumatic cylinders: cylinder $p_1$ under attack $\Sigma^1_{c,v} = \{p_1\}$, cylinder $p_2$ under attack $\Sigma^2_{c,v} = \{p_2\}$, and cylinders $p_1$ and $p_2$ under attacks $\Sigma^3_{c,v} = \{p_1, p_2\}$. For each of these three scenarios, we construct the attacked plants $G^a_1$, $G^a_2$, and $G^a_3$ and the attacked supervisors $S^a_1$, $S^a_2$, and $S^a_3$, respectively. We verify that in all cases, the controlled systems are AE-safe controllable. Intuitively, the supervisor can disable event $s$ when a part is moved to an incorrect position, for example, when an attacker enables event $p_1$ and a red part is pushed into the buffer of location 1. Next, we investigate robust-recovery strategies for these three scenarios.

Figure 22 – Fischertechnik manufacturing testbed.

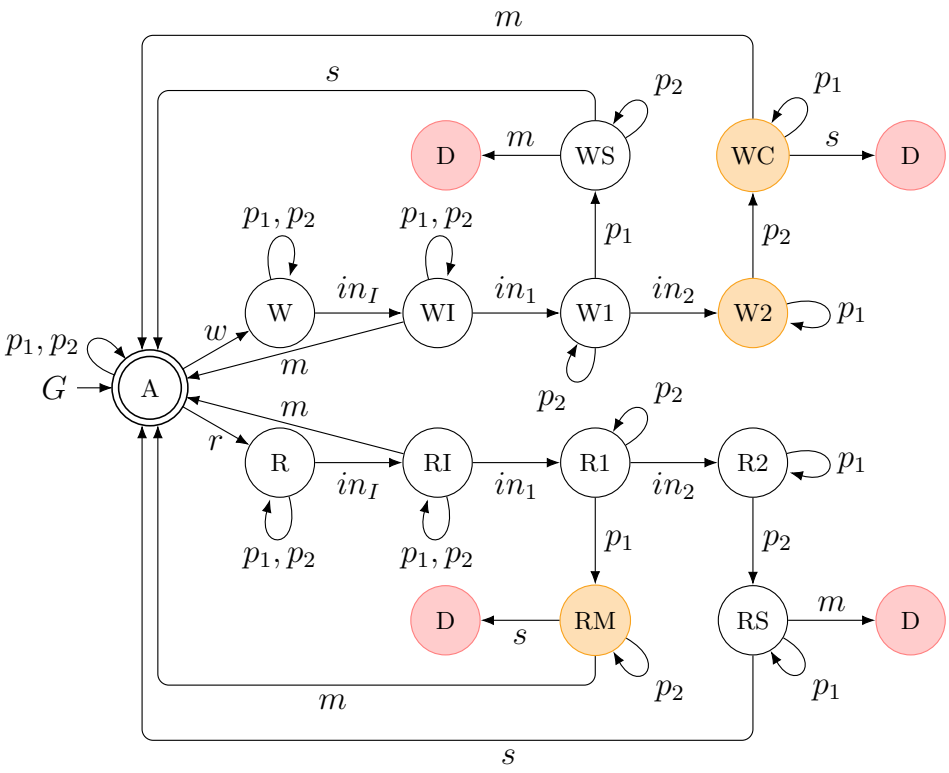



Figure 23 – Model of plant $G$.

### 5.5.2 Robust recovery strategies

To analyze robust-recovery strategies in our case study, we first define the target region $G_{target}$. This region is defined by states $A, W, WI, R$, and $RI$. Upon attack detection, the controlled system should transition to manual sorting to maintain operational continuity. Specifically, when a part reaches the initial position, only event $m$ should be allowed. By applying such a restriction, further vulnerabilities are avoided.

In the first scenario ($\Sigma^1_{c,v} = \{p_1\}$), attacks are only feasible when red parts are being processed. Specifically, after a red part reaches position 1 (event $in_1$), an attacker

Table 2 – Event descriptions for the manufacturing testbed.

| Event | Description |
|---|---|
| $w$ | Arrival of a white part on the conveyor |
| $r$ | Arrival of a red part on the conveyor |
| $in_I$ | Command for the conveyor to move the part to the initial position |
| $in_1$ | Command for the conveyor to move the part to position 1 |
| $in_2$ | Command for the conveyor to move the part to position 2 |
| $p_1$ | Command for pusher 1 to push a part into buffer 1 |
| $p_2$ | Command for pusher 2 to push a part into buffer 2 |
| $s$ | Command to sort and store the part automatically |
| $m$ | Command to route the part for manual sorting |

can enable event $p_1$ when it should be disabled. Following this attack, the system reaches the detection state $(RM, A)$, from which $G_{target}$ can be reached through event $m$, allowing the part to be manually sorted.

If the attack on $p_1$ occurs after a part reaches position 2 (event $in_2$), the system reaches the detection state $(R2, A)$. From there, $G_{target}$ can be reached through the sequence $p_2 s$, enabling recovery. The resilient supervisor $S^{R,1}$ is shown in Figure 24. The opaque red transitions represent attacks that lead directly to $G_{target}$ states, and thus no recovery is needed. The red dashed transitions (non-opaque) denote attacks that are followed by a robust-recovery strategy leading to $G_{target}$[2].

In the second scenario ($\Sigma^2_{c,v} = \{p_2\}$), a similar recovery strategy exists. For instance, when $p_2$ is attacked after a white part reaches position 1, recovery is achieved through the sequence $p_1 s$. When $p_2$ is attacked after a red part reaches position 1, the system recovers through the sequence $p_1 m$.

The third scenario ($\Sigma^3_{c,v} = \{p_1, p_2\}$) presents a critical vulnerability. When both cylinders are vulnerable to attacks, the system $G^{R,3}$ is not AE-robustly recoverable. For instance, when a white part reaches position 1 and event $p_2$ is attacked, the only potential recovery path to $G_{target}$ requires using event $p_1$, which is also vulnerable to attacks. At this point, an AD-attack could prevent recovery. With both recovery paths compromised, $S^{R,3}$ cannot enforce a robust-recovery strategy upon attack detection.

Overall, this case study illustrates both the applicability and the limitations of the proposed framework. When at least one alternative non-vulnerable event remains available, the synthesized supervisor successfully enforces recovery to $G_{target}$ and sustains a post-attack operation despite persistent attacks. Conversely, when all candidate recovery continuations necessarily depend on vulnerable events, as in the scenario $\Sigma^3_{c,v} = p_1, p_2$, the property of robust recoverability is violated. That is because an attacker can induce blocking through AD-attacks by disabling the very events required to complete recovery.

[2] A video demonstrating the behavior of the closed-loop system $S^{R,1}/G^{R,1}$ is available in https://youtu.be/Azlbpt9woHA

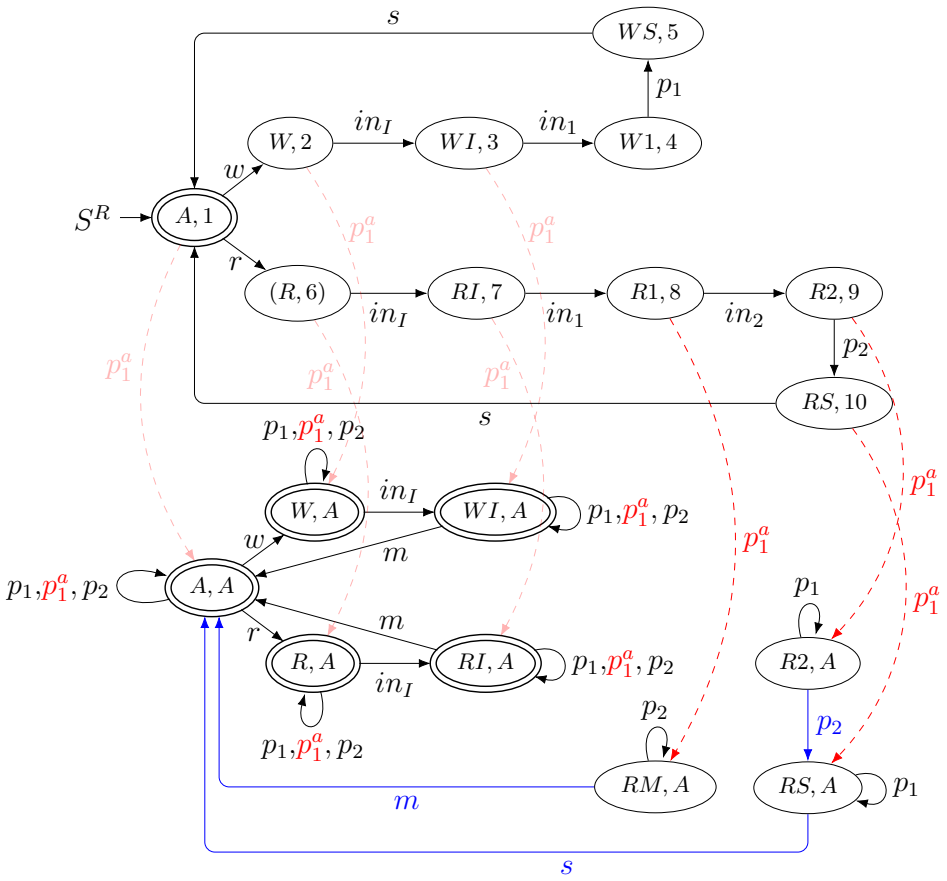



Figure 24 – Robustly recoverable supervisor $S^R$ for the attack scenario where $\Sigma^1_{c,v} = \{p_1\}$.

The developments in this chapter build upon our preliminary results presented at the 64th IEEE Conference on Decision and Control (Oliveira et al., 2025), and substantially extend them by relaxing the requirement of an attack-free target regions, by incorporating refined robustness conditions related to actuator disablement during the recovery phase, and by formalizing recoverability at the language level together with a synthesis procedure for robustly recoverable supervisors.

# 6 CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK

This doctoral thesis addresses cybersecurity problems related to protection and recovery strategies in the context of Cyber-Physical Discrete Event Systems (CPDES) under active attacks. After conducting a systematic literature review, we identified significant research gaps, specifically concerning the protection of CPDES against covert attacks and the recovery of attacked systems.

## 6.1 ACHIEVEMENT OF THE RESEARCH OBJECTIVES

The objectives defined in Section 1.4 have been achieved throughout this thesis.

First, cybersecurity techniques for integrity preservation and robust recovery in CPDES were developed. Chapter 4 introduced an actuator edit-function framework that prevents unsafe controllable events even under covert sensor and actuator attacks. Chapter 5 proposed a robust recovery framework capable of driving attacked systems to a target region while remaining resilient to persistent attacks.

Second, the thesis established formal conditions associated with these techniques. In particular, the property of *live protectability* was defined to characterize when CPDES can be protected against covert attacks using actuator edit functions, while Chapter 5 introduced conditions for *robust recoverability*.

Third, synthesis algorithms were developed to compute the proposed mechanisms. Chapter 4 presented an algorithm for synthesizing global live-protecting edit functions, while Chapter 5 proposed a supervisor synthesis procedure for robust recovery.

Finally, the applicability of the proposed frameworks was demonstrated through illustrative examples and case studies, including the water tank example and a manufacturing system based on the Fischertechnik Factory Simulation 24V.

The results presented in this thesis can be summarized through two main contributions: (i) protection of CPDES against covert attacks with integrity and liveness guarantees; and (ii) robust recovery of CPDES under persistent attacks.

Regarding the first contribution, our framework focuses on systems where the property of safe controllability does not hold (Carvalho et al., 2018). These are systems in which traditional supervisory control cannot prevent unsafe states from being reached once an attack is detected. To address this issue, we consider covert sensor and actuator attacks and propose an actuator edit function coupled to the supervisory control layer. This approach is analogous to strategies that enforce opacity through edit functions; however, instead of hiding secrets, our actuator edit function modifies control commands received by the plant whenever such commands would lead the system to an unsafe state.

Note that while this protection mechanism ensures system safety, attack detection is not guaranteed in this framework. Although the supervisor observes the plant behavior, a modification by the edit function may lead to the execution of an event that the supervisor

already expects, thereby masking the attack. Addressing attack detection in this framework is a direction for future work.

The property of live protectability is formalized to characterize the necessary and sufficient conditions under which a system can be protected against covert attacks while guaranteeing liveness. We also propose a synthesis algorithm for the actuator edit function and demonstrate that a solution to this synthesis problem effectively solves the corresponding verification problem for live protectability.

Liveness is addressed in this framework because it is a crucial requirement for recovery. Beyond preventing unsafe states, it is essential that attacked systems remain operational. If a system avoids unsafe states but becomes blocked, integrity is enforced at the cost of availability.

As a subsequent step in this doctoral research, we developed a robust recovery framework. The term "robust" reflects the fact that recovery is formalized to ensure the system can be driven to a target region, which is defined as a subset of states defined for post-attack operation, even in the presence of persistent actuator enablement and disablement attacks. The property of robust recoverability is formalized to provide necessary and sufficient conditions for a system to be robustly recoverable. Furthermore, we proposed a synthesis algorithm to compute maximally permissive, nonblocking, and robustly recoverable supervisors.

As directions for future work, we consider extending the protection framework to account for attack detection alongside integrity and liveness preservation. Regarding the robust recovery framework, future research will focus on systems subject to both sensor and actuator attacks. By doing so, it will be possible to develop a unified framework that integrates protection against covert attacks with robust recovery mechanisms.

Beyond these direct extensions, an additional research direction concerns the investigation of artificial intelligence–based techniques for enforcing safety and security properties in CPDES. While recent works have explored the use of learning-based methods for attack detection (Amri et al., 2025), their potential to enforce integrity, liveness, or recovery properties remains largely unexplored. Integrating learning-based components with supervisory control frameworks represents a promising avenue for future research.

## 6.2 PUBLICATIONS

During the course of this doctoral research, 11 scientific papers were authored/coauthored. Among them, 8 are directly related to the proposals developed in this thesis, while 3 address related topics that emerged from collaborative research activities.

These papers were produced in collaboration with the *Systems Automation and Robotics Research Group (GASR)* at Santa Catarina State University (UDESC), as well as with the *Control and Automation for Intelligent Systems (CAIS)* group and the *Controlling*

*Autonomous Systems with Assurances (Casa Góes Lab)*, both at The Pennsylvania State University (PSU). Participation in these research groups contributed to the development of the theoretical foundations, modeling approaches, and applications explored throughout this thesis, while also enabling research on complementary topics beyond its main scope.

### 6.2.1 Papers in the Context of this Thesis

- OLIVEIRA, Samuel; LEAL, André B; TEIXEIRA, Marcelo; LOPES, Yuri K. A classification of cybersecurity strategies in the context of Discrete Event Systems. In: **Annual Reviews in Control**. Volume 56, Elsevier, 2023.
- OLIVEIRA, Samuel; LEAL, André B; TEIXEIRA, Marcelo; LOPES, Yuri K. Security of Cyber-Physical Systems Against Actuator Attacks through Cryptography. In: **2023 International Conference on Information Technology (ICIT)**. p. 758-764, Amman, Jordan:IEEE, 2023.
- OLIVEIRA, Samuel; LEAL, André B; TEIXEIRA, Marcelo; LOPES, Yuri K. Segurança de sistemas ciberfísicos contra ataques a atuadores: um método baseado em criptografia. In: **Simpósio Brasileiro de Automação Inteligente (SBAI 2023)**, Manaus, Brasil, 2023.
- OLIVEIRA, Samuel; LEAL, André B; TEIXEIRA, Marcelo; LOPES, Yuri K. Integrity of Cyber-Physical Discrete Event Systems under Covert Actuator Attacks. In: **2024 17th Workshop on Discrete Event Systems (WODES'24)**. Rio de Janeiro, Brazil:IFAC, 2024.
- OLIVEIRA, Samuel; LEAL, André B; TEIXEIRA, Marcelo; LOPES, Yuri K.; MEIRA-GOES, Rômulo. Uma abordagem baseada em substituição de eventos para mitigar ataques a atuadores em Sistemas a Eventos Discretos. In: **Congresso Brasileiro de Automática (CBA 2024)**, Rio de Janeiro, Brasil, 2024.
- OLIVEIRA, Samuel; ANBARANI, Mostafa Tavakkoli; BEAL, Gregory; KOVALENKO, Ilya; TEIXEIRA, Marcelo; LEAL, André B; MEIRA-GÓES, Rômulo. Robust Recovery and Control of Cyber-physical Discrete Event Systems under Actuator Attacks. In: **IEEE Conference on Decision and Control (CDC 2025)**, Rio de Janeiro, Brazil:IEEE, 2025.
- OLIVEIRA, Samuel; ANBARANI, Mostafa Tavakkoli; BEAL, Gregory; KOVALENKO, Ilya; TEIXEIRA, Marcelo; LEAL, André B; MEIRA-GÓES, Rômulo. Robust Recovery from Actuator Attacks in Cyber-physical Discrete Event Systems. (*To be submitted*).

- OLIVEIRA, Samuel; TEIXEIRA, Marcelo; LEAL, André B; MEIRA-GÓES, Rômulo. Protecting Cyber-Physical Discrete Event Systems from Covert Active Attacks via Edit Functions. (*To be submitted*).

### 6.2.2 Papers in related topics

- MOTA, Bruno; OLIVEIRA, Samuel; LEAL, André B. Controle Supervisório Tolerante a Falhas de Sistemas a Eventos Discretos: Estudo de Caso. In: **Simpósio Brasileiro de Automação Inteligente (SBAI 2023)**, Manaus, Brasil, 2023.

- FELLER, Paulo; LEAL, André B.; OLIVEIRA, Samuel. Implementação da Estrutura de Controle Supervisório em Blocos de Função da IEC 61499: um Estudo de Caso. In: **Congresso Brasileiro de Automática (CBA 2024)**, Rio de Janeiro, Brasil, 2024.

- FAHIM, Parastou; OLIVEIRA, Samuel; MEIRA-GÓES, Rômulo. Enhancing sensor attack detection in supervisory control systems modeled by probabilistic automata. (*Under review*).

## REFERENCES

ALVES, Lucas V. R.; PENA, Patrícia N. Secure recovery procedure for manufacturing systems using synchronizing automata and supervisory control theory. **IEEE Transactions on Automation Science and Engineering**, v. 19, n. 1, p. 486–496, 2022. Cited 6 times on pages 20, 21, 39, 45, 59, and 63.

ALVES, Michel R.C.; PENA, Patrícia N.; RUDIE, Karen. Discrete-event systems subject to unknown sensor attacks. **Discrete Event Dynamic Systems: Theory and Applications**, v. 32, n. 1, p. 143–158, 2022. Cited on page 20.

ALVES, Michel R.C.; RUDIE, Karen; PENA, Patrícia N. A security testbed for networked des control systems. **IFAC-PapersOnLine**, v. 55, n. 28, p. 128–134, 2022. ISSN 2405-8963. Cited on page 44.

ALVES, Michel R.C.; RUDIE, Karen; PENA, Patrícia N. Persistent attacks on the supervisory control layer of des. **IFAC-PapersOnLine**, v. 58, n. 1, p. 13–18, 2024. ISSN 2405-8963. 17th IFAC Workshop on discrete Event Systems WODES 2024. Cited on page 20.

ALVES, Marcos V.S. et al. Robust decentralized diagnosability of networked discrete event systems against dos and deception attacks. **Nonlinear Analysis: Hybrid Systems**, v. 44, 2022. Cited on page 41.

AMRI, Omar et al. Cyber-attacks detection in timed probabilistic dess via artificial neural networks. In: **2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC)**. Vienna, Austria: IEEE, 2025. p. 3548–3553. Cited 2 times on pages 39 and 86.

BAIER, Christel; KATOEN, Joost-Pieter. **Principles of model checking**. London, England: MIT Press, 2008. Cited on page 68.

BALUN, Jiří; MASOPUST, Tomáš. Comparing the notions of opacity for discrete-event systems. **Discrete Event Dynamic Systems**, Springer Science and Business Media LLC, v. 31, n. 4, p. 553–582, jul. 2021. ISSN 1573-7594. Cited on page 28.

BARCELOS, Raphael J.; BASILIO, João C. Ensuring utility while enforcing current-state opacity. **IFAC-PapersOnLine**, v. 56, n. 2, p. 4595–4600, 2023. ISSN 2405-8963. 22nd IFAC World Congress. Cited 2 times on pages 33 and 34.

BASILE, Francesco; TOMMASI, Gianmaria De. Assessment of multilevel intransitive non-interference for discrete event systems. **IEEE Control Systems Letters**, v. 6, p. 349–354, 2022. Cited on page 33.

BASILE, Francesco; TOMMASI, Gianmaria De; STERLE, Claudio. Noninterference enforcement via supervisory control in bounded petri nets. **IEEE Transactions on Automatic Control**, v. 66, n. 8, p. 3653 – 3666, 2021. ISSN 00189286. Cited on page 33.

BASILIO, João Carlos; HADJICOSTIS, Christoforos N.; SU, Rong. Analysis and control for resilience of discrete event systems: Fault diagnosis, opacity and cyber security. **Foundations and Trends in Systems and Control**, v. 8, n. 4, p. 285–443, 2021. Cited 2 times on pages 30 and 31.

CAO, Liwei et al. A survey of network attacks on cyber-physical systems. **IEEE Access**, v. 8, p. 44219–44227, 2020. Cited on page 20.

CARDOSO, João M.C.; MOREIRA, Marcos V.; CARVALHO, Lilian K. Synthesis of an obfuscation policy that guarantees utility satisfying a new privacy criterion. **IFAC-PapersOnLine**, v. 56, n. 2, p. 11344–11349, 2023. ISSN 2405-8963. 22nd IFAC World Congress. Cited 2 times on pages 33 and 34.

CARVALHO, Lilian Kawakami et al. Detection and prevention of actuator enablement attacks in supervisory control systems. In: **2016 13th International Workshop on Discrete Event Systems (WODES)**. Xi'an, China: IEEE, 2016. p. 298–305. Cited 3 times on pages 30, 38, and 42.

CARVALHO, Lilian Kawakami et al. Detection and mitigation of classes of attacks in supervisory control systems. **Automatica**, v. 97, p. 121–133, 2018. ISSN 0005-1098. Cited 11 times on pages 20, 31, 38, 42, 48, 52, 60, 62, 65, 67, and 85.

CASSANDRAS, Christos G.; LAFORTUNE, Stéphane. **Introduction to Discrete Event Systems**. 3. ed. [S.l.]: Springer Cham, 2021. Cited 4 times on pages 24, 25, 71, and 79.

CAVALCANTI, Dayse M. et al. Recovery of discrete event systems after active cyberattacks. **IEEE Control Systems Letters**, p. 1–1, 2025. Cited 7 times on pages 20, 21, 39, 43, 45, 59, and 63.

CHEN, Qinrui; SU, Rong; LI, Zhiwu. Attackable detectability of partially-observed discrete-event systems under sensor attack. **IFAC-PapersOnLine**, v. 55, n. 28, p. 121–127, 2022. ISSN 2405-8963. Cited on page 42.

CHEN, Qinrui; SU, Rong; LI, Zhiwu. Synthesis of sensor attacks for tampering detectability of partially observed discrete-event systems. **IEEE Transactions on Automatic Control**, p. 1–15, 2025. Cited 2 times on pages 20 and 42.

CHO, Hangju; MARCUS, Steven I. On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation. **Mathematics of Control, Signals and Systems**, v. 2, n. 1, p. 47–69, 1989. Cited on page 25.

CUI, Bohan; GIUA, Alessandro; YIN, Xiang. Attack-resilient supervisory control of discrete event systems under dynamic-event-protection mechanisms. **IEEE Control Systems Letters**, v. 9, p. 1189–1194, 2025. ISSN 2475-1456. Cited on page 37.

DELAVAL, Gwenael et al. Discrete control of response for cybersecurity in industrial control. **IFAC-PapersOnLine**, v. 53, n. 2, p. 1747–1754, 2020. ISSN 2405-8963. Cited on page 39.

DIBAJI, Seyed M. et al. A systems and control perspective of cps security. **Annual Reviews in Control**, v. 47, p. 394–411, 2019. ISSN 1367-5788. Cited on page 19.

DUBREIL, Jeremy; DARONDEAU, Philippe; MARCHAND, Herve. Opacity enforcing control synthesis. In: **2008 9th International Workshop on Discrete Event Systems**. Gothenburg, Sweden: IEEE, 2008. p. 28–35. Cited on page 33.

EL-MAHDY, Mostafa H.; MAGED, Shady A.; AWAD, Mohammed I. End-to-end fault tolerant control of discrete event system using recurrent neural networks. In: **2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)**. Cairo, Egypt: IEEE, 2022. p. 266–271. Cited on page 43.

ELLIOT, Jim. **The Journals of Jim Elliot**. [S.l.]: F. H. Revell Company, 1978. ISBN 9780800709228. Cited on page 7.

FAHIM, Parastou; MEIRA-GOES, Rômulo. Detecting probability footprints of sensor deception attacks in supervisory control. **IFAC-PapersOnLine**, v. 58, n. 1, p. 192–197, 2024. ISSN 2405-8963. 17th IFAC Workshop on discrete Event Systems WODES 2024. Cited on page 20.

FAHIM, Parastou; OLIVEIRA, Samuel; MEIRA-GÓES, Rômulo. Enhancing sensor attack detection in supervisory control systems modeled by probabilistic automata. **arXiv preprint arXiv:2502.16753**, 2025. Cited on page 20.

FALCONE, Yliès; MARCHAND, Hervé. Enforcement and validation (at runtime) of various notions of opacity. **Discrete Event Dynamic Systems**, Springer Science and Business Media LLC, v. 25, n. 4, p. 531–570, jul. 2014. Cited on page 43.

FRITZ, Raphael; FAUSER, Moritz; ZHANG, Ping. Controller encryption for discrete event systems. In: **2019 American Control Conference (ACC)**. Philadelphia, PA, USA: IEEE, 2019. p. 5633–5638. Cited 2 times on pages 28 and 33.

FRITZ, Raphael; ZHANG, Ping. Detection and localization of stealthy cyber attacks in cyber-physical discrete event systems. **IEEE Transactions on Automatic Control**, p. 1–8, 2023. Cited 3 times on pages 18, 20, and 29.

GUO, Ye et al. Overview of opacity in discrete event systems. **IEEE Access**, v. 8, p. 48731–48741, 2020. Cited on page 28.

HABBACHI, Salwa et al. Secret inference and attacktability analysis of discrete event systems. **Information Sciences**, v. 609, p. 1221–1238, 2022. Cited 2 times on pages 35 and 43.

HADJ-ALOUANE, Nejib B. et al. Characterizing intransitive noninterference for 3-domain security policies with observability. **IEEE Transactions on Automatic Control**, v. 50, n. 6, p. 920–925, 2005. Cited 2 times on pages 28 and 42.

HADJ-ALOUANE, Nejib B. et al. On the verification of intransitive noninterference in mulitlevel security. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 35, n. 5, p. 948–958, 2005. Cited 2 times on pages 28 and 42.

HADJICOSTIS, Christoforos N. et al. Cybersecurity and supervisory control: A tutorial on robust state estimation, attack synthesis, and resilient control. In: **2022 IEEE 61st Conference on Decision and Control (CDC)**. Cancún, Mexico: IEEE, 2022. p. 3020–3040. Cited 2 times on pages 20 and 42.

HE, Zhou; MA, Ziyue; TANG, Wei. Performance safety enforcement in strongly connected timed event graphs. **Automatica**, v. 128, 2021. Cited on page 39.

HE, Zhaoyang; WU, Naiqi; LI, Zhiwu. Estimation and prevention of actuator enablement attacks in discrete-event systems under supervisory control. **IEEE Transactions on Automatic Control**, v. 69, n. 9, p. 5963–5978, Sep. 2024. ISSN 1558-2523. Cited on page 39.

HE, Zhaoyang et al. Cyber-attacks with resource constraints on discrete event systems under supervisory control. **IEEE/CAA Journal of Automatica Sinica**, v. 12, n. 3, p. 585–595, March 2025. ISSN 2329-9274. Cited on page 36.

HU, Shaopeng et al. Supervisory control for active diagnosis in labeled petri nets under coordinated sensor and actuator attacks. **IEEE Transactions on Automatic Control**, p. 1–14, 2025. ISSN 1558-2523. Cited on page 41.

HU, Shaopeng; LI, Zhiwu; LIU, Ding. Diagnosability verification and enforcement in labeled petri nets under sensor attacks. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 55, n. 5, p. 3654–3667, May 2025. ISSN 2168-2232. Cited on page 41.

JACOB, Romain; LESAGE, Jean-Jacques; FAURE, Jean-Marc. Overview of discrete event systems opacity: Models, validation, and quantification. **Annual Reviews in Control**, v. 41, p. 135–146, 2016. ISSN 1367-5788. Cited on page 28.

JI, Yiding; YIN, Xiang; LAFORTUNE, Stéphane. Enforcing opacity by insertion functions under multiple energy constraints. **Automatica**, v. 108, 2019. Cited on page 43.

KANG, Tenglong; LIU, Ding; LI, Zhiwu. Robust prognosability of discrete event systems against stealthy sensor attacks. **Automatica**, v. 185, p. 112740, 2026. ISSN 0005-1098. Cited 2 times on pages 41 and 43.

KANG, Tenglong et al. Fault diagnosis of discrete event systems under attack. In: **2023 62nd IEEE Conference on Decision and Control (CDC)**. Singapore: IEEE, 2023. p. 7923–7929. ISSN 2576-2370. Cited on page 40.

KHOUMSI, Ahmed. Sensor and actuator attacks of cyber-physical systems: A study based on supervisory control of discrete event systems. In: **2019 8th International Conference on Systems and Control (ICSC)**. Marrakesh, Morocco: IEEE, 2019. p. 176–182. Cited on page 35.

LABED, Abdeldjalil et al. Language recovery in discrete-event systems against sensor deception attacks. **Mathematics**, v. 11, n. 10, 2023. Cited on page 30.

LAFORTUNE, Stéphane; LIN, Feng; HADJICOSTIS, Christoforos N. On the history of diagnosability and opacity in discrete event systems. **Annual Reviews in Control**, v. 45, p. 257–266, 2018. ISSN 1367-5788. Cited on page 28.

LEITÃO, Herbert A. S. et al. Fault Handling in Discrete Event Systems Applied to IEC 61499. In: **IEEE Int. Conference on Emerging Technologies and Factory Automation (ETFA)**. Vienna, Austria: IEEE, 2020. v. 1, p. 1039–1042. Cited on page 43.

LI, Yuting; HADJICOSTIS, Christoforos N.; WU, Naiqi. Error- and tamper-tolerant decentralized diagnosability of discrete event systems under cost constraints. In: **2021**

**European Control Conference, ECC 2021**. Delft, Netherlands: IEEE, 2021. p. 42 – 47. Cited on page 41.

LI, Yuting; HADJICOSTIS, Christoforos N.; WU, Naiqi. Tamper-tolerant diagnosability under bounded or unbounded attacks. **IFAC-PapersOnLine**, v. 55, n. 28, p. 52–57, 2022. ISSN 2405-8963. 16th IFAC Workshop on Discrete Event Systems WODES 2022. Cited on page 40.

LI, Yike; TONG, Yin; GIUA, Alessandro. Detection and prevention of cyber-attacks in networked control systems. In: **15th IFAC Workshop on Discrete Event Systems (WODES)**. Rio de Janeiro, Brazil: IFAC, 2020. v. 53, n. 4, p. 7–13. Cited 2 times on pages 30 and 38.

LIMA, Públio M. **Security of Cyber-Physical Systems Against Active and Passive Network Attacks: A Discrete Event System Approach**. Tese (PhD thesis) — COPPE - UFRJ, Rio de Janeiro, RJ, August 2021. Cited 2 times on pages 28 and 33.

LIMA, Públio M. et al. Security against communication network attacks of cyber-physical systems. **Journal of Control, Automation and Electrical Systems**, v. 30, n. 1, p. 125–135, 2019. Cited 2 times on pages 38 and 62.

LIMA, Públio M. et al. Security against network attacks in supervisory control systems. **IFAC-PapersOnLine**, v. 50, n. 1, p. 12333–12338, 2017. ISSN 2405-8963. Cited 3 times on pages 31, 38, and 42.

LIMA, Públio M. et al. Security of cyber-physical systems: Design of a security supervisor to thwart attacks. **IEEE Transactions on Automation Science and Engineering**, v. 19, n. 3, p. 2030–2041, 2022. Cited 4 times on pages 20, 31, 38, and 48.

LIMA, Públio M.; CARVALHO, Lilian K.; MOREIRA, Marcos V. Detectable and undetectable network attack security of cyber-physical systems. **IFAC-PapersOnLine**, v. 51, n. 7, p. 179–185, 2018. ISSN 2405-8963. Cited on page 38.

LIMA, Públio M.; CARVALHO, Lilian K.; MOREIRA, Marcos V. Confidentiality of cyber-physical systems using event-based cryptography. **IFAC-PapersOnLine**, v. 53, n. 2, p. 1735–1740, 2020. ISSN 2405-8963. Cited 2 times on pages 28 and 33.

LIMA, Públio M.; CARVALHO, Lilian K.; MOREIRA, Marcos V. Ensuring confidentiality of cyber-physical systems using event-based cryptography. **Information Sciences**, v. 621, p. 119 – 135, 2023. ISSN 00200255. Cited 2 times on pages 28 and 33.

LIMA, Públio M. et al. Event-based cryptography for automation networks of cyber-physical systems using the stream cipher chacha20. **IFAC-PapersOnLine**, v. 55, n. 28, p. 58–65, 2022. ISSN 2405-8963. Cited 2 times on pages 28 and 33.

LIN, Feng. Opacity of discrete event systems and its applications. **Automatica**, v. 47, n. 3, p. 496–503, 2011. ISSN 0005-1098. Cited on page 33.

LIN, Feng; LAFORTUNE, Stéphane; WANG, Caisheng. Diagnosability of discrete event systems under sensor attacks*. **IFAC-PapersOnLine**, Elsevier BV, v. 56, n. 2, p. 3572–3578, 2023. ISSN 2405-8963. Cited on page 40.

LIN, Feng; LAFORTUNE, Stéphane; WANG, Caisheng. Diagnosability and attack detection for discrete event systems under sensor attacks. **Discrete Event Dynamic Systems**, Springer Science and Business Media LLC, v. 34, n. 3, p. 465–495, jul. 2024. ISSN 1573-7594. Cited on page 40.

LIN, Liyong; SU, Rong. Synthesis of covert actuator and sensor attackers. **Automatica**, v. 130, 2021. Cited 2 times on pages 31 and 46.

LIN, Liyong et al. Observation-assisted heuristic synthesis of covert attackers against unknown supervisors. **Discrete Event Dynamic Systems: Theory and Applications**, v. 32, n. 3, p. 495–520, 2022. Cited 2 times on pages 35 and 44.

LIN, Liyong et al. Synthesis of supremal successful normal actuator attackers on normal supervisors. In: **2019 American Control Conference (ACC)**. Philadelphia, PA, USA: IEEE, 2019. p. 5614–5619. Cited 2 times on pages 35 and 37.

LIN, Liyong; ZHU, Yuting; SU, Rong. Towards bounded synthesis of resilient supervisors. In: **2019 IEEE 58th Conference on Decision and Control (CDC)**. [S.l.: s.n.], 2019. p. 7659–7664. Cited on page 38.

LIN, Liyong; ZHU, Yuting; SU, Rong. Synthesis of covert actuator attackers for free. **Discrete Event Dynamic Systems: Theory and Applications**, v. 30, n. 4, p. 561 – 577, 2020. ISSN 09246703. Cited 2 times on pages 20 and 35.

MA, Ziyue; CAI, Kai. On resilient supervisory control against indefinite actuator attacks in discrete-event systems. **IEEE Control Systems Letters**, v. 6, p. 2942–2947, 2022. Cited 7 times on pages 20, 21, 30, 37, 43, 44, and 63.

MA, Ziyue; CAI, Kai. Optimal secret protections in discrete-event systems. **IEEE Transactions on Automatic Control**, v. 67, n. 6, p. 2816–2828, JUN 2022. ISSN 0018-9286. Cited 2 times on pages 28 and 33.

MA, Ziyue; GIUA, Alessandro; SEATZU, Carla. Actuator-enabling attacks in discrete-event systems with unknown supervisors. In: **2025 American Control Conference (ACC)**. Denver, CO, USA: IEEE, 2025. p. 5210–5215. Cited 2 times on pages 20 and 35.

MA, Ziyue; GIUA, Alessandro; SEATZU, Carla. Weakly harmful actuator-enabling attacks in discrete-event systems with unknown supervisors. In: **2025 IEEE 64th Conference on Decision and Control (CDC)**. Rio de Janeiro, Brazil: IEEE, 2025. p. 287–292. ISSN 2576-2370. Cited on page 35.

MAINHARDT, Ana Maria et al. Formulating attacks with supervisory control. **IFAC-PapersOnLine**, v. 58, n. 1, p. 1–6, 2024. ISSN 2405-8963. 17th IFAC Workshop on discrete Event Systems WODES 2024. Cited on page 20.

MATSUI, Shoma; CAI, Kai. Secret securing with multiple protections and minimum costs. In: **2019 IEEE 58th Conference on Decision and Control (CDC)**. Nice, France: IEEE, 2019. p. 7635–7640. Cited 2 times on pages 28 and 33.

MAYER, Patrícia C. et al. Property-based transparency: a new utility definition*. In: **2024 IEEE 20th International Conference on Automation Science and**

**Engineering (CASE)**. Bari, Italy: IEEE, 2024. p. 2825–2831. ISSN 2161-8089. Cited 2 times on pages 33 and 34.

MEIRA-GÓES, Rômulo et al. Stealthy deception attacks for cyber-physical systems. In: **2017 IEEE 56th Annual Conference on Decision and Control (CDC)**. Melbourne, Australia: IEEE, 2017. p. 4224–4230. Cited 3 times on pages 29, 35, and 42.

MEIRA-GÓES, Rômulo et al. Synthesis of sensor deception attacks at the supervisory layer of cyber–physical systems. **Automatica**, v. 121, p. 109172, 2020. ISSN 0005-1098. Cited 3 times on pages 20, 30, and 35.

MEIRA-GÓES, Rômulo; KWONG, Raymond; LAFORTUNE, Stéphane. Synthesis of sensor deception attacks for systems modeled as probabilistic automata. In: **2019 American Control Conference (ACC)**. Philadelphia, PA, USA: IEEE, 2019. p. 5620–5626. Cited on page 35.

MEIRA-GÓES, Rômulo; KWONG, Raymond H.; LAFORTUNE, Stéphane. Synthesis of optimal multiobjective attack strategies for controlled systems modeled by probabilistic automata. **IEEE Transactions on Automatic Control**, v. 67, n. 6, p. 2873–2888, 2022. Cited on page 35.

MEIRA-GÓES, Rômulo; LAFORTUNE, Stéphane; MARCHAND, Hervé. Synthesis of supervisors robust against sensor deception attacks. **IEEE Transactions on Automatic Control**, v. 66, n. 10, p. 4990–4997, 2021. Cited 2 times on pages 20 and 36.

MEIRA-GÓES, Rômulo; MARCHAND, Hervé; LAFORTUNE, Stéphane. Dealing with sensor and actuator deception attacks in supervisory control. **Automatica**, v. 147, 2023. Cited 8 times on pages 20, 21, 29, 30, 38, 48, 63, and 67.

MOOR, Thomas. A discussion of fault-tolerant supervisory control in terms of formal languages. **Annual Reviews in Control**, v. 41, p. 159–169, 2016. ISSN 1367-5788. Cited on page 20.

MOREIRA, Benjamin G.; LEAL, André B. A proposal for an active diagnoser for safe fault-tolerant control of discrete event systems. **IFAC-PapersOnLine**, v. 53, n. 4, p. 282–287, 2020. ISSN 2405-8963. 15th IFAC Workshop on Discrete Event Systems WODES 2020 — Rio de Janeiro, Brazil, 11-13 November 2020. Cited on page 20.

MOREIRA, Marcos V. et al. A cryptographic system based on automata for networked automation systems abstracted as discrete-event systems. In: **2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)**. Bari, Italy: IEEE, 2024. p. 1462–1468. Cited 2 times on pages 28 and 33.

OLIVEIRA, Samuel et al. Robust recovery and control of cyber-physical discrete event systems under actuator attacks. In: **2025 IEEE 64th Conference on Decision and Control (CDC)**. Rio de Janeiro, Brazil: IEEE, 2025. p. 1220–1226. Cited 9 times on pages 18, 43, 45, 59, 60, 61, 68, 75, and 84.

OLIVEIRA, Samuel et al. A classification of cybersecurity strategies in the context of discrete event systems. **Annual Reviews in Control**, v. 56, p. 100907, 2023. ISSN 1367-5788. Cited 8 times on pages 18, 20, 23, 29, 32, 33, 34, and 43.

OLIVEIRA, Samuel et al. Security of cyber-physical systems against actuator attacks through cryptography*. In: **2023 International Conference on Information Technology (ICIT)**. Amman, Jordan: IEEE, 2023. p. 758–764. Cited 2 times on pages 43 and 59.

OLIVEIRA, Samuel et al. Segurança de sistemas ciberfísicos contra ataques a atuadores: um método baseado em criptografia. In: **Procedings do XVI Simpósio Brasileiro de Automação Inteligente / X Simpósio Brasileiro de Sistemas Elétricos**. Manaus, AM, Brasil: SBA Sociedade Brasileira de Automática, 2023. ISSN 2177-6164. Cited on page 59.

OLIVEIRA, Samuel et al. Integrity of cyber-physical discrete event systems under covert actuator attacks. **IFAC-PapersOnLine**, v. 58, n. 1, p. 198–203, 2024. ISSN 2405-8963. Cited 6 times on pages 18, 20, 43, 59, 62, and 63.

OLIVEIRA, Samuel et al. Uma abordagem baseada em substituição de eventos para mitigar ataques a atuadores em sistemas a eventos discretos. In: **Proceedings do Congresso Brasileiro de Automática**. Rio de Janeiro, RJ, Brasil: SBA Sociedade Brasileira de Automática, 2024. ISSN 2525-8311. Cited on page 59.

PAOLI, Andrea; SARTINI, Matteo; LAFORTUNE, Stéphane. Active fault tolerant control of discrete event systems using online diagnostics. **Automatica**, v. 47, n. 4, p. 639–649, 2011. ISSN 0005-1098. Cited 2 times on pages 20 and 38.

RAMADGE, Peter J.G.; WONHAM, Walter M. Supervisory control of a class of discrete event processes. **SIAM Journal on Control and Optimization**, v. 25, n. 1, p. 206–230, 1987. Cited 2 times on pages 23 and 76.

RAMADGE, Peter J.G.; WONHAM, Walter M. The control of discrete event systems. **Proceedings of the IEEE**, v. 77, n. 1, p. 81–98, 1989. Cited 2 times on pages 25 and 27.

RASHIDINEJAD, Aida et al. Supervisory control of discrete-event systems under attacks: An overview and outlook. In: **2019 18th European Control Conference (ECC)**. Naples, Italy: IFAC, 2019. p. 1732–1739. Cited on page 20.

REIS, Lucas N. R.; CARVALHO, Lilian K.; MOREIRA, Marcos V. Enforcing current-state opacity and utility of cyber-physical systems using multipath event routing. **IEEE Transactions on Automatic Control**, Institute of Electrical and Electronics Engineers (IEEE), p. 1–8, 2026. ISSN 2334-3303. Cited 2 times on pages 33 and 34.

RITSUKA, K. et al. Detectability of discrete event systems under sensor attacks. **Journal of Systems Science and Complexity**, Springer Science and Business Media LLC, v. 38, n. 1, p. 150–177, fev. 2025. ISSN 1559-7067. Cited 2 times on pages 42 and 43.

SABOORI, Anooshiravan; HADJICOSTIS, Christoforos N. Notions of security and opacity in discrete event systems. In: **2007 46th IEEE Conference on Decision and Control**. New Orleans, LA, USA: IEEE, 2007. p. 5056–5061. Cited 3 times on pages 28, 33, and 42.

SABOORI, Anooshiravan; HADJICOSTIS, Christoforos N. Verification of k-step opacity and analysis of its complexity. In: **Proceedings of the 48h IEEE Conference on**

**Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference**. Shanghai, China: IEEE, 2009. p. 205–210. Cited on page 33.

SAKATA, Kousei et al. Model verification of fallback control system under cyberattacks via uppaal. **Advanced Robotics**, Taylor & Francis, v. 37, n. 3, p. 156–168, 2023. Cited 4 times on pages 20, 21, 40, and 63.

SAKATA, Kousei; FUJITA, Shintaro; SAWADA, Kenji. Synthesis of resilient third-party monitoring system against cyberattacks via supervisory control. In: **2021 IEEE International Conference on Consumer Electronics (ICCE)**. Las Vegas, NV, USA: IEEE, 2021. p. 1–6. Cited on page 39.

SAKATA, Kousei et al. Synthesis of resilient fallback control system under cyber-attacks via supervisory control. **Advanced Robotics**, Taylor & Francis, v. 38, n. 9-10, p. 659–671, 2024. Cited 4 times on pages 20, 21, 40, and 63.

SHAMLOO, Naeimeh Fakhr; SANTIS, Elena De; BENEDETTO, Maria Domenica Di. Security and diagnosability of finite state machines under cyber-attacks. **IEEE Transactions on Automation Science and Engineering**, v. 22, p. 5108–5116, 2025. ISSN 1558-3783. Cited 2 times on pages 41 and 43.

SU, Rong. A cyber attack model with bounded sensor reading alterations. In: **2017 American Control Conference (ACC)**. Seattle, WA, USA: IEEE, 2017. p. 3200–3205. Cited 2 times on pages 30 and 35.

SU, Rong. Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations. **Automatica**, v. 94, p. 35 – 44, 2018. ISSN 00051098. Cited 4 times on pages 20, 36, 42, and 46.

SU, Rong. About existence of resilient supervisors against smart sensor attacks. In: **2022 IEEE 61st Conference on Decision and Control (CDC)**. Cancun, Mexico: IEEE, 2022. p. 4263–4269. Cited on page 36.

TAI, Ruochen; LIN, Liyong; SU, Rong. Supervisor fortification against covert actuator attacks. In: **2023 62nd IEEE Conference on Decision and Control (CDC)**. Singapore: IEEE, 2023. p. 7917–7922. ISSN 2576-2370. Cited 2 times on pages 37 and 43.

TAI, Ruochen; LIN, Liyong; SU, Rong. Synthesis of optimal covert sensor–actuator attackers for discrete-event systems. **Automatica**, v. 151, p. 110910, 2023. ISSN 0005-1098. Cited on page 29.

TAI, Ruochen; LIN, Liyong; SU, Rong. On decidability of existence of fortified supervisors against covert actuator attackers. **IEEE Transactions on Automatic Control**, v. 69, n. 3, p. 1898–1905, March 2024. ISSN 1558-2523. Cited 2 times on pages 37 and 43.

TAI, Ruochen; LIN, Liyong; SU, Rong. On decidability of existence of resilient supervisors against covert sensor-actuator attacks*. In: **2024 IEEE 63rd Conference on Decision and Control (CDC)**. Milan, Italy: IEEE, 2024. p. 4462–4467. ISSN 2576-2370. Cited 2 times on pages 37 and 43.

TAI, Ruochen et al. Synthesis of the supremal covert attacker against unknown supervisors by using observations. **IEEE Transactions on Automatic Control**, v. 68, n. 6, p. 3453–3468, 2023. Cited 3 times on pages 20, 35, and 44.

TAI, Ruochen et al. Synthesis of distributed covert sensor–actuator attackers. **IEEE Transactions on Automatic Control**, v. 69, n. 8, p. 4942–4957, Aug 2024. ISSN 1558-2523. Cited on page 36.

THORSLEY, David; TENEKETZIS, Demosthenis. Intrusion detection in controlled discrete event systems. In: **Proceedings of the 45th IEEE Conference on Decision and Control**. San Diego, CA, USA: IEEE, 2006. p. 6047–6054. Cited 3 times on pages 20, 38, and 42.

TONG, Yin; CAI, Kai; GIUA, Alessandro. Decentralized opacity enforcement in discrete event systems using supervisory control. In: **2018 57th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)**. Nara, Japan: IEEE, 2018. p. 1053–1058. Cited on page 33.

TONG, Yin; CAI, Kai; SEATZU, Carla. Actuator attack mitigation using the detection-protection mechanism. **IFAC-PapersOnLine**, v. 58, n. 1, p. 180–185, 2024. ISSN 2405-8963. 17th IFAC Workshop on discrete Event Systems WODES 2024. Cited 2 times on pages 20 and 38.

TONG, Yin et al. Current-state opacity enforcement in discrete event systems under incomparable observations. **Discrete Event Dynamic Systems: Theory and Applications**, v. 28, n. 2, p. 161–182, 2018. Cited on page 43.

TONG, Yin; WANG, Yucheng; GIUA, Alessandro. A polynomial approach to verifying the existence of a threatening sensor attacker. **IEEE Control Systems Letters**, v. 6, p. 2930–2935, 2022. Cited 2 times on pages 18 and 38.

WANG, Kejin; TONG, Yin. Sensor and actuator attack identification in discrete event systems. In: **2022 41st Chinese Control Conference (CCC)**. Hefei, China: IEEE, 2022. p. 1605–1610. Cited on page 31.

WANG, Yu et al. Attack-resilient supervisory control of discrete-event systems: A finite-state transducer approach. **IEEE Open Journal of Control Systems**, v. 2, p. 208–220, 2023. Cited 2 times on pages 20 and 38.

WANG, Yi et al. Supervisory control of discrete-event systems under external attacks. **Information Sciences**, v. 562, p. 398–413, 2021. Cited on page 37.

WANG, Yu; PAJIC, Miroslav. Attack-resilient supervisory control with intermittently secure communication. In: **2019 IEEE 58th Conference on Decision and Control (CDC)**. Nice, France: IEEE, 2019. p. 2015–2020. Cited on page 38.

WANG, Yu; PAJIC, Miroslav. Supervisory control of discrete event systems in the presence of sensor and actuator attacks. In: **2019 IEEE 58th Conference on Decision and Control (CDC)**. Nice, France: IEEE, 2019. p. 5350–5355. Cited on page 38.

WANG, Ze Y. et al. Mitigation of classes of attacks using a probabilistic discrete event system framework∗. **IFAC-PapersOnLine**, v. 53, n. 4, p. 35–41, 2020. ISSN 2405-8963. 15th IFAC Workshop on Discrete Event Systems WODES 2020 — Rio de Janeiro, Brazil, 11-13 November 2020. Cited 2 times on pages 20 and 62.

WATANABE, Ana T.Y. et al. Fault prognosis of discrete event systems: An overview. **Annual Reviews in Control**, v. 51, p. 100–110, 2021. ISSN 1367-5788. Cited 2 times on pages 41 and 43.

WATANABE, Ana T. Y. et al. Combining online diagnosis and prognosis for safe controllability. **IEEE Transactions on Automatic Control**, v. 67, n. 10, p. 5563–5569, 2022. Cited 2 times on pages 20 and 43.

WINTENBERG, Andrew et al. A dynamic obfuscation framework for security and utility. In: **2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)**. Milano, Italy: IEEE, 2022. p. 236–246. Cited 3 times on pages 33, 34, and 43.

WINTENBERG, Andrew; OZAY, Necmiye; LAFORTUNE, Stéphane. Integrating obfuscation and control for privacy. **IEEE Transactions on Automatic Control**, v. 70, n. 10, p. 6545–6560, 2025. Cited 3 times on pages 33, 34, and 43.

WU, Yi-Chin; LAFORTUNE, Stéphane. Synthesis of insertion functions for enforcement of opacity security properties. **Automatica**, v. 50, n. 5, p. 1336–1348, 2014. Cited on page 43.

WU, Yi-Chin; LAFORTUNE, Stéphane. Synthesis of optimal insertion functions for opacity enforcement. **IEEE Transactions on Automatic Control**, v. 61, n. 3, p. 571–584, 2016. Cited on page 33.

WU, Yi-Chin et al. Synthesis of obfuscation policies to ensure privacy and utility. **Journal of Automated Reasoning**, Springer Science and Business Media LLC, v. 60, n. 1, p. 107–131, jul. 2017. ISSN 1573-0670. Cited 2 times on pages 33 and 34.

XIE, Gang et al. Resilient supervisor synthesis for labeled petri nets against sensor attacks. **IEEE Transactions on Automatic Control**, v. 70, n. 6, p. 4045–4052, 2025. Cited on page 20.

XIE, Yujie; TONG, Yin. Resilient and nonblocking supervisor synthesis for discrete-event systems under actuator attacks. In: **2025 44th Chinese Control Conference (CCC)**. Chongqing, China: IEEE, 2025. p. 1562–1567. ISSN 1934-1768. Cited on page 37.

YANG, Jingkai; DENG, Weilin. Opacity verification for a class of modular discrete event systems. **IEEE Access**, v. 13, p. 54080–54089, 2025. Cited on page 33.

YAO, Jingshi; YIN, Xiang; LI, Shaoyuan. On attack mitigation in supervisory control systems: A tolerant control approach. In: **59th IEEE Conference on Decision and Control (CDC)**. Jeju, Korea (South): IEEE, 2020. p. 4504–4510. Cited 4 times on pages 20, 36, 38, and 43.

YAO, Jingshi; YIN, Xiang; LI, Shaoyuan. Sensor deception attacks against initial-state privacy in supervisory control systems. In: **2022 IEEE 61st Conference on Decision and Control (CDC)**. Cancun, Mexico: IEEE, 2022. p. 4839–4845. Cited on page 43.

YU, Zhenhua et al. Detection of actuator enablement attacks by petri nets in supervisory control systems. **Mathematics**, v. 11, n. 4, 2023. ISSN 2227-7390. Cited on page 38.

YU, Zhenhua; JIA, Yifei; CONG, Xuya. Detection of actuator enablement attacks by time petri nets in supervisory control. **Transactions of the Institute of Measurement and Control**, SAGE Publications, jan. 2026. ISSN 1477-0369. Cited on page 38.

ZHANG, Qi. Robust predictability in discrete event systems under sensor attacks. **Frontiers in Physics**, v. 11, 2023. Cited on page 41.

ZHANG, Qi et al. Stealthy sensor attacks for plants modeled by labeled petri nets. **IFAC-PapersOnLine**, v. 53, n. 4, p. 14–20, 2020. ISSN 2405-8963. Cited on page 35.

ZHANG, Qi et al. Joint state estimation under attack of discrete event systems. **IEEE Access**, v. 9, p. 168068–168079, 2021. Cited on page 41.

ZHANG, Qi et al. Selection of a stealthy and harmful attack function in discrete event systems. **Scientific Reports**, v. 12, n. 1, 2022. Cited 2 times on pages 20 and 42.

ZHENG, Shengbao; SHU, Shaolong; LIN, Feng. Modeling and control of discrete event systems under joint sensor-actuator cyber attacks. In: **2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE)**. Dalian, China: IEEE, 2021. p. 216–220. Cited on page 38.

ZHENG, Shengbao; SHU, Shaolong; LIN, Feng. Modeling and control of discrete-event systems under joint sensor–actuator cyberattacks. **IEEE Transactions on Control of Network Systems**, v. 11, n. 2, p. 782–794, June 2024. ISSN 2325-5870. Cited 2 times on pages 38 and 42.

ZHOU, Sian et al. Homomorphic encryption of supervisory control systems using automata. **IEEE Access**, v. 8, p. 147185–147198, 2020. Cited 2 times on pages 28 and 33.

ZHU, Yuting; LIN, Liyong; SU, Rong. Supervisor obfuscation against actuator enablement attack. In: **2019 18th European Control Conference (ECC)**. Naples, Italy: IEEE, 2019. p. 1760–1765. Cited 3 times on pages 21, 37, and 43.

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
BIBLIOTECA UNIVERSITÁRIA
REPOSITÓRIO INSTITUCIONAL

CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT

**ATESTADO DE VERSÃO FINAL**

Eu, André Bittencourt Leal, professor do Programa de Pós-Graduação em Engenharia Elétrica, declaro que esta é a versão final aprovada pela comissão julgadora da tese intitulada: **"Protection and Robust Recovery Strategies for Cyber-Physical Discrete Event Systems under Active Attacks"** de autoria do acadêmico Samuel Silva de Oliveira.

Joinville, 13 de Março de 2026.

Assinatura digital do orientador:

gov.br
Documento assinado digitalmente
ANDRE BITTENCOURT LEAL
Data: 13/03/2026 15:27:38-0300
Verifique em https://validar.iti.gov.br

_____

André Bittencourt Leal