UNIVERSIDADE D
COIMBRA

João António da Silva Melo

Opacity-Based Defense for Deterministic Finite Automata Against Passive and Actuator-Enablement Attacks

Dissertation in the context of the Master in Informatics Security, advised by Professors Tiago Cruz, Paulo Simões, Graziana Cavone, and Federica Pascucci, and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July 2024

DEPARTAMENTO DE
ENGENHARIA INFORMÁTICA
FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

João António da Silva Melo

Opacity-Based Defense for Deterministic Finite Automata Against Passive and Actuator-Enablement Attacks

Dissertation in the context of the Master in Informatics Security, advised by Professors Tiago Cruz, Paulo Simões, Graziana Cavone, and Federica Pascucci, and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July 2024

DEPARTAMENTO DE
ENGENHARIA INFORMÁTICA
FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

João António da Silva Melo

Defesa Baseada em Opacidade para Autómatos Finitios Deterministicos Contra Ataques Passivos e de Ativação de Atuadores

Dissertação no âmbito do Mestrado em Segurança Informática, orientada pelos Professores Tiago Cruz, Paulo Simões, Graziana Cavone e Federica Pascucci, e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Julho 2024

Acknowledgements

I wish to thank,

my supervisors in Italy for their extraordinary assistance in understanding complex concepts and overcoming challenges.

my supervisors in Portugal for making this opportunity possible and providing persistent encouragement.

my family for their acts of love and belief in me.

my friends for their companionship and support in maintaining balance and focus.

my girlfriend for being a motivational and positive influence.

my parents, who have always been there for me and have provided me with the opportunities that led me here.

Thank you all for making this objective easier to accomplish.

Abstract

In our rapidly evolving technological landscape, safeguarding systems against cyber threats is imperative. This dissertation, part of the Master's program in Informatics Security, focuses on advancing security methodologies within the domain of Discrete Event Systems (Discrete Event Systems (DES)), specifically targeting Deterministic Finite Automata (Deterministic finite automata (DFA)). Unlike conventional time-driven systems, certain human-engineered systems, such as real-time communication networks and automated manufacturing processes, exhibit distinctive event-driven traits requiring a unique approach for effective protection.

The primary objective of this research is to propose and validate an innovative security technique based on the concept of opacity within DES, with a particular emphasis on defending DFA systems against active attacks. Opacity ensures confidentiality and resilience by recognizing attackers as active intruders rather than passive observers.

A comprehensive literature review revealed a gap in the field concerning security through opacity, particularly in the context of active attacks. This research aims to bridge this gap by developing a novel opacity-based technique. The study spans two semesters, encompassing the modeling of the Hydra testbed as a DES and the iterative development, testing, and refinement of the proposed security technique.

In the initial phase, DES theory and opacity literature were explored, and the Hydra testbed, a water distribution system simulation, was introduced as a case study. The subsequent phase focused on formulating, testing, and validating the opacity-based technique against Actuator-Enablement (AE) attacks. The developed method successfully enhances security while relaxing some assumptions of previous works, making it more adaptable for practical use in physical systems like Hydra.

Through practical experiments within the Hydra system, the efficacy of the developed technique was validated, demonstrating its ability to defend DFA systems against active attacks effectively. While some limitations were identified, such as reliance on predefined sets of events, these findings provide valuable insights and lay the groundwork for future research to refine and extend the technique. This research contributes significantly to reinforcing security and resilience in DES systems under active threat scenarios.

Keywords

DES, DFA, Opacity, Active Attacks, Actuator-Enablement Attack, Supervisor, Mitigation Module

Resumo

No nosso cenário tecnológico em rápida evolução, proteger os sistemas contra ameaças cibernéticas é imprescindível. Esta dissertação, parte do programa de Mestrado em Segurança Informática, foca-se no avanço das metodologias de segurança no domínio dos Sistemas de Eventos Discretos (DES), com especial enfoque nos Autómatos Finitos Determinísticos (DFA). Ao contrário dos sistemas tradicionais orientados pelo tempo, certos sistemas projetados pelo homem, como as redes de comunicação em tempo real e os processos de produção automatizados, exibem características distintas orientadas por eventos, exigindo uma abordagem única para uma proteção eficaz.

O objetivo principal desta pesquisa é propor e validar uma técnica inovadora de segurança baseada no conceito de opacidade dentro dos DES, com ênfase particular na defesa dos sistemas DFA contra ataques ativos. A opacidade garante a confidencialidade e a resiliência ao reconhecer os atacantes como intrusos ativos, em vez de observadores passivos.

Uma revisão abrangente da literatura revelou uma lacuna no campo relativamente à segurança através da opacidade, particularmente no contexto de ataques ativos. Esta pesquisa visa preencher essa lacuna desenvolvendo uma técnica nova baseada em opacidade. O estudo abrange dois semestres, incluindo a modelação da plataforma de teste Hydra como um DES e o desenvolvimento, teste e refinamento iterativo da técnica de segurança proposta.

Na fase inicial, a teoria dos DES e a literatura sobre opacidade foram exploradas, e a plataforma de teste Hydra, uma simulação de um sistema de distribuição de água, foi introduzida como um caso prático. A fase subsequente concentrou-se na formulação, teste e validação da técnica baseada em opacidade contra ataques de Ativação de Atuadores (AE). O método desenvolvido melhora com sucesso a segurança enquanto relaxa algumas das suposições dos trabalhos anteriores, tornando-o mais adaptável para uso prático em sistemas físicos como o Hydra.

Através de experiências práticas no Hydra system, a eficácia da técnica desenvolvida foi validada, demonstrando a sua capacidade de defender eficazmente os sistemas DFA contra ataques ativos. Embora algumas limitações tenham sido identificadas, como a dependência de conjuntos de eventos predefinidos, estas descobertas fornecem insights valiosos e estabelecem as bases para futuras pesquisas para refinar e expandir a técnica. Esta pesquisa contribui significativamente para reforçar a segurança e a resiliência nos sistemas DES em cenários de ameaças ativas.

Palavras-Chave

DES, DFA, Opacidade, Ataques Ativos, Ataques de Ativação de Atuadores, Supervisor, Módulo de Mitigação

Contents

Acronyms

DES Discrete Event Systems.

DFA Deterministic finite automata.

NFA Nondeterministic finite automata.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

We inhabit a world increasingly shaped by technological dominance across every facet of our daily lives, where cyber-physical systems play an increasingly dominant role. In this landscape, the imperative for applying and evolving cybersecurity techniques has become non-negotiable, due to several factors ranging from security and safety issues to the potential negative impact of service interruptions [Rosa et al., 2021]. Specifically, this dissertation is driven by the central objective of exploring and advancing security techniques within the realm of Discrete Event Systems DES.

While the majority of systems are usually related to a time-driven paradigm, a notable subset of man-made systems, such as real-time communication networks or automated manufacturing processes, displays distinctive event-driven characteristics. These traits, marked by events triggering system responses, elude effective treatment through conventional time-driven models. Instead, these dynamic systems necessitate a nuanced approach that departs from process-neutral strategies (such as correlation or signature-based techniques [Rosa et al., 2015]), being able to capture system intricacies, ensuring a comprehensive understanding of their behaviour and responses to various events.

This dissertation, which is part of the Master's program in Informatics Security, delves into the concept of opacity within the confidentiality realm of security, focusing on DES, more specifically in DFA. It intends to provide a valuable contribution to informatics security by proposing an effective technique capable of withstanding active threats while upholding the crucial aspect of system opacity.

The field of DES has undergone significant developments in recent years, propelled by extensive research in the realm of cybersecurity, particularly concerning active attackers. According to [Oliveira et al., 2023], as of 2022, the number of papers dedicated to exploring the characteristics of active attacks has surpassed the contributions focused on passive attacks. This evolving landscape has ignited interest in comprehending how these attacks elicit reactions from systems and the methods by which such systems can be fortified against these threats.

A comprehensive literature review reveals a gap in the field, notably in introducing the concept of security by opacity and its application to active attacks. Within this dynamic context, this study delved into this unexplored territory and formulated a novel security technique applicable to DFA. The research is designed not only to address the specific aspects and questions surrounding security by opacity and active attacks but also to make a substantial contribution to the existing body of knowledge in the cybersecurity domain. The objective is to provide valuable insights into reinforcing security and resilience, particularly in the context of control applications.

1.2 Objectives and Contributions

This chapter delineates a roadmap that guides the research efforts of this dissertation within the realm of opacity in security, focusing on (DES). Rooted in the acknowledgement of attackers as active intruders, rather than passive observers, the objectives outlined here traverse two semesters, orchestrating a meticulous exploration of concepts and the development and application of a novel security technique.

Within these objectives lies the commitment to model the Hydra testbed as a DES, ensuring its consistency as a case-study for the opacity-based proposed technique. Simultaneously, the proposed technique will undergo iterative refinement, aligning it with the intricacies of active attacks. As the investigation progresses, a deliberate effort will be made to connect theoretical insights with practical applications, emphasizing the real-world implications of the developed technique.

The primary goal of this work is to define a novel and general technique, based on opacity, to ensure cybersecurity of DESs in case of active attacks, thus acknowledging attackers as active intruders, not just passive observers. Then, the first part of this work consists in analysing the state of the art on opacity of DES and the existing techniques devoted to opacity analysis and enforcement in the context of cybersecurity, with the aim of identifying the literature gaps and open issues. In addition, a suitable literature DES to be used to test and validate the opacity-based security technique is selected. In particular, the selected physical system, modeled as a DES, serves as a target for active attacks and to validate the effectiveness of the opacity-based technique.

The chosen system, is a test-bed named Hydra, which emulates a water distribution system, employing a combination of gravity and pumps to circulate the fluid within the system. While the physical structure of the testbed is designed with a low-cost approach, it is interfaced to the control system through an industrial PLC over a Modbus/TCP network. This system has been employed in previous research projects, as demonstrated by the example in [Battisti et al., 2018].

The second part of this work involves defining, testing, and validating an opacity-based technique. This process combines two distinct methods to enforce current-state opacity against passive and active attackers in Deterministic Finite Automata (DFA). The ultimate goal is to develop a technique that enhances security, ensur-

ing the system remains opaque, particularly in the face of active threats.

The technique focuses on defending a DFA system against Actuator-Enablement (AE) attacks. The developed method successfully achieves this goal while relaxing assumptions of previous works, making it more adaptable for use in physical systems, such as Hydra, the real system used in this dissertation's case study.

The following list summarises the key objectives to be pursued across the first and second semesters, offering a structured framework for navigating the environment of DES security and opacity.

- Delve into the theoretical foundations of DES, comprehending the fundamental principles about the dynamic interaction of events and state transitions within complex systems.
- Conduct a thorough literature review on opacity, focusing on DES susceptibility to active attacks.
- Develop a comprehensive understanding of the behaviour and responses of DES to various events, particularly in the presence of active attackers.
- Model the Hydra testbed as a DES, ensuring its consistency as a case-study for the opacity-based technique.
- Propose and outline a technique for ensuring opacity within DES systems, considering the nuances introduced by active attackers.
- Conduct a preliminary evaluation of the proposed technique through simulated scenarios, emphasizing its effectiveness in maintaining opacity under active attacks.
- Refine the proposed technique based on the outcomes of the practical case study, ensuring its viability and effectiveness.
- Implement the proposed technique on the Hydra testbed, transforming it into a practical case study for further examination.
- Compile final documentation, including the comprehensive report/dissertation, summarizing the entire research process and detailing the developed technique's contribution to reinforcing security and resilience in DES systems.

1.3 Structure

The dissertation structure for the rest of the document is described by:

- **Background -** This chapter provides the essential theoretical background for understanding the concepts of DES and opacity. Its contents and their organization assume that the reader may have limited knowledge of the topic. All the necessary information to comprehend the subsequent chapters is defined and explained here.

- **State-of-the-art -** In this chapter a comprehensive review of existing literature on the topic is conducted. The primary emphasis is on active attacks in DES and their integration with security measures, particularly the use of opacity in DES. In the end of this chapter, there is a discussion and analysis of the existant literature gap is also done.

- **Proposed Technique -** This chapter details the steps required to create the technique and a formal presentation of the technique itself. All of the decisions and conclusions regarding the literature presented in Chapter 3 are done in this chapter, as well as the evolution of the technique.

- **Case Study -** A brief description is provided, introducing the Hydra system, which will be used to implement the proposed technique. The physics-based model of the system is first introduced and then converted into a representation of a DFA model. The technique is applied to this system, going through the types of attacks. A final conclusion regarding the effectiveness of the technique is done in the end of the chapter.

- **Methodology -** In this chapter, the approach to address the problem is outlined. Risks are identified with consideration given to their probabilities, potential impacts, and proposed mitigations.

Chapter 2

Background

2.1 Introduction

In this chapter, notations are introduced to facilitate discussions of opacity and control in Discrete Event Systems (DES). The primary focus is on DES modeled through regular languages and finite automata. More detailed discussions about the introductory material can be found in relevant sources such as [Cassandras and Lafortune, 2010].

2.2 Discrete Event Systems

An event-driven system, often denoted as a discrete-event system, exhibits dynamic behavior within a discrete state space. The system's state evolves through segmented trajectories influenced by abrupt physical events, causing unpredictable shifts. Both the timing and characteristics of these events are not predictable.

The system's state, represented by logical or symbolic values rather than numerical ones, undergoes changes triggered by events described in non-numeric terms. Automata or finite state machines commonly serve as models for these systems. While initially perceived as distinct from time-driven systems, it is noteworthy that a physical system initially described by a time-driven model can often find an alternative representation in a discrete-event model. This abstraction process involves simplification to retain essential properties while obscuring non-essential details.

2.3 Formal Languages and Finite Automata

For simplicity and to avoid repetition throughout this section, all examples used in the following contents of the Background, have been inspired by [Basilio et al., 2021] and materials from Professor Giua 1.

1

2.3.1 Alphabets and Words

Within the domain of formal language theory, an essential field in the examination of discrete structures and automata, fundamental elements emerge as foundational components for the explication and analysis of languages. At the core of this lie the fundamental concepts of an alphabet, words, and diverse operators tasked with manipulating these linguistic structures.

Formally, an *alphabet* A is defined as a finite and non-empty set of *symbols*. The *cardinality* of the alphabet, represented by $|A|$, corresponds to the number of symbols contained within it.

Example: Considering the Latin alphabet as $A_1 = \{a, b, c, \ldots, x, y, z\}$, it is said that its cardinality is $|A_1| = 26$.

From the usage of the alphabet A, it is possible to create words, which are the sequential arrangement of symbols from A. The length of a word, denoted by $|w|$, corresponds to the count of symbols it comprises. Additionally, $|w|_s$ denotes the number of occurrences of a specific symbol $s \in A$ within the word w.

Example: Considering the alphabet defined in the previous example, A_1, and word $w_1 = example$, it is possible to conclude that w_1 is defined on A_1, $|w_1| = 7$, and $|w_1|_e = 2$.

The collection of words formed from an alphabet A is symbolized as A^*. The empty word, a sequence of zero length, exists across all alphabets and is consistently represented by ε. It is therefore empirical to mention that, while an alphabet A is always a finite set, A^* is always an infinite set, as there is no constraint to how long words can be.

In a scenario where DES are used, A is the alphabet of events that can occur in DES, and each symbol in A denotes an action that is assumed to be instantaneous in the system.

2.3.2 Operators on Words

Multiple operators are used on words, the first to be considered is the binary operator: *concatenation*. When two words $w_1 \in A^*$ and $w_2 \in A^*$ are concatenated, the result is a new word, $w_c = w_1 \cdot w_2$, composed by the sequence of symbols present in w_1 followed by the sequence of symbols in w_2. Usually, the symbol $\cdot$ is omitted, and the operation is represented as $w_1 w_2$.

From the definition of concatenation, one can understand that:

- the length of a word obtained by concatenation is equal to the sum of the length of the words that suffered the operation;

[1] Materials used from Professor Giua

- it is associative, i.e., $w_c = (w_1 w_2)w_3 = w_1(w_2 w_3) = w_1 w_2 w_3$;
- it is not commutative, i.e., considering $w_1 \neq w_2$, if $w_c = w_1 w_2$, then $w_c \neq w_2 w_1$;

it is important to mention that the *identity element* of this operator is the empty word ε, meaning that, for every word $w \in A^*$, it holds that $\varepsilon w = w\varepsilon = w$.

As it is used for multiplication in elementary arithmetic's, the *exponent notation* is commonly implemented to represent the concatenation of x identical symbols, by the usage of exponent x. For instance, the word *aaabbc* can be expressed as $a^3b^2c^1$, meaning that a is repeated three consecutive times, b twice, and c appears once in the end of the word. To generalize this, for any symbol $s \in A$, the convention of denoting $e^0 = \varepsilon$ is used. This choice is made because $s^k s^0 = s^{k+0} = s^k$, and for this equation to hold true, s^0 must be the identity element, ε.

It is essential to consider key notations that describe various relationships between words within a larger word, including prefixes, suffixes, and substrings. For example, given word $w \in A^*$ that can be written as $w = xyz$ where $x, y, z \in A^*$, then word x is a prefix of w, represented as $u \preceq w$, word y is called a substring of w, and z is a suffix of w. Considering the **following example**, where word $w = 1234$, the prefixes of w are ε, 1, 12, 123, and 1234. Its suffixes are ε, 4, 34, 234, and 1234. Finally, its substrings comprise the set of all prefixes, suffixes, and the following strings: 2, 3, and 23.

Another operation to take into consideration is the projection of words from alphabet A^* on a subset $A' \subseteq A$. It is denoted by $w \uparrow A'$, and its result is the word obtained by removing from w all the symbols that don't belong to A'. **For example**, considering $A = \{1, 2, 3\}$ and $A' = \{1, 2\}$. When the following word is taken into consideration, $w = 1131223233$, its projection on A' is obtained by removing all 3's from it, so $w \uparrow A' = 111222$.

2.3.3 Languages

A language L, defined in an alphabet A, is a collection of words formed from the symbols in A. The *cardinality* of this language, represented by $|L|$, is the number of distinct words it contains.

The following **examples** of languages defined on alphabet $A = \{1, 2, 3\}$ may be considered to better understand this concept.

$$L_1 = \{1, 12, 123\}, \quad L_2 = \{\varepsilon, 1\}, \quad L_3 = \{w \in A^* \mid |w| = 5\}$$

In language L_1, there are three possible words; therefore, $|L_1| = 3$. Language L_2 consists of two words, including the empty word. Finally, in L_3, all words of length five defined on A are contained in this language.

As languages are sets of words, it is possible to make comparisons between two of them through the *inclusion*, $\subseteq$, and *strict inclusion*, $\subset$, relations. **For example**,

language $L_1 = \{1\}$ is strictly included in language $L_2 = \{1, 11\}$, and none of these is included in language $L_3 = \{111\}$.

2.3.4 Operators on Languages

There are many operators that can be applied to languages, to start off, the usual binary set operators, such as *intersection* and *union*, can be taken into consideration.
To better understand these concepts, let $L_1 \subseteq A_1^*$ and $L_2 \subseteq A_2^*$ be two languages, and let $A_I = A_1 \cap A_2$ be the intersection and $A = A_1 \cup A_2$ the union of the alphabets. The definition of the union and intersection of L_1 and L_2 is given by:

- ***Union:*** $L_1 \cap L_2 = \{w \in A^* \mid w \in L_1 \vee w \in L_2\}$;
- ***Intersection:*** $L_1 \cup L_2 = \{w \in A_I^* \mid w \in L_1, w \in L_2\}$;

For example, considering $L_1 = \{\varepsilon, DES\}$ and $L_2 = \{DES, DFA, NFA\}$, then $L_1 \cap L_2 = \{DES\}$ and $L_1 \cup L_2 = \{\varepsilon, DES, DFA, NFA\}$.

Some characteristics of the intersection and union operators are that:

- Both are associative and commutative;
- The intersection operator has as the identity element the language A^*, the explanation is that, for all $L \subseteq A^*$ it holds that $L \cap A^* = A^* \cap L = L$;
- The language $\emptyset$ is the union operator's identity element, since, for all $L \subseteq A^*$ it holds that $L \cup \emptyset = \emptyset \cup L = \emptyset$;

As previously defined in 2.3.2, the concatenation operator also exists as an operator to Languages. Considering $L_1, L_2 \subseteq A^*$ as two languages, the *concatenation* of L_1 and L_2 is defined as the language

$$L_1 L_2 = \{w = w_1 w_2 \in A^* \mid w_1 \in L_1, w_2 \in L_2\},$$

which consists of all possible combinations obtained by concatenating a word from L_1 with a word from L_2. **For example**, considering $L_1 = \{\varepsilon, DES\}$ and $L_2 = \{DES, DFA, NFA\}$, then $L_1 L_2 = \{\varepsilon \cdot DES\} \cup \{\varepsilon \cdot DFA\} \cup \{\varepsilon \cdot NFA\} \cup \{DES \cdot DES\} \cup \{DES \cdot DFA\} \cup \{DES \cdot NFA\} = \{DES, DFA, NFA, DESDES, DESDFA, DESNFA\}$.

As previously defined in 2.3.2, when applied into languages, the *concatenation* operator, shares some of the characteristics, as it is associative, non-commutative and the identity element is a language that consists of the empty word $\{\varepsilon\}$, i.e., for all $L \subseteq A^*$, $L\{\varepsilon\} = \{\varepsilon\}L = L$. Other ways of denoting this operator are, for all $L \subseteq A^* : L^0 = \{\varepsilon\}$, $L_3 = LLL$, *etc.*
Finally, the concatenation operator is distributive with respect to the union, i.e, $(L_1 \cup L_2)L_3 = L_1 L_3 \cup L_2 L_3$.

Some unary operators on languages are the *Kleene star*, the *prefix closure*, and the *complement*.

The *Kleene* Star (or *Kleene closure*), when applied to language $L \subseteq A^*$, is the language

$$L^* = \{\varepsilon\} \cup L \cup LL \cup \ldots \bigcup_{i=0}^{\infty} L^i,$$

which includes all words formed by concatenating words from language L, an unlimited number of times.
For example, if $L = \{22\}$ is a language defined by alphabet $A = \{2\}$, then $L^* = \{\varepsilon\} \cup \{22\} \cup \{2222\} \cup \ldots = \{(22)^n \mid n \geq 0\} \subset A^*$.

The *prefix closure* of a given language $L \subseteq A^*$ is the language,

$$pref(L) = \{u \in A^* \mid \text{there is } w \in L : u \preceq w\}$$

which consists of all prefixes of words in L. Evidently, $L \subseteq pref(L)$, since when a word w is in L, then w is also in $pref(L)$. A language L is called *prefix closed* if $L = pref(L)$ holds.
To better understand the concept of *prefix closed*, consider $L_1 = \{\varepsilon, 1, 2, 12\}$ and $L_2 = \{1, 2, 12\}$. Then, $L_1 = pref(L_1)$ holds, and therefore this language is prefix closed. On the other hand, for the case of L_2, $L_2 \neq pref(L_2) = \{\varepsilon, 1, 2, 12\}$, thus making L_2 not prefix closed.
As mentioned previously, another unary operator used on languages is the *complement*, which given language $L \subseteq A^*$, its complement is the language,

$$L' = \{w \in A^* \mid w \notin L\}$$

which consists of all words that don't belong to L. L' can also be written as, $L' = A^* \backslash L$.
A simple **example**, considering $L = \{1, 11, 111\}$, defined on alphabet $A = \{1\}$; the complement of L is $L'_1 = \{1^k \mid k = 0 \vee k \geq 4\}$.

2.4 Deterministic Finite Automata (DFAs)

As shown in the previous chapter, formal languages can be described either by listing their words, or by set notation. However, in this chapter languages will be defined using a type of *generators*, i.e., a structure to which a language can be associated, these *generators* are referred to as *discrete event models* and in this chapter the focus will be on *Deterministic finite automata (DFA)*. This model is built on two basic components: states and transitions. It simply describes how a dynamic system changes from one state to another when specific discrete events happen.

2.4.1 Definition

A DFA is represented by a quintuple

$$G = (X, A, \delta, x_0, X_m)$$

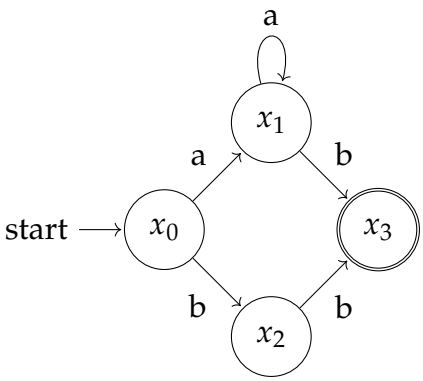


Figure 2.1: A DFA

where:

- X is a finite set of *states*;
- A is an alphabet;
- δ: $X \times A \rightarrow X$ is a transition function;
- $x_0 \in X$ is an *initial state*;
- $X_m \subseteq X$ is a set of *final states*

DFAs are used to describe DES, meaning that the alphabet will represent a finite set of events. The dynamics of the automaton are specified in the transition function, meaning that, if $\bar{x} = \delta(x, e)$, then the occurrence of event e, when the current-state of the system is x, will lead to state $\bar{x}$.

As represented in figure 2.1, an automaton can be described as a graph in which each node corresponds to a distinct state, symbolized by circular shapes. In this specific representation, the initial state is indicated by an incoming arrow, designating it as the system's starting point, while the final state is denoted by a double circle. 2.1 shows the graphical structure of an automaton with $X = \{x_0, x_1, x_2, x_3\}$, alphabet $A = \{a, b\}$, the initial state is x_0, and the set of final states is $X_m = x_3$. The transition function for this automata can be given by the following table:

δ	a	b
x_0	x_1	x_2
x_1	x_1	x_3
x_2	-	x_3
x_3	-	-

Table 2.1: Transition function for Figure 2.1

To understand this table, the value x_2 at the intersection between row x_0 and column b, denotes that $\delta(x_0, b) = b$. A box with the symbol '-', such as the one at the intersection between row x_3 and column a, means that such transition is not defined.

The set of events *enabled* (or active) in state x of a given DFA $G = (X, A, \delta, x_0, X_m)$, is

$$\mathcal{A}(x) = \{e \in A \mid \delta(x, e) \text{ is defined}\}.$$

To specify that an event $e \in \mathcal{A}(x)$, another type of notation is $\delta(x, e)!$, which indicates that for the pair (x, e) the function δ is defined.

A *run* of length k, in a given DFA $G = (X, A, \delta, x_0, X_m)$, is a sequence of states and transitions

$$x_0 \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_{k-1}} x_{k-1} \xrightarrow{e_k} x_k$$

in which, for all $i = 0, \ldots, k$ it is certain that $x_i \in X$ and for all $i = 1, \ldots, k$ it is also certain that $x_i = \delta(x_{i-1}, e_i)$, i.e, state x_i is reached through the occurrence of event e_i when the current state of the system is x_{i-1}. From this run, it is known that it started from state x_0 and that it produced the word $w = e_1 e_2 ... e_k$ when state x_k is reached.
Some important aspects to consider is that a run doesn't need to mandatorily start from an initial state, and it is also possible to define a run with length zero, this is run in which no transition occurs and thus the current state of the system stays constant.

2.4.2 Languages of DFAs

As mention in the previous subsection, at the end of a run of an automata there is a word on alphabet A that is produced. Therefore, the set of all possible runs that start from the initial state will also be associated with all corresponding words that define language $L \subseteq A^*$.

Considering a DFA $G = (X, A, \delta, x_0, X_m)$, word w can be:

- **generated** if $\delta^*(x_0, w)!$, i.e, a run that starts from the initial state and produces w, exists;
- **accepted** if $\delta^*(x_0, w) = x \in X_m$, i.e, a run that starts from the initial state, reaches a final state, and produces w, exists;

From these definitions, two languages can be associated with the given automata:

- the **generated language**, i.e, the set that has all generated words:

$$L(G) = \{w \in A^* \mid \delta^*(x_0, w)!\} \subseteq A^*;$$

- the **accepted language**, i.e, the set that has all accepted words:

$$L_m(G) = \{w \in A^* \mid \delta^*(x_0, w) \in X_m\} \subseteq L(G);$$

it is important to note that the language that is generated by a DFA is always prefix closed, i.e, $L(G) = pref(L(G))$: if a word can be generated, so can all of its prefixes.
On the other hand, the language that is accepted by a DFA is not necessarily prefix closed, since not all prefixes of an accepted word need to be accepted, thus it holds that $L_m(G) \subseteq pref(L_m(G))$. The only way that $L_m(G) = pref(L_m(G))$ is if and only if $X_m = X$, i.e, all states of automaton G are final.

2.4.3 Modeling with deterministic automata

In this subsection about DFA, a brief example of a DES is discussed and a corresponding DFA model is presented. The system represents a communication protocol.

Considering a system that operates in a network environment with occasional disruptions. Initially in the *standby* state, when a data transmission request **(request)** is received, the system transitions to the *preparing data* state. Once the data is ready, it is sent over the network **(send)**, and the system awaits an acknowledgment **(ack)** from the receiving end to confirm successful transmission. Upon receiving the acknowledgment, the system returns to the *standby* state, ready for new transmission requests.
However, if the acknowledgment is not received within a predefined time interval, indicating a likely communication failure **(error)**, the system initiates a retry mechanism to resend the data. The process of attempting to resend the data continues until successful transmission.
A simplified representation of this protocol can be described by a DFA, and is represented in Figure 2.2. The alphabet considered is $A = \{\text{req, send, error, ack}\}$. The proposed model has three states, the association of these to the physical meaning is done next to the figure of the DFA. It is assumed that a unique final state, perhaps the *standby* state, is reached because, after receiving a request, the transmission process must be brought to completion.
The DFA diagram representing this system can be designed with appropriate states and transitions to capture the dynamics of the described communication protocol.

2.5 Nondeterministic finite automata

For this section, a second discrete event model, called *Nondeterministic finite automata (NFA)*, is introduced. This new model can be considered a generalization of a DFA.

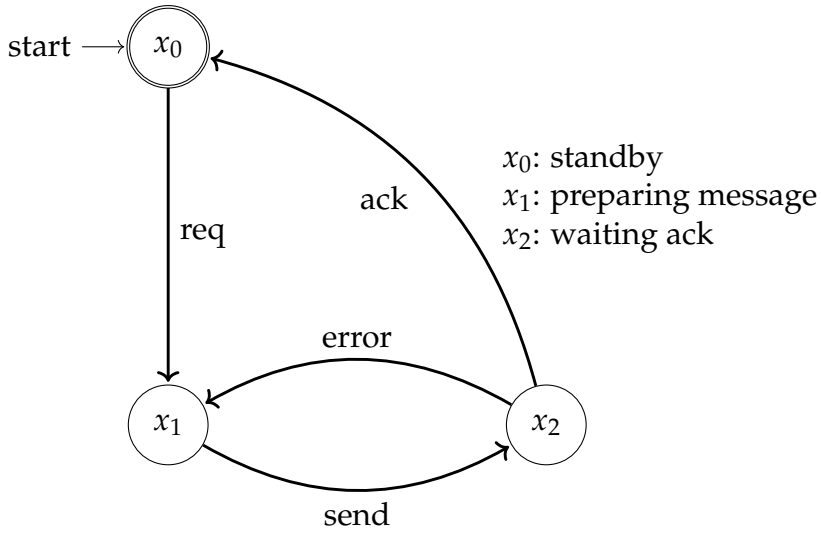


Figure 2.2: Model of a DFA for a communication protocol

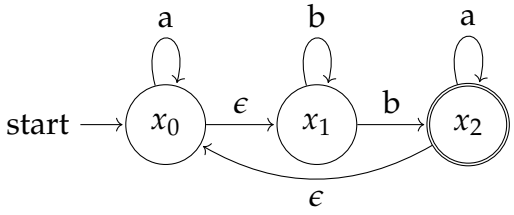


Figure 2.3: NFA Example

2.5.1 Definition

In Section 2.4.1, the definition of a DFA was done using a quintuple. Similarly, the definition of a NFA is also formulated using a quintuple.

$$G = (X, A, \Delta, x_0, X_m),$$

where:

- X is a finite set of *states*;
- A is an alphabet;
- $\Delta \subseteq X \times A_\epsilon \times X$ is the *transition relation*, with $A_\epsilon = A \cup \{\epsilon\}$;
- $x_0 \in X$ is an *initial state*;
- $X_m \subseteq X$ is a set of *final states/ marked states*.

It is in the transition relation where the dynamics of the automaton are specified, since, if $(x, e', \bar{x}) \in \Delta$, then when the system's current state is x, the occurrence of an e'*-transition*, which can be a symbol of A^*, leads to state $\bar{x}$.
As it was done in 2.1, a graphical representation of an NFA can be given using the same formalism.

Figure 2.3 shows an NFA with $X = \{x_0, x_1, x_2\}$, alphabet $A = \{a, b\}$, the initial state x_0 and the set of final states $X_m = \{x_2\}$. The transition function can be given by:

$$\Delta = \{(x_0, a, x_0), (x_0, \epsilon, x_1), (x_1, b, x_1), (x_1, b, x_2), (x_2, a, x_2), (x_2, \epsilon, x_0)\}$$

As previously mentioned in the beginning of this section, a NFA can be seen as a generalization of a DFA. The transition relation Δ is essentially a generalization of the transition function δ and it introduces two different nondeterministic primitives, that can be observed in the figure 2.3.

- The introduction of transitions using the empty word ϵ, these are also referred to as ϵ-transitions, and they describe "*silent*"/"*unobservable*" events which, as the name suggests, occur without being possible to observe.
- Multiple transitions outgoing from the same state having the same event in the label. These transitions describe "*indistinguishable events*", i.e, it is possible to detect that an event has occurred but it is not possible to determine uniquely which transition was taken by the system among two or more transitions with the same label.

Another similarity between DFAs and NFAs, is that the behavior of both is given by all of their possible evolution's characterized by their runs.
Given an NFA $G = (X, A, \Delta, x_0, X_m)$, a *run* of length k is a sequence of states and transitions

$$x_0 \xrightarrow{e_1} x_1 \xrightarrow{e_2} \dots \xrightarrow{e_{k-1}} x_{k-1} \xrightarrow{e_k} x_k$$

where: $x_i \in X$ for all $i = 0, \dots, k$, and $(x_{i-1}, e_i, x_i) \in \Delta$ for all $i = 1, \dots, k$. it is important to remind that an event $e \in A_\epsilon$ may be an event of A or the empty word ϵ. The run that was presented, starts from state x_0 and produces the word $w = e_1 e_2 \dots e_k$, reaching state x_k. Considering the following run of the automaton in Figure 2.3,

$$x_0 \xrightarrow{a} x_0 \xrightarrow{\epsilon} x_1 \xrightarrow{b} x_1 \xrightarrow{b} x_1$$

it starts from state x_0, and produces the word $w = abb$, reaching state x_1. This run has a *length* smaller than the actual number of transitions that were involved in it, since $\mid w \mid = 3$, while the run contains 4 transitions.
Since the Δ is a transition relation and not a function, it is possible to have two distinct runs that start from the same state and produce the same word, for example,

$$x_0 \xrightarrow{a} x_0 \xrightarrow{\epsilon} x_1 \xrightarrow{b} x_1 \xrightarrow{b} x_2$$

this run also starts from x_0 and produces the word $w = abb$, reaching state x_2. This is the characteristic that makes the automaton nondeterministic. The concept of nondeterminism may appear distinct from the commonly used notion in systems theory, where a system is deemed deterministic if, given a specific initial

condition and an input signal, there exists only one possible trajectory of evolution. However, these two notions align when a sequence of events is framed as the input for the system and its execution as the corresponding evolution.
Following this concept, and given a NFA $G = (X, A, \Delta, x_0, X_m)$, the *transitive and reflexive closure* of the transition relation Δ is the relation $\Delta^* \subseteq X \times A^* \times X$, such that, $(x, w, \bar{x}) \in \Delta^*$ if there exists a run

$$x \xrightarrow{e_1} x_1 \xrightarrow{e_2} \dots \xrightarrow{e_{k-1}} x_{k-1} \xrightarrow{e_k} \bar{x}$$

that begins in state x and produces the word $w = e_1 e_2 \dots e_k$, reaching state $\bar{x}$. An importation observation is that $(x, \epsilon, x) \in \Delta^*$ for all $x \in X$, i.e, taking any state as the starting point in a run with length zero, the automaton remains in the same state and produces the empty word. For **example**, for the automaton in figure 2.3, it is true that

$$(x_0, abb, x_1) \in \Delta^* \text{ and } (x_0, abb, x_2) \in \Delta^*.$$

2.5.2 Languages of a NFAs

Since nondeterminism exists in a NFA, the notion of a word that is accepted by it needs to be treated with particular care.
Considering a NFA $G = (X, A, \Delta, x_0, X_m)$, a word $w \in A^*$ is:

- **generated** state $x \in X$ exists, such that $(x_0, w, X) \in \Delta^*$, i.e, a run that starts from the initial state, and produces w, exists;
- **accepted** state $x \in X_m$ exists, such that $(x_0, w, X) \in \Delta^*$, i.e, a run that starts from the initial state, reaches the final state, and produces w, exists;

One can conclude that, due to the nondeterminism of these systems, word w may be produced by multiple runs that start from an initial state. A word w is **accepted**, if at least one of these runs ends in a final state. Considering the figure 2.3, and the word $w = abb$, as shown before, its possible to obtain it through different runs, and it is an accepted word by the system. These runs,

$$x_0 \xrightarrow{a} x_0 \xrightarrow{\epsilon} x_1 \xrightarrow{b} x_1 \xrightarrow{b} x_1$$
$$x_0 \xrightarrow{a} x_0 \xrightarrow{\epsilon} x_1 \xrightarrow{b} x_1 \xrightarrow{b} x_2$$

show that on the first one, as it doesn't lead to a final state, the run by itself proves that the word is *generated* by the system. As for the second run, since it leads to state x_2, the word *abb* is accepted.

Given a NFA $G = (X, A, \Delta, x_0, X_m)$, two languages can be associated to it:

- the **generated language**, i.e, the set of all **generated** words:

$$L(G) = \{w \in A^* \mid \text{ there exists } x \in X : (x_0, w, x) \in \Delta^*\} \subseteq A^*$$

- the **accepted language**, i.e, the set of all **accepted** words:

$$L_m(G) = \{w \in A^* \mid \text{ there exists } x \in X_m : (x_0, w, x) \in \Delta^*\} \subseteq L(G).$$

2.5.3 Observer of a partially observable system: Models and Constructions

The usual method to describe what an external observer can see in a given NFA is to assume that it can only observe a subset $A_o (A_o \subseteq A)$ of the events. This notation divides the events into observable and unobservable, and the remaining unobservable are denoted by $A_{uo} := A - A_o$.
In order to map words executed in the system to the sequence of observations associated with it, the natural projection $P_o : A^* \to A_o^*$ can be used. It is defined recursively as $P_o(s\sigma = P_o(s)P_o(\sigma)$, for all $s \in A^*$ and $\sigma \in A$, with

$$P_o(\sigma) = \begin{cases} \sigma, & \text{if } \sigma \in A_o, \\ \epsilon, & \text{if } \sigma \in A_{uo} \cup \{\epsilon\}, \end{cases}$$

where ϵ represents the empty observation.

For a given language $L \subseteq E^*$, it is defined that $P_o(L) := \{P_o(s) \mid s \in L\}$, and the inverse is also defined, the image map of P_o as $P_o^{-1} : A_o^* \to 2^{A^*}$, where,

$$(\forall t \in A_o^*) P_o^{-1}(t) := \{s \in A^* \mid P_0(s) = t\}.$$

Given language $L \subseteq A_o^*$, it is defined that $P_o^{-1}(L) := \bigcup_{s \in L} P_o^{-1}(s)$.

Given NFA $G = (X, A, \delta, x_0, X_m)$, and after the occurrence of word s has happened in the system, the observer records the word $w = P_o(s)$, and based on w, there are a variety of estimation tasks that the observer might partake in, in which, they may learn about the system, for example, by estimating the current-state or initial-state of the system. Given word w, the set of possible current states for system G, is given by:

$$\hat{X}(w) = \{x \in X \mid (\exists s \in A^*)(\exists x_0 \in X_0) x \in \delta(x_0, s) \wedge P_o(s) = w\}$$

On the other hand, the set of potential initial states for system G, after observing w, is given by:

$$\hat{X}_0(w) = \{x_0 \in X_0 \mid (\exists s \in A^*) \delta(x_0, s) \neq \emptyset \wedge P_o(s) = w\}$$

Other techniques to obtain systematically estimator constructions for current and initial-state estimation can be found in [Cassandras and Lafortune, 2010] and [Saboori and Hadjicostis, 2013].

The observer, also referred to as current-state estimator for the previously given NFA G under a natural projection map P_o with respect to alphabet A, that can be divided into a set of observable events $A_o, A_o \in A$. The observer automaton $G_{obs} = (X_{obs}, A_o, \delta_{obs}, X_{0,obs})$ is a DFA where:

1. $X_{obs} \subseteq 2^X$, meaning that the states of G_{obs} are a subset of X.
2. $X_{0,obs} = \{x \in X \mid (\exists t \in A_{uo}^*) x \in \delta(X_0, t)\}$, returns the initial state, i.e, $X_{0,obs}$ is the set of states of G that are reachable from an initial-state in X_0 via zero, or more unobservable events.

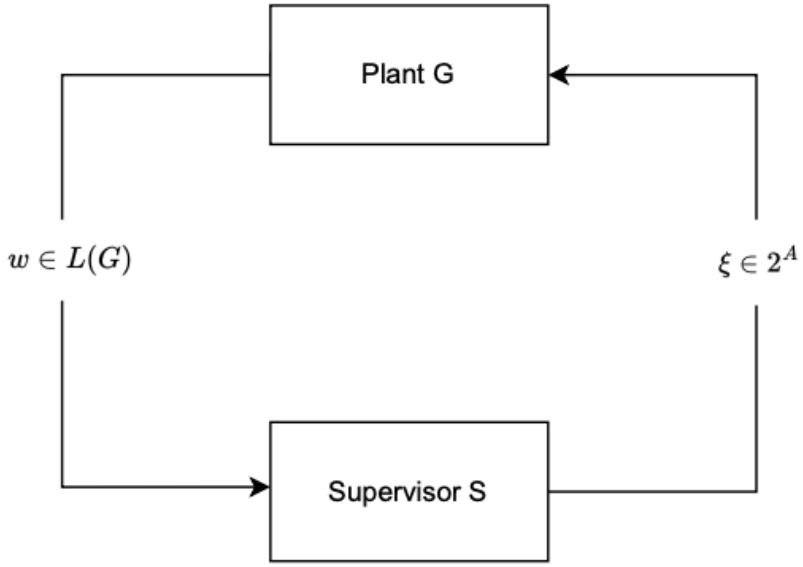


Figure 2.4: Plant G controlled by supervisor S

3. For $x_{obs} \in X_{obs}$ and $\sigma_o \in A_o$, the transition function δ_{obs} is defined as $x'_{obs} = \delta_{obs}(x_{obs}, \sigma_o)$, with

$$x'_{obs} = \{x' \in X \mid (\exists s \in A^*) x' \in \delta(x_{obs}, s) \wedge P_o(s) = \sigma_o\}.$$

X_{obs} is usually limited to be the set of states in G_{obs} that are reachable from the initial state $X0, obs$. If x'_{obs} is the empty set, then it is not included in the construction of G_{obs}, i.e, if $\delta(x_{obs}, \sigma_o) = \emptyset$, then $\delta_{obs}(x_{obs}, \sigma_o)$ is undefined.

A fundamental property property of the observer automaton is that given any sequence of events that occurred in G, $s \in L(G)$, and generated the observed word $w = P_o(s), w \in A_o^*$, it is considered that,

$$\hat{X}(w) = \delta_{obs}(X_{0,obs}, w)$$

, i.e, given the observation of word w, the set of possible current states of G is captured by the state reached in the observer, starting from the initial state and applying word w.

2.6 Supervisory Control

This section is created to focus on the main theoretical topics of *Supervisory control*, since these concepts are used for some studies in the field of opacity.
This concept originated from the work [Ramadge and Wonham, 1989], in which the authors define a framework for the control of logical DES. Following some of the concepts, in supervisory control theory, a *plant*, i.e, a system that is going to

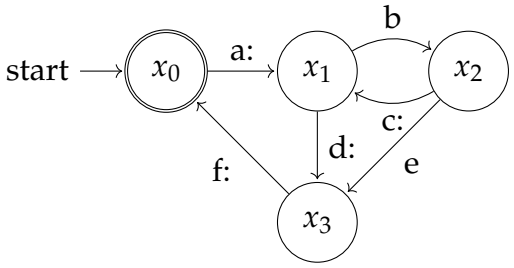


Figure 2.5: DFA representing a plant G

be controlled, is considered to be guided by a control agent called a *supervisor*. The *closed loop system*, i.e, the plant subjected to the control of the supervisor, is represented in Figure 2.4.
The supervisor oversees the events generated by the plant and influences its evolution by selectively preventing certain events. it is important to note that, in this framework, the supervisor has the capability to *limit* the plant's behavior but lacks the authority to *expand* it. In other words, the closed-loop system is constrained to produce only words that were initially within the language of the plant.

2.6.1 Components - Plant, supervisor and closed-loop system

In supervisory control theory, a *plant* G is a system that's going to be controlled by a supervisor. The behaviour of it can be described by a sequence of events it can generate, i.e **language**, that are defined on an alphabet A. The alphabet of the plant is partitioned into *controllable*, A_c, and *uncontrollable*, A_{uc}, events alphabets.

$$A = A_c \cup A_{uc} \text{ (with) } A_c \cap A_{uc} = \emptyset$$

A controllable event $\sigma \in A_c$ signifies an action that can be externally disabled, for instance, through a controller. Conversely, an uncontrollable event $\sigma' \in A_{uc}$ denotes the opposite—an event whose occurrence cannot be disabled by any controller.

A practical example, can be considered with plant G, represented in Figure 2.5, in which the character ':' identifies a controllable event. In this case, the alphabet $A = \{a, b, c, d, e, f\}$ is divided into $A_c = \{a, b, d, f\}$ and $A_{uc} = \{b, e\}$
This DFA represents a coffee machine that, when in idle (initial state x_0), can be activated (event a), indicating that it is ready to brew coffee (state x_1). From this state, the user can initiate the brewing process (event b), leading to the brewing state (x_2), or they can switch the machine off (event d), transitioning to the off state (x_3).
When the coffee machine is actively brewing (state x_2), the operation may conclude successfully (event c), returning the machine to the ready state (x_1). Alternatively, a brewing error may occur (event e), autonomously causing the machine to shut off. Performing maintenance (event f) brings the machine back from the off state to the idle state. In this DFA, the final state coincides with the initial one,

signifying that after one or more brewing cycles, the machine should return to the idle state.
Regarding the sets of controllable and uncontrollable events, they carry clear physical meanings. For instance, event a denotes the activation of the coffee machine, a user-initiated action that could be disabled by a supervisor if needed. Similarly, events b (start brewing), d (switch off), and f (maintenance) are controllable. In contrast, event e represents a brewing fault, an undesirable event beyond the supervisor's control. Similarly, it is assumed that event c is uncontrollable, as once the brewing process has started, there is no control action to prevent its successful completion.

As it can be seen in Figure 2.4, it is assumed that the supervisor can observe the word of events $w \in L(G)$, generated by plant G. A general definition of active events, when DFA $G = (X, A, \delta, x_0, X_m)$, can be divided into:

- For every state $x \in X$, the set of active events in G **at** state x is

$$\mathcal{A}_G(x) = \{s \in A \mid \delta(x, s) \text{ is defined } \}.$$

- For every word $w \in L(G)$ the set of active events in G after w is

$$\mathcal{A}_G(w) = \{e \in A \mid we \in L(G)\}.$$

For the sake of simplicity, the same notation $\mathcal{A}_G$ is used two indicate two different functions: a mapping $\mathcal{A}_G : X \to 2^A$ and the another mapping $\mathcal{A}_G : L(G) \to 2^A$. For a DFA G, if state $x = \delta^*(x_0, w)$ is reached from initial state x_0, generating word w, then it holds that $\mathcal{A}_G(x) = \mathcal{A}_G(w)$.

After word $w \in L(G)$ is generated by the plant, an active controllable event $e \in \mathcal{A}_G(w) \cap A_c$ can be *disabled* by the supervisor, whereas, when it comes to uncontrollable event, the supervisor has no way of preventing the occurrence of an active uncontrollable event $e' \in \mathcal{A}(w) \cap A_u c$. An event is called *enabled*, when the supervisor doesn't disable it.

A plant's progression is steered by a supervisor as it generates the control inputs. Given plant G defined on alphabet A, a control input is a subset of events $\xi \subseteq A$, 2^A is denoted as the set of all possible control inputs.
It is also important to note that, if event $e \in \xi$, then e is enabled by the supervisor, whereas if $e \notin \xi$, e is considered disabled. Due to the fact that uncontrollable events can't be disabled by the supervisor, considering a plant G and word $w \in L(G)$, a control input ξ, is called *admissible after* w if $A_{uc} \cap \mathcal{A}_G(w) \subseteq \xi$, i.e, all uncontrollable events that are active in G after w are contained in it.
For **example**, considering the plant in Figure 2.5, the control input $\xi = \{a, b, c\}$ is admissible after event a since $\mathcal{A}(a) = \{b, d\}$, meaning that, among all active events, only controllable event d is disabled. If the same control input is considered after ab, then it won't be permissible since $\mathcal{A}_G(ab) = \{c, e\}$, and consequently it disables the uncontrollable active event e.

Combining the earlier concepts, it is now possible to provide a precise definition for a supervisor.

Given plant G, the supervisor s controlling it, can be represented by the control function

$$f : L(G) \to 2^A,$$

which generates a sequence of admissible control inputs $\xi_0, \xi_1, \xi_2, \ldots \subseteq A$

$$\xi_0 = f(\epsilon),\ \xi_1 = f(e_1),\ \xi_2 = f(e_1e_2),\ \ldots$$

in response to the generated sequence of events $w = e_1e_2 \ldots \in L(G)$ produced by the plant.

In this definition the assumption that for all word w generated by the plant, the control input $f(w)$ that is produced by the supervisor is admissible after word w.

Considering the scenario depicted in Figure 2.5. The supervisor's objective is to ensure a good brewing process, allowing at most one brewing operation each time while the machine is activated. In Table 2.2, the control function $(f(w)$ of a supervisor designed to enforce this desired behavior, is illustrated.
Examining the table, it is evident that initially, the supervisor disables no event, meaning, $f(\epsilon) = f(a) = E$. However, once the sequence ab is observed, indicating the start of brewing a coffee, the supervisor intervenes. It effectively disables event b until a new activation a is registered. Consequently, $f(ab) = f(abc) = f(abcd) = f(abcdf) = E \setminus \{b\}$, and $f(abcda) = E$. This example will be marked with $*$, since it will be reference in the beginning of 2.6.2.

2.6.2 Representing a Supervisor as a DES and closed-loop system

Previously, the supervisor was defined as a function $f : L(G) \to (2^A)$. In this subsection, it will be represented as a DES, i.e, as a DFA $S = (\hat{X}, A, \hat{\delta}, \hat{x}_0, \hat{X}_m)$, using the same alphabet that is defined for the plant.
This describes the operational process of a DFA supervisor, where, upon each event generated by the plant, the supervisor executes the same event. When the supervisor is in a state $\hat{x}$, it sends ti the plant a control input $\mathcal{A}_S(\hat{x})$ which contains all active events in S from state $\hat{x}$.
The following procedure, describes the evolution of the closed-loop system. A plant $G = (X; A, \delta, x_0, X_m)$ that is controlled by the supervisor $S = (\hat{X}, A, \hat{\delta}, \hat{x}_0, \hat{X}_m)$, is given

1. At the start, plant G is at state $x = x_0$, and the supervisor in state $\hat{x} = \hat{x}_0$.
2. Considering $w = \epsilon$.
3. The control input produced by the supervisor is $\xi_w = \mathcal{A}_S(\hat{x})$.
4. An event $e \in \mathcal{A}_G(x) \cap \xi_w$ is generated by the plant, and it goes to state $x' = \delta(x, e)$.
5. The supervisor executes e and goes to state $x' = \hat{\delta}(\hat{x}, e)$.

w	x	$\xi = f(w)$
ε	x_0	E
a	x_1	E
ab	x_2	$E \backslash \{b\}$
abc	x_1	$E \backslash \{b\}$
$abcd$	x_3	$E \backslash \{b\}$
$abcdf$	x_0	$E \backslash \{b\}$
$abcda$	x_1	E
...	...	...
ad	x_3	E
adf	x_0	E
$adfa$	x_1	E
...	...	...

Table 2.2: Control function for the example

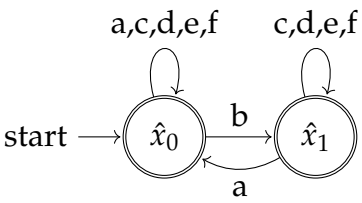


Figure 2.6: DFA representing the Supervisor for the plant in Figure 2.5

6. Considering $w = we, x = x', \hat{x} = \hat{x}'$
7. Repeat step 3.

An example to clarify the procedure is, for **example**, considering the Supervisor in the last example of 2.6.1 marked with $*$, can also be represented as a DFA in alphabet A, as it can be seen in Figure 2.6. There are two states in the DFA, state $\hat{x}_0$ that has all events active (enabled), and as soon as an event b occurs, S moves to state $\hat{x}_1$, where event b is not active (disabled). When event a occurs, S is brought back to state $\hat{x}_0$.

To demonstrate the equivalence of supervisor S to the one described by Table 2.2, Table 2.3 displays the control inputs produced by S. For every word generated by the plant under supervision, the state $\hat{x}$ reached by S is shown along with the set of active events that determine the control input. A simple comparison of the two tables confirms that they describe the same control law.

w	x	$\hat{x} = \hat{\delta}(\hat{x}_0, w)$	$\xi = \mathcal{A}_S(\hat{x})$
ε	x_0	$\hat{x}_0$	E
a	x_1	$\hat{x}_0$	E
ab	x_2	$\hat{x}_1$	$E\backslash\{b\}$
abc	x_1	$\hat{x}_1$	$E\backslash\{b\}$
$abcd$	x_3	$\hat{x}_1$	$E\backslash\{b\}$
$abcdf$	x_0	$\hat{x}_1$	$E\backslash\{b\}$
$abcda$	x_1	$\hat{x}_0$	E
$\dots$	$\dots$	$\dots$	$\dots$
ad	x_3	$\hat{x}_0$	E
adf	x_0	$\hat{x}_0$	E
$adfa$	x_1	$\hat{x}_0$	E
$\dots$	$\dots$	$\dots$	$\dots$

Table 2.3: Control function for S

An inherent advantage of representing a supervisor as a DFA is the immediate determination, in this case, of the DFA representing the closed-loop system. Following the procedure described before, it becomes evident that a word w generated by the closed-loop system is both a word of G (as it is generated by the system) and a word of S (since at each step, the generated event belongs to the control input and is consequently active in S).

Another definition to consider is how to construct the automaton of the *closed-loop system*, of a plant G controlled by a supervisor S. It is the automaton $S/G = G \cap S = G \| S$, whose closed language is $L(S/G) = L(G) \cap L(S)$ and the marked language is $L(S/G) = L(G) \cap L(S)$.

When a DFA supervisor is considered, there are two possibilities, a *non-marking supervisor*, and a *marking supervisor*.

- A **non-marking supervisor** is when the supervisor doesn't specify which are the final words, thus making all words that supply the a final state in the plant are considered final. In this case, it is assumed that the set of final states of the supervisor is $\hat{X}_m = \hat{X}$, i.e, all its states are final, and thus the marked language of the closed loop system is,

$$L_m(S/G) = L_m(G) \cap L_m(S) = L_m(G) \cap L(S)$$

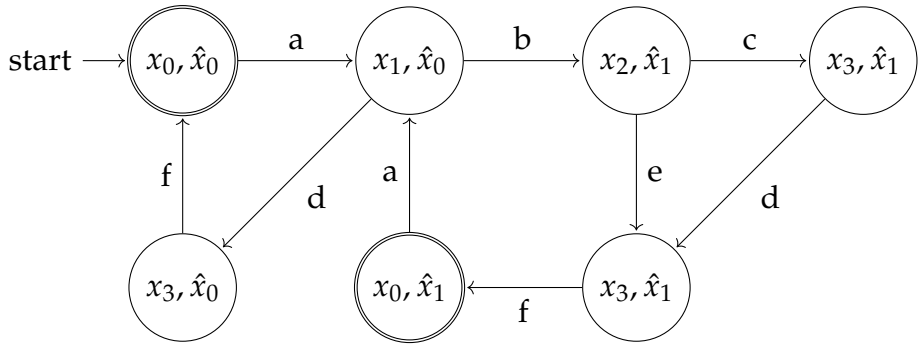


Figure 2.7: The closed-loop system

in essence, it contains all words that are accepted by the plant and survive under supervision.

- A **marking supervisor** specifies which words are final, meaning that a word is final if it produces a final state on both the plant and the supervisor. In this case, it is assumed that the set of final states of the supervisor is $\hat{X}_m \subsetneq \hat{X}$, i.e, not all states are final.

An example to be considered is regarding plant G in Figure 2.5, controlled by supervisor in Figure 2.6, the *closed-loop* system S/G constructed by concurrent composition can be see in Figure 2.7. In this case a supervisor that is non-marking is considered: the marked states in the closed-loop systems are all states $(x, \hat{x})$ where $x \in X_m$, i.e, $(x_0, \hat{x}_0)$ and $(x_0, \hat{x}_1)$.

2.7 Opacity in DES

2.7.1 Introduction

Opacity, a confidentiality property, has gained prominence in analysing various partially observed DES. It provides a systematic approach to capturing, verifying, and, when needed, enforcing critical aspects of security and privacy within applications of these systems. Since its inception less than two decades ago, opacity has become essential for addressing security and privacy concerns in DES applications.
According to [Lafortune et al., 2018], the introduction of the concept of opacity in computer systems literature by [Mazaré, 2004]. Opacity was initially applied to investigate security and privacy in information flow within cryptographic protocols. Subsequent research built upon [Mazaré, 2004] work, with papers employing Petri nets [Bryans et al., 2005] and transition systems [Bryans et al., 2008] as modeling formalisms. Concurrently, European researchers introduced the concept of enforcing "concurrent secrets" in the dynamics of discrete event systems in 2007 [Badouel et al., 2007].

In this section, some of the predominant formulations of opacity are reviewed, along with the verification and enforcement of it.

In its fundamental essence, opacity seeks to characterize the extent of inferences and conclusions that a passive observer, one that lacks the ability to influence system operations, can derive concerning a designated subset of a given DES behavior. Typically, this subset, known as the system's secret behavior, is encapsulated by a predicate with a binary outcome, evaluating either to be true or false. In this formulation, the core question posed by opacity pertains to whether specific activities within the system can generate observations enabling an external observer to affirm the truth of the predicate. If such an assertion becomes possible, it is declared that *the secret is revealed*, indicating a violation of opacity.

The passive external observer, while restricted to acquiring observations generated by system activity, may possess partial knowledge of the system model. This knowledge serves as a basis for the observer to assess, based on the gathered observations, whether the predicate holds true. It is crucial to note that the external observer's role is solely observational and devoid of any capacity to influence the system's functioning.

Opacity, as an information flow property, holds significant implications for privacy and security. Broadly, a system achieves opacity when certain secrets remain undisclosed throughout its operation.

This dissertation centers around the exploration of *logical* opacity, with Chapter 2 dedicated to an in-depth examination of this specific aspect. However, it is essential to acknowledge the existence of another variant known as *probabilistic* opacity formulations. In these formulations, various system behaviors may be linked to distinct probabilities, suggesting that external observers may also be intrigued by scenarios where the concealed information can be deduced with a high level of certainty.

2.7.2 Concept of Opacity

For this subsection, it will be assumed that the system in which the concepts will be utilized is a, usually nondeterministic, finite automaton $G = (X, A, \delta, X_0)$, under the natural projection map P_o with respect to an alphabet with observable events A_o, $A_o \subseteq A$, as seen in 2.5.3. As previously mentioned in subsection 2.7.1, this concept can be applied in other systems under partial observation, such as Petri nets, and others.

Opacity can be classified into *state-based* and *language-based*, and the classification relies on the nature of the considered secret behavior. In the case of state-based opacity, an individual is provided with a subset of states S, $S \subset X$, and the core investigation is focused on determining if there exists a word, w, $w \in L(G)$, such that the sequence of observations generated by it, $w_o = P_o(w)$ reveals to the external observer that at a certain point, the state in which the system lies is in S, the set of secret states. For instance, the scenario of current-state opacity, in which the analysis revolves around whether the set of potential system states subsequent to the observation of w_o creates a subset of S (i.e., if $\hat{X}(w_o) \subseteq S$). Should this condition be true, is is designated that w a violation of current-state opacity occurs upon the occurrence of w.

In a like manner, if the set of possible initial states, after the observation of w_o is a subset of S (i.e., if $\hat{X}(w_o) \subseteq S$), then there is a violation of initial-state opacity.

Regarding language-based opacity, a secret language $L_S \subseteq L(G_n)$ is provided, and the core investigation is focused on determining if there exists a word w, $w \in L(G)$, in a way that the observation that it generates, $w_o = P_o(w)$, enables the external observer to conclude that system generated a word that lies in the secret language L_S, meaning, the set,

$$P_{o,L}^{-1} := \{s \in L(G) \mid P_o(s) = w\}$$

satisfies $P_{o,L}^{-1} \subseteq L_S$.

Opacity can be categorized into strong and weak formulations. In the context of strong opacity, it necessitates that every behaviour within the system produces a sequence of observations that does not result in a violation of opacity. On the other hand, weak opacity mandates that some, though not necessarily all, system behaviours should generate a sequence of observations that avoids violating opacity. it is important to note that while weak opacity is encompassed by strong opacity, the reverse is not necessarily true. In stochastic systems, one can further enhance the definition of weak opacity by specifying the a priori probability associated with the likelihood of the system-producing behaviour leading to an opacity violation. In this section, the notions of opacity will all be considered in a strong manner.

State-Based Opacity

In this section of the work, the focus lies on exploring the definitions of current and initial-state opacity, as these represent the most prevalent applications within the realm of opacity.

Considering an NFA $G = (X, A, \delta, X_0)$ under a natural projection map P_o with respect to an alphabet of observable events A_o, $A_o \subseteq A$. Considering S as a set of secret states, $S \subset X$, G is

- **current-state opaque** if for all $w \in L(G)$ it holds that

$$\hat{X}(P_o(s)) \nsubseteq S;$$

- **initial-state opaque** if for all $w \in L(G)$ it holds that

$$\hat{X}_o(P_o(s)) \nsubseteq S;$$

it is important to remember that $\hat{X}(w)$ was defined in subsection 2.5.3. Therefore, for the definition of current-state opacity it is required that, for each $w \in L(G)$, it is possible to find a $t \in L(G)$, such that,

$$\{\delta(X_0, t) \nsubseteq S\} \wedge P_o(t) = P_o(s)$$

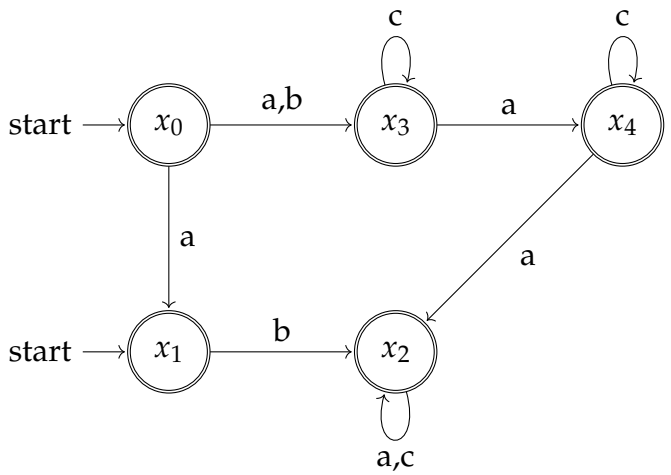


Figure 2.8: NFA G considered for examples in the section

In a like manner, for the case of initial-state opacity, it is required that, for each $s \in L(G)$, it is possible to find a $t \in L(G)$, such that,

$$\{\delta(X_{0,NS}, t) \neq \emptyset\} \wedge P_o(t) = P_o(s)$$

having $X_{0,NS} := X_0 - S$ as the set of initial states that are not secret. For initial-state opacity, it may considered, without loss of generality, to define the secret set as $S_0 := S \cap X_0$. If S_0 is either empty or identical to X_0, then the initial-state opacity problem becomes trivial.

Considering a NFA $G = (X, A, \delta, X_0)$ in Figure 2.8, with $X = \{x_0, x_1, x_2, x_3, x_4\}$, alphabet $A = \{a, b, c\}$, the transition function that can be seen in figure, and the set of initial states $X_0 = \{x_0, x_1\}$. It is assumed that the alphabet of observable events $A_o = \{a, b, c\}$, meaning that, $A_o = A$.

To better understand current-state opacity, three different cases can be considered:

- **Case 1**: Assuming the set of secret states is $S_1 = \{x_4\}$, various scenarios exist in which current-state opacity is violated. For instance, words like *aa* or *aca* enable the external observer to infer that $(\hat{X}(\text{aa}) = \hat{X}(\text{aca}) = \{x_4\})$ (more generally, $\hat{X}(\text{ac}^*\text{ac}^*) = \{x_4\}$. Consequently, it is concluded that G is not current-state opaque with S_1.

- **Case 2**: Considering the scenario where the set of secret states is $S_2 = \{x_3\}$, once again, there exist multiple situations in which current-state opacity is violated. For instance, words in the form of acc^* allow the external observer to deduce that $\hat{X}(\text{acc}^*) = \{x_3\}$. As a result, it is concluded that G is not current-state opaque with respect to S_2

- **Case 3**: Now, if the set of secret states is $S_3\{x_1\}$, there are no identified sequences that permit the external observer to ascertain that the system is in state 2 (a systematic demonstration of this is done in the next subsection). Consequently, it is concluded that G is current-state opaque concerning S_3.

The same can be done for the concept of initial-state opacity:

- **Case 1**: Assuming the set of secret initial states is $S_1 = \{x_4\}$. In this case, there is no violation of initial-state opacity since the set of initial states $X_0 = \{x_0, x_1\}$ does not include state x_4. Consequently, it is concluded that G is initial-state opaque with respect to S_1.
- **Case 2**: Now, considering the scenario where the set of secret initial states is $S_2 = \{x_0\}$. Under various scenarios, the external observer can be certain that the initial state was state $\{x_0\}$. For instance, words like ac, ab, and aa can only be initiated from state x_0 (but not state x_1), i.e., $\hat{X}_0(\text{ac}) = \hat{X}_0(\text{ab}) = \hat{X}_0(\text{aa}) = \{x_0\}$. Actually, if the first observation is a then the external observer will know that the initial state is state $\{x_0\}$. Consequently, it is concluded that G_n is not initial-state opaque with respect to S_2.
- **Case 3**: Now, let's assume the set of secret initial states is $S_3 = \{x_1\}$. In this case, no sequence of observations allows the external observer to conclude that the system's initial state was x_1. Any word initiated from state x_1 is of the form $b(a+c)^*$ and can also be generated starting from state x_0, i.e., $\hat{X}_0(b(a+c)^*) = \{x_0, x_1\}$. Consequently, it is concluded that G is initial-state opaque with respect to S_3 (systematic verification of initial-state opacity is discussed in the next subsection).

There are other notions of state-based opacity that can be considered, but as these fall out of the scope of this dissertation, these won't be explored, but some references for works exploring them are given. These notions are K-step opacity, it can be explored in [Falcone and Marchand, 2015] and [Saboori and Hadjicostis, 2011a], infinite-step opacity, [Saboori and Hadjicostis, 2011b].

Language-Based Opacity

Considering an NFA $G = (X, A, \delta, X_0)$ under a natural projection map P_o with respect to an alphabet of observable events A_o, $A_o \subseteq A$. Given a secret language L_S, $L_S \subset L(G)$, it is said that language-based opacity is not violated for G, if for all $t \in L_S$, there is another word $w \in (L(G) - L_S)$, such that $P_o(t) = P_o(w)$. Put differently, for the external observer any secret sequence of events $t \in L_S$ seems is identical to the appearance of at least one other non-secret sequence of events $w \in (L(G) - L_S)$.

2.7.3 Verification of Opacity

The validation of different aspects of state-based opacity relies on the assessment of whether the estimations of possible states at specific time points possess a particular undesirable characteristic, at least under certain system behaviors.
For example, in the context of current-state opacity, the focus is on determining whether the set of potential current states of the system (for any given sequence of observations) is a subset of the secret states S. Notably, the observer automaton,

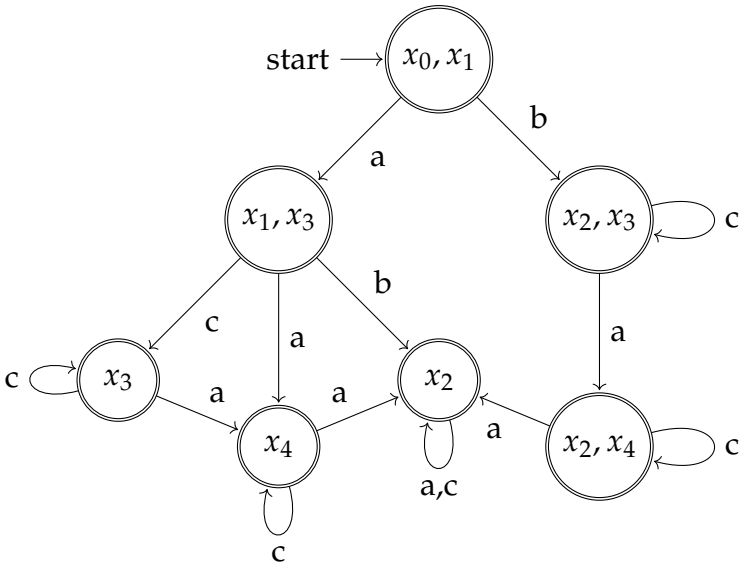


Figure 2.9: Observer G_{obs} of G considered for examples in the section

detailed later in this subsection, serves as a systematic tool for tracing the potential current states of the system based on any sequence of observations, playing a pivotal role in verifying the notion of opacity.

As mentioned in 2.5.3, the construction of the observer for a given NFA G ensures the property that any word $w_o = P_o(w)$, $w_o \in A_o^*$, it is true that,

$$\hat{X}(w) = \delta_{obs}(X_{0,obs}, w).$$

Utilizing the mentioned characteristic of the observer, it is easy to verify current-state opacity for a given system G, since:
Supposing that there exists a state x_{obs} of the observer that,

1. is non-empty,
2. is reachable from $X_{0,obs}$,
3. satisfies $x_{obs} \subseteq S$;

then, the system can't be current-state opaque, and current-state opaque if no such state can be found.

Considering again the NFA $G = (X, A, \delta, X_0)$ in Figure 2.8, with $X = \{x_0, x_1, x_2, x_3, x_4\}$, alphabet $A = \{a, b, c\}$, the transition function that can be seen in figure, and the set of initial states $X_0 = \{x_0, x_1\}$. As for the previous example, it is considered that the alphabet of observable events $A_o = \{a, b, c\}$, meaning that, $A_o = A$. The first step is to build the observer G_{obs} for G_n, following the procedure already mentioned, and it can be observed in Figure 2.9. It is of the form, $G_{obs} = (X_{obs}, A_o, \delta_{obs}, X_{0,obs})$, where,

$$X_{obs} = \{\{x_0, x_1\}, \{x_1, x_3\}, \{x_2\}, \{x_4\}, \{x_3\}, \{x_2, x_3\}, \{x_2, x_4\}\}$$

$X_{0,obs} = x_1, x_2$ and δ_{obs} is as observed in the figure.

Considering again the three different cases for current-state opacity in the previous example,

1. **Case 1**: Assuming the set of secret states is $S_1 = \{x_4\}$. Owing to the presence of state $\{x_4\}$ in the observer and the fact that this state is reachable from the initial state via at least one word (e.g., *aa*, *aca*), it is concluded that G is not current-state opaque with respect to S_1.

2. **Case 2**: Now, let's consider the scenario where the set of secret states is $S_2 = \{x_3\}$. Again, due to the presence of state $\{x_3\}$ in the observer and the fact that this state is reachable from the initial state via at least one sequence of observations (e.g., *ac*, *acc*), it is concluded that G is not current-state opaque with respect to S_2.

3. **Case 3**: Now, let's assume the set of secret states is $S_3 = \{x_1\}$. In this case, no (non-empty) state in the observer is found to be a subset of S_3 (i.e., there is no $\{x_1\}$ state), and it is concluded that G is current-state opaque with respect to S_3.

2.7.4 Enforcing Opacity

In this section, approaches proposed to enforce opacity in situations where certain system behaviors violate it are described. Specifically, the section explores strategies based on supervisory control to minimally restrict and disable system behavior that leads to opacity violations.

Enforcing through Supervisory Control

For this part of the sub-subsection of the chapter, the enforcement of current-state opacity by using supervisory control strategies is described. Considering a NFA $G = (X, A, \delta, X_0)$, under a natural projection map P_o with respect to an alphabet of observable events A_o, $A_o \subseteq A$, and assuming a basic supervisory control setting where a certain subset of events A_c, $A_c \subseteq A$, are referred as controllable events and can be disabled ate any given time, by the supervisory control system. To be more precise, alphabet A can be divided into an alphabet of controllable events A_c and an alphabet of uncontrollable events A_{uc}, such that $A_c \cap A_{uc} = \emptyset$ and $A_c \cup A_{uc} = A$. it is also assumed that controllable events are observable, meaning that $A_c \subseteq A_o$.
With the previously outlined setup, it becomes evident that for the supervisory control system to effectively enforce current-state opacity, the system, obtained by constraining the behavior of G to uncontrollable events, for example, by having the supervisor disable all controllable events), must exhibit current-state opacity. Considering system $G' = (X, A_{uc}, \delta', X_0)$, where $\delta' : X \times A_{uc} \to 2^X$ is a restriction of δ to A_{uc}, meaning that, for all $x \in X$ and all $\sigma_{uc} \in A_{uc}$, it holds that,

$$\delta'(x, \sigma_{uc}) = \delta(x, \sigma_{uc}).$$

Additionally, it is assumed that G' is observed under the natural projection $P_{A'_o}$ in which the set of observable events is $A'_o = A_o \cap A_{uc}$.
Given the assumptions, it becomes clear that if system G' is not current-state opaque, then no supervisory control strategy will be able to enforce it for the system, since, if G'_{obs} is considered for G', and G' is not current-state opaque, then the observer contains a reachable state x'_{obs} that $x'_{obs} \subseteq S$. Evidently, state x'_{obs} is also in the observer G_{obs} of G under any supervisory control strategies since it is reachable from uncontrollable events. In this instance, the condition for verifying the existence of a supervisory control strategy enabling the enforcement of current-state opacity for a system G is both necessary and sufficient.
As a strategy that disables all controllable events at all times may be overly restrictive, the challenge that arises is if it is possible to create a supervisory control strategy that disables events only when absolutely necessary, thus maintaining minimal restrictiveness.

A minimally restrictive supervisory control strategy can be created, since a supervisor has access to all observable events, and can obtain an estimate of the system state based on the observations that are generated by the system. Considering $G_{obs} = (X_{obs}, A_o, \delta_{obs}, X_{0,obs})$ of system G, this estimate can be obtained, though it needs to be considered that any reachable state $x_{obs} \in X_{obs}$ that holds $x_{obs} \subseteq S$ means that there is a violation of current-state opacity, and must be avoided. The simplest way to deal with such state is to verify all transitions that go to said state and disable them, if possible, the following cases are considered:

- Assuming that $\delta_{obs}(x'_{obs}, \sigma_c) = x_{obs}$ for a controllable, thus observable, event $\sigma_c \in A_c$. This scenario outlines a situation in which, by having the supervisor disable event σ_c whenever the system is at state x'_{obs}, it is possible to avoid state x_{obs}.
- Given that $\delta_{obs}(x'_{obs}, \sigma_{uc}) = x_{obs}$ for uncontrollable event σ_{uc} that $\sigma_{uc} \in (A_{uc} \cap A_o)$. In this case, since event σ_{uc} can't be disabled, the solution is to altogether avoid reaching state x'_{obs}. To clarify, although state x'_{obs} doesn't violate current-state opacity, as uncontrollable event σ_{uc} can lead the system from this state to a state that does, state x'_{obs} has to be avoided.

The preceding remarks imply that it is possible to disable transitions to states that have to be avoided, until all of the paths that lead to them have been eliminated. When this is not possible to do, it means that G', previously defined as a restriction of system G to uncontrollable events, is not current-state opaque, implying that no supervisory control strategy can be applied to enforce this type of opacity.

An **example** to demonstrate how the methodology described before to obtain a minimally restrictive supervisory control strategy is, if NFA $G = (X, A, \delta, X_0)$ from 2.8, with $X = \{x_0, x_1, x_2, x_3, x_4\}$, alphabet $A = \{a, b, c\}$, the transition function that can be seen in figure, and the set of initial states $X_0 = \{x_0, x_1\}$. It is assumed that the alphabet of observable events $A_o = \{a, b, c\}$, meaning that, $A_o = A$. The observer is in Figure 2.9. It is of the form, $G_{obs} = (X_{obs}, A_o, \delta_{obs}, X_{0,obs})$, where,

$$X_{obs} = \{\{x_0, x_1\}, \{x_1, x_3\}, \{x_2\}, \{x_4\}, \{x_3\}, \{x_2, x_3\}, \{x_2, x_4\}\}$$

$X_{0,obs} = x_1, x_2$ and δ_{obs} is as observed in the figure. Several cases for the alphabet of controllable events, A_c, and set of secret states S, can be considered:

1. **Case 1**: Assuming $A_c = A_o = \{a, b, c\}$ and $S = \{x_3\}$. In this scenario, one supervisory control strategy to enforce opacity would be to disable event c at state $\{x_1, x_3\}$ of the observer. Note that this strategy observes all events in A_o and, if it observes a, disables c until a or b occurs. This ensures that state $\{x_3\}$ of the observer is not visited, thereby preventing a violation of current-state opacity. This would be a maximally permissive supervisory control strategy. Another strategy (not maximally permissive) could be to disable a at startup until b occurs.

2. **Case 2**: Assuming $A_c = A_o = \{a, b, c\}$ and $S = \{x_4\}$. In this case, the maximally permissive supervisory control strategy to enforce current-state opacity would be to disable event a at states $\{x_1, x_3\}$ and $\{x_3\}$ of the observer. Another strategy (not maximally permissive) might involve disabling event a at startup.

3. **Case 3**: Assuming $A_c = \{a, b\}$ (i.e., event c is uncontrollable) and $S = \{x_3\}$. In this scenario, the maximally permissive supervisory control strategy to enforce current-state opacity would be to disable event a at observer state $\{x_0, x_1\}$. This is because if a occurs, it leads to the state $\{x_1, x_3\}$, which is weakly forbidden since it can reach the forbidden state $\{x_3\}$ via an uncontrollable event.

Determining which events to disable at each observer state to enforce current-state opacity may need multiple iterations, especially when dealing with situations involving the disablement of an event leading to blocking. Blocking refers to a scenario where no events are permitted to occur at a particular state. In such cases, it becomes imperative to ensure that the state deemed blocking is not visited, and this is achieved by strategically disabling events prior to reaching this state. The need for several iterations becomes apparent in resolving these intricate scenarios, contrasting with the simpler example provided earlier.

2.8 Summary

This chapter explores key concepts essential for understanding the background of the dissertation. The exploration begins with an examination of DES and their characteristics. Formal languages and Finite Automata are introduced, covering elements such as alphabets, words, and operators. DFAs are defined, and their languages and modeling applications are discussed.

The study extends to NFAs, delving into their definition, languages, and the observation of partially observable systems. Supervisory Control is introduced, breaking down its components—plant, supervisor, and closed-loop system—while representing a supervisor as a DES and closed-loop system.

Opacity in DES is a key focus of this work. The concept is initially explored, covering both state-based and language-based notions. Following this, verification and enforcement methods are explored. This exploration establishes the foundation for understanding the significance of Opacity within the context of Discrete Event Systems, serving as the basis for comprehending the rest of the work.

Chapter 3

State of the Art

3.1 Introduction

This dissertation delves into security within the context of DES. Furthermore, it explores the concept of opacity, adding a unique perspective to its application in these systems. As detailed in Chapter 2, the adversary in a DES employing opacity is passive. This implies that the attacker cannot tamper with or influence system operations in any manner.

The primary focus of this work is to investigate how a system, aiming to maintain opacity, responds when subjected to active attackers. Through the development of a technique, the goal is to enhance the system's security against these malicious actors while preserving the core concept of opacity.

This section aims to provide a comprehensive understanding of the current landscape in the field. By exploring recent research, methods, and key developments, this chapter sets the foundation for the upcoming analysis and discussion, while also doing a brief discussion about the usefulness of the literature and an analysis of the existing gap.

From Section 3.4 onwards, the content is focused on the conclusions taken from the works analysed in the following sections.

3.2 Research Regarding primary objectives of Work

For this section of the state-of-the-art, a systematic review of papers and documents related to the dissertation is conducted.

3.2.1 Methodology for the Systematic Review

The systematic review was initially structured around two primary objectives.

The first objective aimed to analyze the methodologies employed in the literature

concerning cyber-attacks against DES. Specifically, it sought to identify the main techniques applied and studied, with a focus on addressing active attacks against these systems.

The second objective, which is closely related to the first, involved reviewing literature where opacity was implemented in scenarios involving active attackers. This objective aimed to explore how opacity was utilized as a defence mechanism in the presence of active adversaries.

With these objectives in mind, the next part is divided into these objectives, where the keywords will be specified, and a review of the gathered literature is done.

Research Questions

RQ1 – *what are the main characteristics of DES, regarding this problem?* Discover if the DES that is subject to active attacks implement any kind of controllers, like supervisors.

RQ2 – *what kind of active attack will be more useful, concerning the amount of literature?* Understand what the most suitable type of active attack for the systems is.

Due to the evolution of the Dissertation document, the segment addressing the answers to the initial research questions has been removed. However, in Section 3.4, although not explicitly identified as answers, the choices and conclusions drawn regarding the initial research and technique evolution implicitly address both research questions.

3.2.2 Active Attacks in DES

The set of keywords chosen for the search regarding this first objective was the following:

Discrete Event Systems, Cybersecurity

This dissertation predominantly focuses on opacity and active attacks, but the foundational elements are DES and cybersecurity. Hence, these keywords are deemed appropriate for the initial stage of the systematic review, where the main objective is to understand which are the more frequent types of attacks against DES and understand the security techniques used to mitigate them.

Aligning with the defined objective, the paper by [Oliveira et al., 2023] conducts an analysis and categorization of multiple papers, guided by the distinct characteristics of cybersecurity strategies, the types of attacks considered, and the employed modelling formalism. This specific paper was chosen since it contained a relevant systematic review of the required concepts, thus revealing that **review** should have been added to the keywords, as the other papers from the results were technique-specific, for example, the works of [Zheng et al., 2023] and [Fritz and Zhang, 2023].

In the work done in [Oliveira et al., 2023], the systematic exploration is driven by the goal of synthesizing relevant research in the literature to present the current

state of the art in cybersecurity strategies for DES. Which aligns perfectly with the objective defined.

From the review conducted, findings uncover a predominant focus on cybersecurity methods designed to safeguard systems against passive attacks. The use of automata as the chosen modelling formalism is also verified. Nevertheless, recent years have witnessed a noteworthy shift in emphasis, with a growing body of literature addressing the critical aspect of **active attacks**. This shift is substantiated by an increasing number of publications across various journals and conferences, providing valuable insights into the evolving landscape of research in this field.

As the proposed technique advanced, the emphasis within the work outlined in [Oliveira et al., 2023] also evolved, focusing more on the category of security measures to counteract active attacks, more so, regarding the subcategory of Attack-Tolerant Control, where the literature aligns with the objectives for the technique.

3.2.3 Opacity Against Active Attackers in DES

This subsection focuses on the research and review of works that apply opacity in systems that are subjected to active attacks, therefore, the keywords chosen were:

Discrete Event Systems, Opacity, Active Attacks

The research results underscored that this field has not received significant attention in recent years, as the limited number of findings suggests a relative scarcity of comprehensive studies. However, three papers on this subject were found, [Hélouët et al., 2018], [Partovi et al., 2020] and [Yao et al., 2024]. The work done in [Hélouët et al., 2018], upon reading the abstract proved to be out of scope for this dissertation, since the modelling format and the mathematical approach were different to the one learnt during the background development. The remaining two papers adopted distinct perspectives and were selected to be reviewed. Analyzing these papers will be valuable, as certain ideas can be repurposed in future work and integrated into the proposed technique.

Opacity of Discrete Event Systems with Active Intruder - [Partovi et al., 2020]

The paper states that traditional treatments of opacity in the DES literature often assume the presence of passive intruders—observers who merely scrutinize the system's behaviour. However, in cybersecurity contexts, particularly in web services, addressing active intruders who can exert influence beyond passive observation becomes essential. This prompts extending existing opacity notions to encompass scenarios involving active intruders.

To tackle this challenge, the system is modelled as a non-deterministic finite-state transducer. The modelling assumes that intruders possess comprehensive knowledge of the system structure and can interact by injecting diverse inputs while observing system responses. Within this context, the paper introduces the notion of reactive current-state opacity (RCSO), characterizing a property indicating that

the system does not reveal its secret state, regardless of the intruder's manipulations.

This concept is further extended to language-based and initial-state reactive opacity notions, with an exploration of the relationships among them. Notably, the paper reveals that all proposed reactive opacity notions are equivalent to RCSO. Consequently, the focus narrows to RCSO, and the paper delves into the associated verification problem. The study demonstrates that RCSO verification can be achieved by constructing an observer automaton. Throughout the paper, illustrative examples are provided to clarify key definitions and highlight the effectiveness of the proposed opacity verification approach.

As mentioned, the system is modelled as a non-deterministic finite-state transducer (NFT), which is a type of DES that captures the transformation of data by processing inputs and generating outputs with finite memory, thus characterizing the interaction between the system and its environment, and emphasizing a system model capable of receiving input from an active intruder.
Due to the characteristics of the system used, the authors quickly demonstrate that the technique employed to evaluate whether a system is current-state opaque, specifically in systems where only the occurrence of events is considered, does not apply in these cases. To show these discoveries, all the figures shown that regard this work were taken from the original paper. To summarize the work presented in this paper, attention is directed towards a key example prominently featured throughout the text. This example, crafted by the original authors, serves as a central illustration to facilitate a clearer understanding of the methodologies and findings outlined in the paper. It is essential to acknowledge that the creation of this example is attributed to the original authors, and any figures or adaptations provided herein are solely for clarification.

Considering the open DES G depicted in Figure 3.1 with $\Delta_o = \{\delta_1, \delta_2, a\}$, $\Delta_{uo} = \{b\}$, and $Q_s = \{3\}$. In the passive intruder scenario, only observable outputs are accessible through the projection function P. To assess current-state opacity on G, associate a NFA A with the open DES G. Let $A_G = (Q, \Delta, Q_0, T'_a)$, where the transition function T'_a, for any $q, q' \in Q$, and $\delta \in \Delta$, is defined as $q' \in T'_a(q, \delta)$, if there exists $x \in X$ such that $q' \in T(q, x)$ and $\delta \in \lambda(q, x)$; otherwise, $T'_a(q, \delta)$ is not defined. An observer automaton can be constructed to check if A_G is current-state opaque for P and Q_s. The observer, shown in Figure 3.2, indicates that the secret state $\{3\}$ never entirely lies on a single state of the observer. Hence, A is current-state opaque concerning Q_s and P.

However, if the intruder is capable of providing a certain input word to the system and observing the system's output through P, she can infer when the system is in a secret state. Specifically, consider the input word $w = x_1 x_2^* x_1$ that drives the system to land on one of the states $\{2, 3\}$. If the active intruder chooses x_2, i.e., $w \cdot x_2$, and observes a, she can infer the current state of the system is certainly at the secret state $\{3\}$. However, if a is an unobservable event, the active intruder with the same input word $x_1 x_2^* x_1 x_2$ cannot determine whether the system is at $\{3\}$ or $\{2\}$.

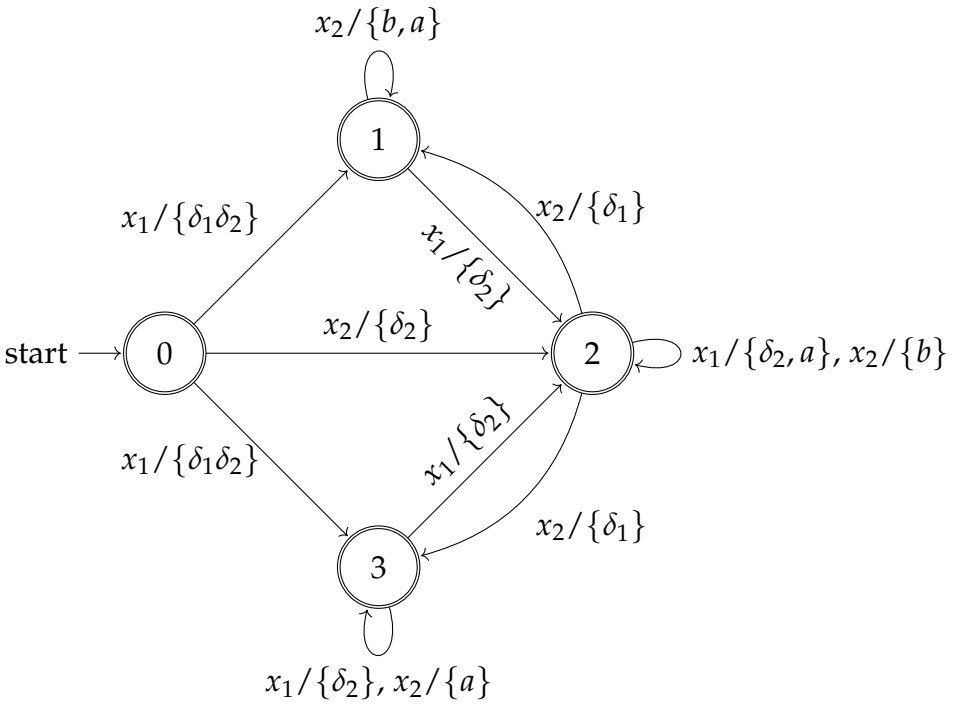


Figure 3.1: Open DES G. Noting that $\epsilon \in \lambda(q,x)$, for all $q \in Q$, and $x \in X_\epsilon$. Transitions with ϵ were removed for clarity purposes. See 1.

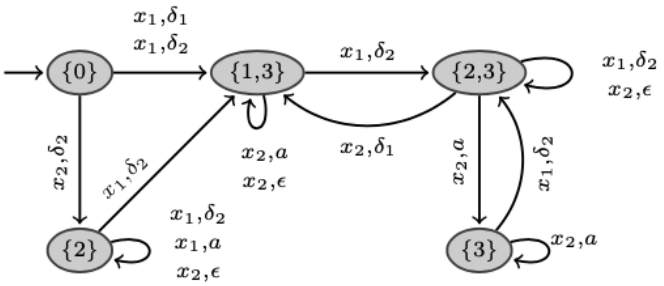


Figure 3.2: Observer automata for Figure 3.1

1

To answer the problem shown in the previous example, the authors of the paper introduce the concept of *Reactive Current State Opacity (RCSO)*, stating that, given an open DES $G = (Q, X, \Delta, Q_0, T, \lambda)$, with projection function P, and set of secret states $Q_S \in Q$, a system is RCSO, if for any $w \in L(G)$, exists $q_0 \in Q_0$ that verifies:

- $T(q_0, w) \cap \{Q - Q_S\} \neq \emptyset$,
- $\forall t \in P(O(w, q_0))$, having $\widetilde{Q}_{q0} \cap \{Q - Q_S\} \neq \emptyset$

Thus, this means that for G to be RCS-opaque, if, for any input word w recognized by G, $w \in L(G)$, the following conditions are met:

1. There exists an initial state $q_0 \in Q_0$ such that the system with w does not exclusively reach the secret states;
2. For any potential observable output word associated with the input, $t \in P(O(w, q_0))$, it is known that $\widetilde{Q}_{q_0}(w, t) \cap \{Q - Q_S\} \neq \emptyset$. This implies that the intruder cannot utilize the observed output events to resolve the non-determinism of the transition function $T(q_0, w)$ and infer the current secret state of the system.

Considering this definition, a solution is then possible to derive, to be able to verify if a system is RCS opaque. Similarly, with the concept of current-state opacity involving a passive intruder, the authors propose the utilization of an observer automaton to assess the RCS opacity of an DES. In the context of reactive opacity, it is important to note that the intruder possesses knowledge of the injected input word, thereby having awareness of the system's non-deterministic transitions. For RCS-opacity verification, the observer must account for both potential input variations and observable output behaviours to monitor the estimated states. Additionally, it is noteworthy that an open DES might exhibit only a singular, and possibly unique, unobservable output event for a given input, which could potentially unveil a secret state. In contrast to traditional opacity scenarios involving passive intruders, an active intruder is capable of leveraging even an unobservable response to deduce the states of the open DES. This capability needs to be explicitly incorporated into the observer for an active intruder.

Given the previous definition, and considering the open DES G in Figure 3.1, with $\Delta_o = \{\delta_1, \delta_2, a\}$, $\Delta_{uo} = \{b\}$, and $Q_s = \{3\}$, the observer that regards the previous definitions is represented in Figure 3.3, where an edge label is of the form x, δ, where $x \in X$ and $\delta \in \Delta_{o,\epsilon}$. From an analysis of the figure, it is concluded that the secret state $\{3\}$ is reachable from the initial state of the observer, thus meaning that G is not RCS-opaque, as initially predicted in the first example.

[1] This representation was recreated for this dissertation, as the original had a small overlap.

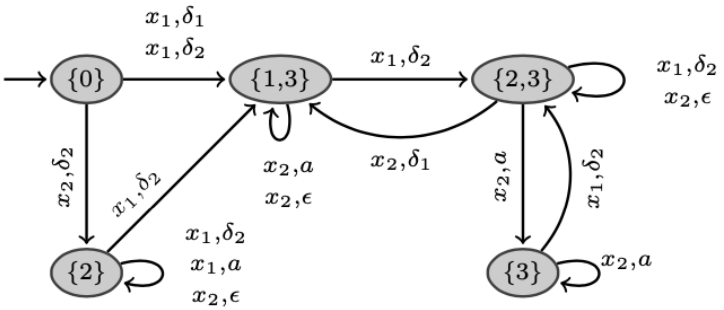


Figure 3.3: Observer automata for Figure 3.1, using to verify RCS opacity

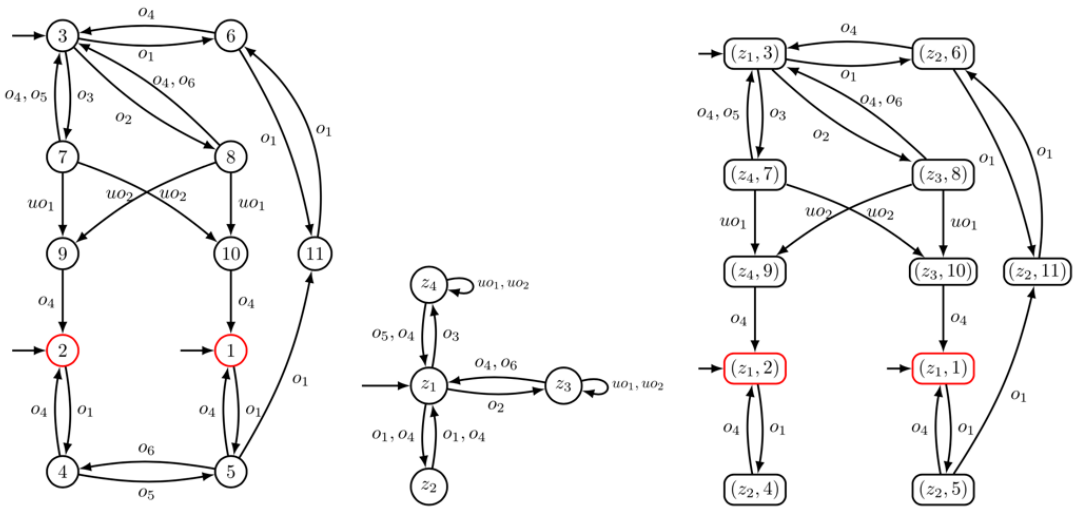
Sensor deception attacks against security in supervisory control systems - [Yao et al., 2024]

This paper delves into the synthesis of sensor deception attacks concerning security within the realm of supervisory control for DES. The focus is on a DES plant controlled by a supervisor with a security requirement to conceal the initial-state of the system, ensuring it remains undetected—that is, avoiding any disclosure of its initiation from a secret state. In this scenario, an active attacker is present, capable of manipulating the observations that the supervisor will receive. This manipulation could involve actions such as hacking into the communication channel between the sensors and the supervisor. The primary objective of the attacker is to deceive the supervisor, leading to the revelation of the secret initial-state, through incorrect control actions.

This investigation adopts the attacker's perspective and concentrates on synthesizing attack strategies that threaten the system's security. Two levels of success for the attacker are considered: one requires the attacker to almost surely detect the initial-state, while the other only necessitates the possibility of detecting the initial-state. For both cases, the paper presents algorithms designed for synthesizing successful attack strategies. The approach is rooted in the All Attack Structure (AAS), which records state estimates for both the supervisor and the attacker. Additionally, structural properties of the security requirements are leveraged to mitigate synthesis complexity.

To illustrate the proposed synthesis procedures, a running academic example is provided throughout the paper. This example is present in Figure 3.4 fig a), and it represents system G, where $A_0 = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ is the alphabet of observable events and $X_0 = 1, 2, 3$ is the set of initial states, supervisor S is modelled as figure b) of Figure 3.4, and figure c) of Figure 3.4 is the closed-loop system under control S/G is modelled through the automaton $G \times H$.

This paper delves into the perspective of the attacker, which doesn't seem as useful as the paper [Partovi et al., 2020], since, for now, the perspective that will be taken into consideration in the dissertation, will be of the system. Nonetheless, a protection against the All Attack Structure (AAS) methodology, could be explored, but, as it's mentioned in the paper, the simplified version of this approach, regarding an attacker that discovers the secrets of the system and remains undetected by the supervisor, has a complexity that is exponential concerning the size



(a) System G (b) Supervisor H (c) Closed-loop system $G \times H$

Figure 3.4: Figures present in [Yao et al., 2024]

of the system, in the worst case scenario.
It's worth noting that the authors also highlight the inevitability of encountering exponential complexity in the worst-case scenario for the majority of partial-observation synthesis problems.

3.3 Enforcing opacity and Attack detection followed by mitigation

This literature section discusses some of the conclusions taken from the proposed technique. As explained in Chapter 4, there is the need to find techniques for the enforcement of Current-State Opacity in a system and to detect and mitigate an active attack. Considering this, this section explores the literature found regarding these three concepts.

3.3.1 Enforcement of Current-State opacity

In Chapter 4, it will be explained that a system not opaque by default needs transformation to acquire this characteristic. The literature examined, focusing on DES modelled using automata and Current State Opacity, displayed varying characteristics in the approaches used. The following keywords were used for the research conducted:

Discrete Event Systems, Current-State Opacity, Enforcement, Automata

From the usage of these, the results obtained were evaluated based on the abstract, how recent the literature was and the number of citations. It is also important to mention that if **Utility** or **Supervisory Control** was in the title, then it would be analysed, since keeping the utility of the system is also one of the

main objectives to achieve, and the usage of a supervisor to achieve opacity in a system can also be useful. From the results obtained, a first selection of literature was done, through the elimination of publications before 2016. This made it so that the literature selected was: [Tong et al., 2016], [Barcelos and Basilio, 2023] and [Tong et al., 2018].

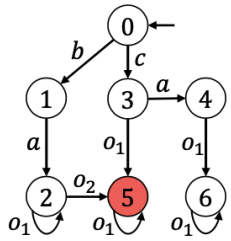
In [Barcelos and Basilio, 2023], the title suggests that the utility of the system is maintained, and in the abstract, it is mentioned that one of the main criticisms regarding opacity enforcement, concealing the system's secret behaviour from intruders requires obfuscating additional information from legitimate receivers, is addressed in it.

On the other hand, the works in [Tong et al., 2016] and [Tong et al., 2018] focus on a supervisory approach. In [Tong et al., 2016], despite being an older publication, the approach presented has not been replicated in more recent years with comparable characteristics, since the concept of "Uncomparable Observation" was introduced, where it is assumed that the supervisor can only observe and control a subset of events within the system. There is no predefined containment relationship between the events that are observable by the intruder and those by the supervisor. The paper [Tong et al., 2018], which shares some of the authors of [Tong et al., 2016], considers the approach of multiple and simultaneous attacks, which falls outside the scope of the research. Both of these papers fall outside the scope of this dissertation since the specific considerations about the observations in [Tong et al., 2016] and the multiple simultaneous attackers in [Tong et al., 2018] would require the rest of the techniques, attack detection, and mitigation, to also consider these, which is infeasible.

Nonetheless, from the references of both these papers, literature that takes into consideration the supervisory control, for opacity enforcement where the supervisor and the attackers have the same event observability was analysed. The following paper was identified [Yin, 2015].

After analysing the paper [Yin, 2015], as this publication is older, after a search looking for other publications about opacity enforcement through supervisory control in Partially-Observed DES, the paper [Barcelos and Basilio, 2023] was also analysed.

As both techniques present in [Barcelos and Basilio, 2023] and [Yin, 2015] are different, the chosen technique to be applied to the proposed technique in this dissertation will be inspired in the one of [Yin, 2015]. The reasons behind this decision were the need to adapt the opacity enforcement technique with the active attack mitigation technique, the technique developed in [Yin, 2015] due to its complexity of application was chosen to be used in the proposed technique. It is important to mention that the technique in [Barcelos and Basilio, 2023] could still prove to be important to implement in future work under the same characteristics.



System G with $E_c = \{a, b, c\}$, $E_o = \{o_1, o_2\}$, and $X_S = \{5\}$

Figure 3.5: Model of system G in [Yin, 2015]

A New Approach for Synthesizing Opacity-Enforcing Supervisors for Partially-Observed Discrete-Event Systems - [Yin, 2015]

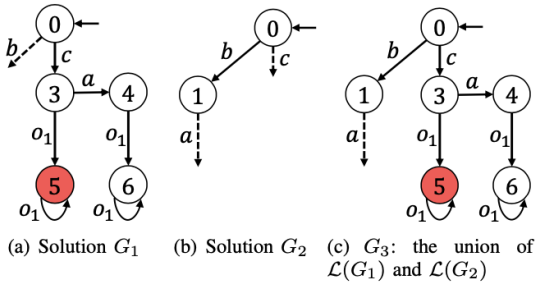
The authors of this paper delve into the critical realm of cybersecurity and privacy within discrete-event systems. The primary objective revolves around enforcing opacity on systems that are initially not opaque, employing supervisory control to restrict system behaviour effectively.

A notable contribution of this research lies in the development of a novel transition system termed the "All Inclusive Controller for Opacity" (AIC-O). This finite bipartite transition system encapsulates all valid opacity-enforcing supervisors within its structure. Through the construction of the AIC-O, the authors present an algorithm facilitating the synthesis of opacity-enforcing supervisors, thereby ensuring maximal permissiveness while enforcing opacity.

One distinctive aspect of the approach is its applicability to scenarios involving partial observations, where not all controllable events are observable. By relaxing the assumption of complete observability, the authors offer a more flexible and realistic framework for opacity enforcement via supervisory control. Moreover, the methodology extends beyond previous works by accommodating diverse observability properties of controllable events, thereby broadening the scope of applicability in practical settings.

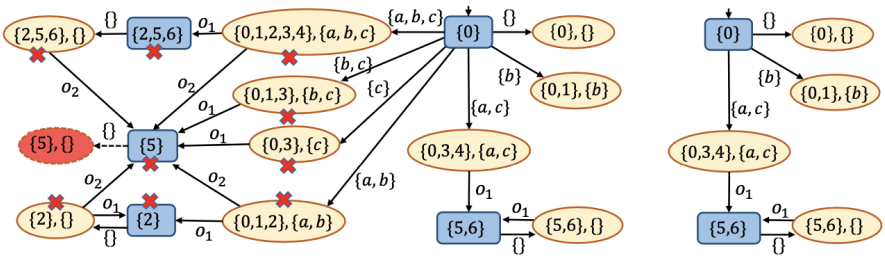
In the concluding remarks, the authors highlight the potential avenues for future research, emphasizing the need to address challenges such as relaxing the assumption of identical observable events between the supervisor and the intruder, as it was later explored in the already cited paper [Tong et al., 2016]. Additionally, prospect of leveraging the AIC-O framework for optimal opacity enforcement control, incorporating cost considerations into the synthesis process is also analysed. Overall, the innovative approach and theoretical contributions pave the way for advancements in ensuring the security and privacy of cyber and cyber-physical systems.

An analysis of the technique presented in the paper, regards Figure 3.5(a), which is the representation of system G, of which the sets $E_o = \{o_1, o_2\}$ and $E_c = \{a, b, c\}$ represent the set of observable and controllable events, respectively, and are deemed incomparable. As perceived by the model of system G, it is not opaque regarding its secret state 5, since the intruder can infer that the system's current state is state 5 once event o_2 is observed. To enforce opacity, a sub-



(a) Solution G_1 (b) Solution G_2 (c) G_3: the union of $\mathcal{L}(G_1)$ and $\mathcal{L}(G_2)$

Figure 3.6: Incomparable solutions example in [Yin, 2015]



(a) The resulting structure after procedure DoDSF (b) The constructed AIC-O

Figure 3.7: Construction of the AIC-O. Blue rectangles correspond to Y-states and yellow oval states correspond to Z-states.

language of $L(G)$ that is controllable, observable, and opaque needs to be found. In Figures 3.6(a), 3.6(b), solutions $L(G_1)$ and $L(G_2)$ are shown. These are two maximal controllable, observable and opaque solutions, however, the union of these two solutions, represented in 3.6(c), is not a valid solution, since event a needs to be enabled at state 1 and disabled at state 3, by the system. Since states 1 and 3 are indistinguishable in $L(G_3)$, the property of observability is therefore violated.
Regarding the Bipartite Transition System, the algorithm and definitions are present in the original paper, a remark to this type of system, regarding G, is in Figure 3.7(b), which is a type of BTS. Regarding the initial Y-state $y_0 = \{0\}$, with the control decision $\gamma = \{a, b, o_1, o_2\}$, uncontrollable events o_1, o_2 are omitted in the figure. The $Z-$state $z = h^T_{YZ}(y_0, \gamma) = (\{0, 3, 4\}, \{a, c, o_1, o_2\})$. From z, event o_1 is the only one that may occur, thus leading the system to the next $Y-$state $y = h^T_{ZY}(z, o_1) = \{5, 6\}$. The BTS represented in Figure 3.7(b) is considered complete. If control decision $\{o_1, o_2\}$, represented as $\{\}$ in the figure, is used at the initial $Y-$state $\{0\}$, no future behavior may occur, thus leading to a supervisor S_P that is BTS-included, and is defined by $S_P(\epsilon) = \{o_1, o_2\}$.

The referenced BTS in Figure 3.7(b) is the $AIC - O$ of system G, this can be shown at initial $Y-$state $\{0\}$, control decision $\{a, b, c\}$ cannot be made, since it would lead to $Z-$state $(\{0, 1, 2, 3, 4\}, \{a, b, c\})$. This is due to the occurrence of event o_2, $Y-$state 5 would be reached, and no other control decision could be taken to make the system not reveal its secret.

By applying algorithm FIND-AIC-O to system G, the resulting BTS obtained after procedure DoDSF is executed is represented in 3.7(a). The depth-first search ends at $Y-$state $\{5\}$, since any control decision made at state 5 will result in encounter-

ing a $Z-$state (marked in red in figure 3.7(a)) that reveals the secret of the system. After this procedure, function Prune is executed, starting by removing $Y-$state $\{5\}$, since there is no successor state defined from it. Following the removal of the state, all its predecessor $Z-$states, i.e., $(\{0,3\},\{c\}),(\{0,1,2\},\{a,b\})$, should be removed. In the final step, inaccessible states $\{2,5,6\}$ and $\{2\}$ are removed, and the AIC-O represented in Figure 3.7(b) is obtained.

When deciding on the control decision, if locally maximal control decision $\{a,c\}$ at the initial $Y-$state $\{y_0\}$ and the unique control decision $\emptyset$ at the reach-able $Y-$state are picked, meaning that all controllable events are disabled, the solution obtained will be the maximal solution, which was shown in Figure 3.6(a). On the other hand, if the control decision $\{b\}$ at $\{0\}$ is picked, also being another locally maximal decision, then no behaviour may happen after, meaning that it corresponds to the maximal solution shown in Figure 3.6(b).

3.3.2 Detection and Mitigation of an Active Attack

For this segment of the state-of-the-art, the research aimed to find a technique that could successfully detect and mitigate an active attack against a DES, through the usage of a supervisor. The research for such a technique was not easy and proved to be unsuccessful.
Although some literature regarding this problem used both a technique to detect and mitigate an active attack, it proved to be the case that in the analyzed literature, the mitigation technique would simply require the Supervisor to deactivate all controllable events, effectively halting the system, this technique, originally present in [Carvalho et al., 2018].
As a result, the research had to be divided, and later on, the proposed technique would combine both selected techniques. The research for a mitigation technique appears first since it was the primary focus, aiming to find an alternative approach distinct from the one proposed in [Carvalho et al., 2018].

Mitigation

When researching about possible mitigation techniques, the keywords used were the following:

DES, Automata, Supervisor, Attack Mitigation

From the results of the search, not many options that could be applied to the dissertation were left, since the modelling of the systems needs to be done with using automata. The work done in [Yao et al., 2020] aligns with the research objectives, and is based on a mitigation technique for an actuator enablement attack. This work is an important finding since the mitigation technique does not solely rely on disabling all the controllable events of a system.

The following section is dedicated to the main principles and concepts introduced by the authors, with the example of the presented mitigation algorithm working

in a system. As the paper is quite complex, the full description of all the concepts can not be present in this subsection, as it will focus on showing how the technique works.

On Attack Mitigation in Supervisory Control Systems: A Tolerant Control Approach - [Yao et al., 2020]

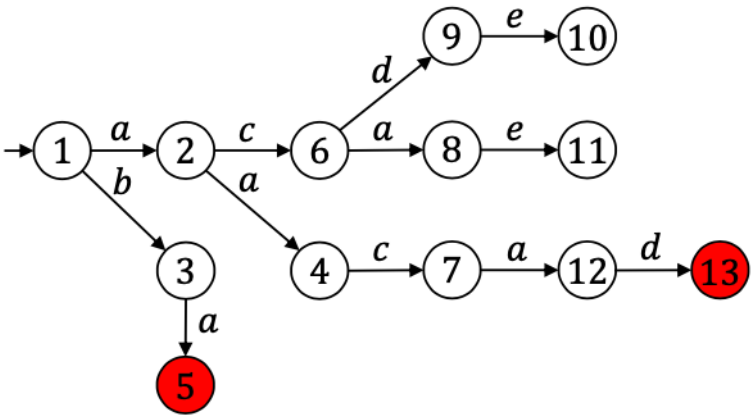
In this paper, the authors delve into the challenge of mitigating attacks within supervisory control systems for discrete event systems. The authors make a distinction between controllable events, which the regular supervisor can deactivate but are susceptible to attacks, and defendable events, which can be definitively disabled by a specialized mitigation module although potentially at higher costs. The primary aim of the paper is to develop a strategy to mitigate attacks effectively while minimizing resultant damage.

The authors approach the problem by formulating it as a tolerant control problem under partial observation. The goal of this paper has a dual objective: to maximize the desirable behaviour, which aligns with normal system specifications, while simultaneously minimizing tolerable but undesirable behaviour. To tackle this challenge, the authors propose an innovative online algorithm. This algorithm offers a novel attack mitigation strategy, expanding upon existing methodologies found in the literature. Notably, the authors demonstrate that their proposed strategy has the capability to prevent damage even in scenarios where the safe-controllability condition, a prerequisite in conventional strategies, is not met. A sub-chapter is present in the literature, where the authors analyse the technique present in [Carvalho et al., 2018], mentioning that such technique waits for the attack to be detected, and only after the mitigation module will defend the system, by disabling all the events in the set of controllable events.

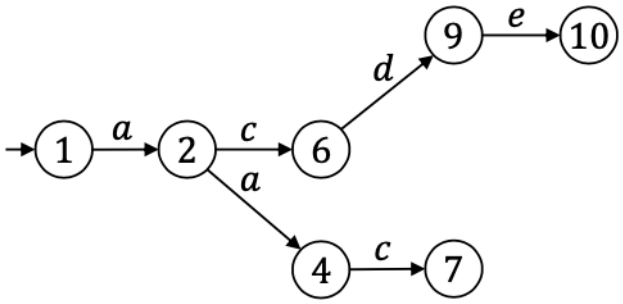
To illustrate the proposed technique procedure, an example is given at the end of the paper, which is going to be included here.

Considering system G shown in Figure 3.8 (a) with $\Sigma_c = \{a, b, d, e\}$ and $\Sigma_o = \{c\}$. It is supposed that the system is originally controlled by S_P to achieve the desirable specification $L(S_P/G) = \{acde, aac\}$ and S_P/G is shown as Figure 2(b). Specifically, the supervisor disables event b initially and disables event a after observing c. It is assumed that the vulnerable events set is $\Sigma_v = \{a, b\}$, defendable events set is $\Sigma_d = \{b\}$, and $X_{\text{bad}} = \{5, 13\}$ is the set of unsafe states. Under AE-attack, in both unsafe states, 5 and 13 can be reached. Furthermore, the strategy in [Carvalho et al., 2018] cannot prevent unsafe states.

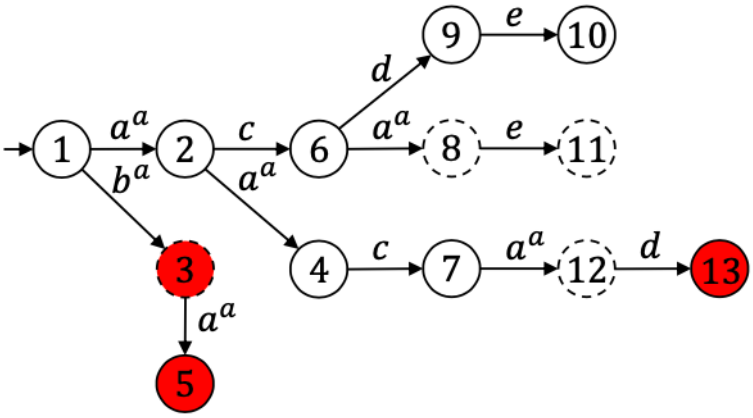
To apply the online mitigation strategy, first, the DFA $\tilde{G}_a$ is constructed, shown in Figure 3.8 (c) with $\Sigma_{c,a} = \{b^a, d, e\}$. It is of relevance to mention that the state space of G_a is already partitioned as $X_D = \{1, 2, 4, 6, 7, 9, 10\}$, $X_T = \{3, 8, 11, 12\}$, and $X_F = \{5, 13\}$. Since state 3 can reach unsafe state 5 via an uncontrollable event a^a, it is assumed that $X_F^+ = \{3, 5, 13\}$. It is then possible to construct the mitigation strategy M using Algorithm 1. To start, $\alpha = \varepsilon$ and $\hat{E}(\alpha) = \{1\}$. Procedure CHOOSEACTION($\{1\}$) is then called as detailed in Algorithm 2. Initially, $\gamma \leftarrow \{c, a^a\}$ and it is assumed that events in EventList are ordered by $[b^a, d, e]$.



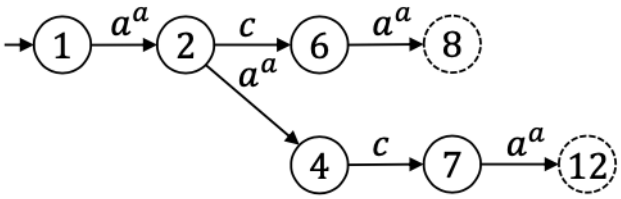
(a) G



(b) S_P/G



(c) $\tilde{G}_a$



(d) $M/\tilde{G}_a$

Figure 3.8: Figures present in [Yao et al., 2020]

First, we test whether or not event b^a can be added to γ. However, by doing so, state 3 which is in X_F^+ will be reached; therefore, it is deleted. Then events d and e are tested, in order, and both events are not feasible within the unobservable reach. Consequently, these are skipped and $M(\varepsilon) = \{c, a^a\}$ is returned, i.e., the mitigation module needs to defend event b initially. Next, when event c is observed, it is known that $\alpha = c$ and $\hat{E}(\alpha) = \{6, 7\}$. Again, procedure CHOOSEACTION({6, 7}) is called to compute the current decision to be made. First, event b^a is assessed, which is not feasible and, therefore is skipped. Then by adding event d, state 13 becomes reachable and so event d is deleted. For event e, although it is feasible from state 8, which is in the unobservable reach of $\hat{E}(\alpha) = \{6, 7\}$, it is not feasible in X_D. Therefore, event e also needs to be skipped because it does not contribute to the desirable behaviour although it is safe. Then, the returned decision $M(c) = \{c, a^a\}$ is obtained. The overall closed-loop system under the mitigating strategy M is shown in Figure 3.8 (d). The closed-loop system sacrifices two desirable strings *acd* and *acde*, reaches two tolerable but not desirable states 8 and 12, but successfully avoids reaching unsafe states.

It is important to reference once again that the Algorithms mentioned for the execution of this technique are present in the original paper, and will be examined in Chapter 4. These were chosen not to be included here, only the main example that explains how to apply such technique and the end result for a system.

Detection

When researching about detection techniques, the keywords used were the following:

DES, Automata, Supervisor, Attack Detection

From the results obtained, some that have already been previously mentioned, as is the case of [Carvalho et al., 2018], the main interest was also regarding the complexity of said technique. Therefore, the literature in [Tong et al., 2022] seemed appropriate to analyse, for this objective.

A Polynomial Approach to Verifying the Existence of a Threatening Sensor Attacker - [Tong et al., 2022]

In this paper, the authors address the issue of cyber-attacks targeting supervised discrete event systems within cyber-physical systems (CPS). These attacks involve manipulating sensor readings to make the closed-loop system reach undesired states. To combat this threat, the authors propose a novel attack detection mechanism where the supervisor only needs to monitor the most recent observable event. They introduce the concept of a **sensor attacker** defined as a threat capable of pushing the system into forbidden states. The primary objective is to determine whether such a menacing sensor attacker exists for a given controlled system. To achieve this, they introduce the All Sensor Attack (ASA) structure, designed to encompass all potential sensor attacks orchestrated by the attacker. Utilizing the ASA automaton, the authors establish a necessary and sufficient condi-

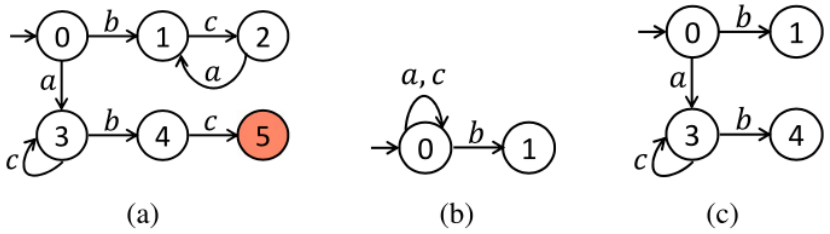


Figure 3.9: (a) System G, (b) Supervisor S, (c) Controlled System S/G

tion for the existence of a stealthy threatening sensor attacker. Importantly, they demonstrate that this condition can be verified efficiently in polynomial time, offering a promising avenue for practical implementation and real-time cybersecurity management in CPS.

AS previously done in other analysed literature, the main example used throughout this paper will be exposed, thus revealing how this technique works.

Considering the plant G in Figure 3.9, where $E_c = \{a, c\}$, $E_o = \{b, c\}$, and $X_l = X \setminus \{5\}$. A supervisor S that prevents G from reaching state 5 is shown in Fig. 1(b), which encodes the control function $\varphi(c^*) = \Gamma_S(0) = \{a, b, c\}$ and $\varphi(c^*b) = \Gamma_S(1) = \emptyset$. The controlled system S/G is shown in Fig. 1(c). Considering this system, and assuming that the sets of events subject to SE-attacks (Sensor-Erasure) and SI-attacks (Sensor-Insertion) are $E_{si} = \{c\}$ and $E_{se} = \{b\}$, respectively. Its supposed that the sensor attacker is ξ, for $w = \epsilon$;

$$\xi(w) = \begin{cases} \epsilon, & \text{for } w = \epsilon; \\ c^{n_1}b - c^{n_2}, & \text{for } w = c^{n_1}bc^{n_2}, \text{ with } n_1, n_2 \in \mathbb{N}. \end{cases}$$

In words, the attacker erases event b after observing it. Supposing that $\sigma = acb$ occurs in $L(S/G)$. The plant reaches state 4. The attacker observes $P_o(acb) = cb$ and erases b (i.e., $\hat{P}_o(\xi(cb)) = c$). As a result, the supervisor observes c and reaches state 0, where event c is enabled. Therefore, the plant reaches the forbidden state 5 from state 4 with the occurrence of c. Since $cc \in P_o(L(S))$, the attack on $acbc$ is stealthy.

Analogously, the controlled system under the attack of ξ can be constructed, which is identical to G.

The controlled system under the action of the sensor attacker ξ is denoted as S_ξ/G.

The corresponding $\hat{G}$ and $\hat{S}$ are shown in Figure 3.10. To make the construction more illustrative, additional transitions of events in E_+, E_-, and E_{uc} are coloured in red, blue, and green respectively.

The ASA model $H = \hat{G} \mid\mid \hat{S}$ can be observed in Figure 3.11. It is known that $Q_d = \{q_2, q_9\}$ (marked in grey), and $Q_f = \{q_7\}$ (marked in red).

Since $Q_f = \{q_7\}$, by Theorem 1, for G and S in Figure 3.9, $E_{si} = \{c\}$ and $E_{se} = \{b\}$, there exists a threatening sensor attacker. Indeed, the string ab_c,

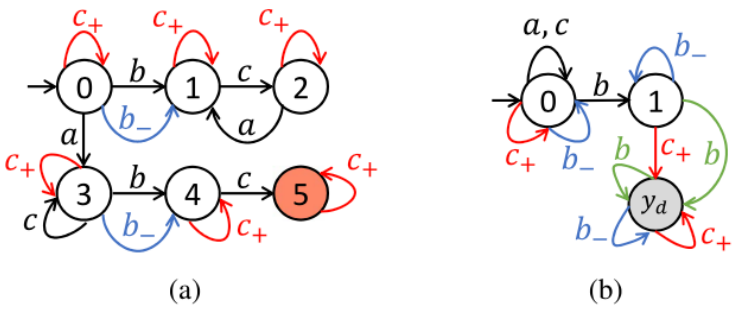


Figure 3.10: (a) Model $\hat{G}$ of the plant under attack. (b) Model $\hat{S}$ of the supervisor under attack

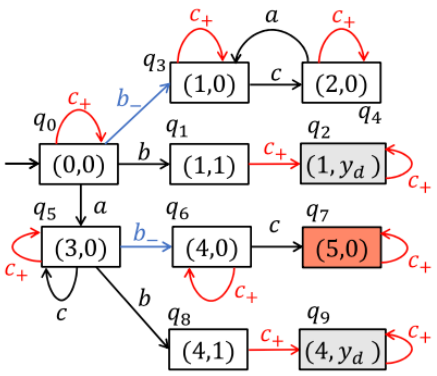


Figure 3.11: ASA model H of $\hat{G}$ and $\hat{S}$ of Figure 3.10

for instance, leading to q_7 implies that for $\sigma = ab$ (i.e., after observing b), the attack erasing b misleads the supervisor to enable c and causes the plant to reach 5. This string captures one attack action of the attacker ξ previously mentioned, using the attack string $acbc$.

The threatening attacker does not always exist. If the supervisor is replaced with the one in Figure 3.12. To check whether there exists a stealthy threatening attacker, first the corresponding S (as shown in Figure 3.12 is constructed, and then the ASA model H in Figure 3.13. It is known that $Q_d = \{q_3, q_5\}$ and $Q_f = \emptyset$. By Theorem 1, there is no threatening sensor attacker with the capability of $E_{si} = \{c\}$ and $E_{se} = \{b\}$ that can cause damage to the controlled system S/G.

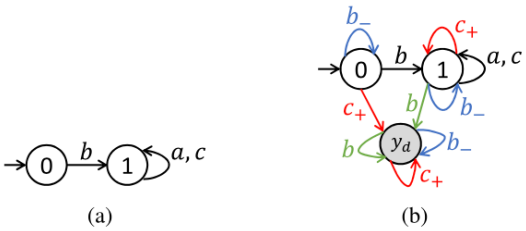


Figure 3.12: (a) Example of a supervisor S enforcing the specification G, (b) corresponding model $\hat{S}$ of the supervisor under attack

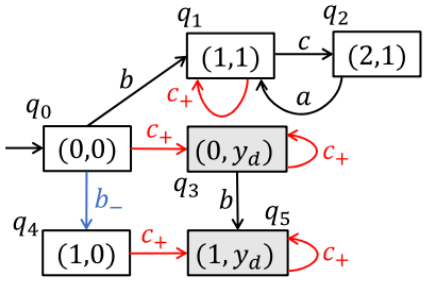


Figure 3.13: The ASA model H of $\hat{G}$ in Figure 3.10(a) and $\hat{S}$ in Figure 3.12(b)

3.4 Discussion of the Literature and Gap Analysis

The literature explored in this chapter proves the interest of the scientific community in investigating the the definition of novel methodologies to take advantage of opacity to secure DES.

An important observation and step towards the development of the technique proposed in this thesis work arose with the analysis of the automata reported in Figures 3.1 and 3.3, presented in the work [Partovi et al., 2020]. This automaton has a crucial characteristic: State 1 always preserves current-state opacity when within the set of secret states, i.e., $S = \{1\}$. This means that even though the DES is influenced by an active attacker, having State 1 as the sole secret state makes it impossible for the attacker to discover it, due to the redundancy of observable events that occur, never truly revealing that the system's current state is state 1.

In the case a system is not intrinsically current-state opaque with respect to active attacks, then the work done in [Yao et al., 2024], as mentioned, it delves into the attacker's perspective. The automaton considered in the paper is initially opaque through the enforcement of opacity by a controller, in this case, a supervisor, a technique analyzed in subsection 2.7.4. Figure 3.4 shows the original model of the system considered. A system with these characteristics is significant because a system being opaque does not necessarily need to be an inherent feature, as in [Partovi et al., 2020]; it can be implemented through a controller. The objective of the literature was to deceive the supervisor into revealing the system's secret, specifically the set of secret initial states, thereby subjecting the system to a Man-in-the-Middle (MitM) attack.

3.5 Systems to be Considered and Security by opacity

Considering both types of referenced systems, the objective of the technique and the concept of security by opacity evolve once again.

3.5.1 Opaque System by default

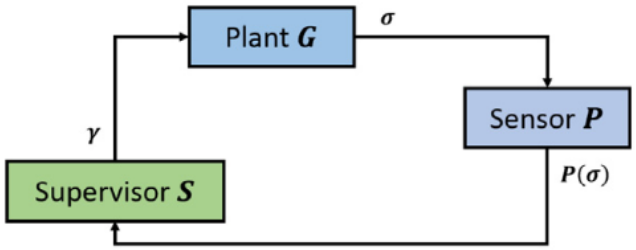
In an inherently opaque system, like the one described in [Partovi et al., 2020], opacity is a built-in characteristic. This original system includes a state that preserves current-state opacity against all types of attackers. Therefore, if the main goal is to safeguard a critical state of the system, having such a characteristic ensures that the system's critical state remains undisclosed to any type of attacker, whether active or passive.

It's crucial to note that in a Man-in-the-Middle (MitM) attack, opacity remains unbroken since the system cannot reveal the secret. However, additional security measures are necessary because the attacker could still disrupt the system's actions, potentially creating new risky states by blocking transitions or preventing actions from actuators. While this consideration is important for this dissertation, which aims to explore securing a DES through opacity, it falls outside the main objective. Nevertheless, the system's ability to maintain the secrecy of the state remains theoretically intact.

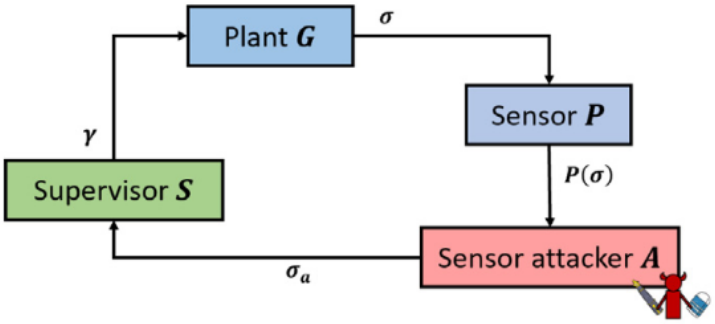
3.5.2 Opaque System through controller enforcement

When a system is already developed, it's unlikely to be inherently opaque. Thus, a controller like a supervisor needs implementation to achieve opacity, as demonstrated in [Yao et al., 2024]. However, systems of this type are vulnerable to cyber attacks, such as Man-in-the-Middle (MitM) attacks, which can deceive the supervisor and compromise the system's secrecy. This means, that another type of security technique needs to be employed, opacity will not be sufficient to protect the system from revealing its secret. These vulnerabilities are particularly pronounced in networked control systems, where powerful attackers can actively manipulate sensor observations to mislead feedback decisions, a tactic known as sensor deception attacks. For example, attackers could exploit sensor deception attacks to compromise a supervisor enforcing current-state opacity, as depicted in Figure 3.14 (b), being originally in the work of [Yao et al., 2024].

In systems with this characteristic, the proposed technique operates as a system with two layers. At its core, the supervisor controls the system to prevent the revelation of the secret. Surrounding this core is an outer layer comprising a security mechanism against Man-in-the-Middle (MitM) attacks to ensure the supervisor isn't deceived into revealing the secret. In essence, the technique in such a system must detect and mitigate the interference of an active attacker while safeguarding the secret.



(a) Nominal supervisory control system S/G.



(b) Supervisory control system S_A/G under attack.

Figure 3.14: Sensor deception Attacks

3.6 Opacity and Active Attackers

As explained in the previous section, two types of systems are considered, either a system is inserted in Subsection 3.5.1 or Subsection 3.5.2. Accordingly, the defense technique for each will to be different.

For systems falling under the category of Subsection 3.5.1, no additional security technique is necessary. These systems inherently maintain the secrecy of the system's state, ensuring opacity aligns with the thesis's focus. Consequently, any other security measures are beyond the scope of this thesis.

In contrast, the systems described in Subsection 3.5.2 require a different approach, which will be the focus of this section. Building upon the literature analyzed in Chapter 3, particularly in Section 3.3, this technique will be developed based on the insights gained from the literature review.

Starting with the Detection and Mitigation defense techniques discussed in the literature review, there's a notable gap regarding mitigation techniques for Active Attacks against DES modeled using automata. While many techniques in the literature adopt the approach proposed by [Carvalho et al., 2018], as discussed in [Yao et al., 2020], this technique may not always be sufficient to mitigate attacks without repercussions to the system.

The analysis in [Yao et al., 2020] reveals that while it successfully mitigates active attacks, the developed technique has some drawbacks. Firstly, it assumes the invulnerability of "defending" events, implying that attackers cannot interfere with the actions of the specialized mitigation module. While this assumption may hold in some real-world scenarios, it's a crucial consideration to bear in mind.

Although the initial intention was to search for detection and mitigation techniques separately to combine them for the proposed technique within this dissertation, the work of [Yao et al., 2020] showed that such a technique doesn't require a separate detection system. Therefore, attempting to merge the techniques from [Yao et al., 2020] and [Tong et al., 2022] is unfeasible.

Consequently, the action plan regarding the proposed technique evolved to merging the Opacity Enforcement technique in [Yin, 2015] with the Active Attack mitigation technique of [Yao et al., 2020]. If successfully implemented, this technique would maintain the system's current-state opacity even when under the influence of an active attacker, pioneering a novel approach in the field. The work done to implement this

3.7 Summary

This chapter delved into related works, initiating with an initial search. The exploration extends to a focused investigation on opacity and active attackers. The chapter aims to offer insights into existing literature, emphasizing discoveries from both the general search and the specific inquiry into opacity, active attackers, enforcement of opacity and attack mitigation techniques. The final sections of the chapter delve into a discussion of literature and the gap present in it. From this discussion, the findings and conclusions will be applied in the following Chapter, the proposed technique developed for this dissertation.

Chapter 4

Opacity-Based Defense for DFA Against Passive and AE-Attacks

4.1 Introduction

The dissertation's primary objective is to contribute to the area of cybersecurity that regards DES and explores the concept of opacity and how it may be applied in a different environment. This technique started with the intention of securing a DES using opacity, and, as it will be explained, evolved to something more complex, as an active attacker was regarded. The following sections aim to explore and explain the technique that has been created and the steps that were needed to reach it.

4.2 Enforce opacity and mitigate attacks via supervisory control

Here the aim is to merge the techniques by [Yin, 2015] and [Yao et al., 2020] to achieve the proposed novel technique to defense DFAs from passive and actuator-enabling attacks. To this purpose the system specific characteristics considered in both works need to be analysed, so as to define the assumptions to be considered for the application of the novel methodologies. A test system is used as an example and for initial tests.

First, considering the system and technique in [Yin, 2015], the system must satisfy the following assumptions:

- **Set of Controllable and Observable Events** - There exist 4 sets, A_c, A_{uc}, A_o, and A_{uo}, where $A_c \cap A_{uc} = \emptyset$ and $A_o \cap A_{uo} = \emptyset$. An observable event does not necessarily have to be controllable or vice versa, i.e., $A_c \cap A_o = \emptyset \vee A_c \cap A_o \neq \emptyset$;
- **Existence of a Secret State in the System** - To maintain opacity, the secret

of the system must be identified before applying the technique. The secret can be one ore multiple states of the system.

In this literature, when defining a Bipartite Transition System, Information States are introduced. A Y-state is an information state from which the supervisor can issue control decisions, while a Z-state is an information state where observable events occur within the set of enabled events. Although these states characterize the system, they do not need specific characteristics associated with them. The existence of the previously mentioned sets will automatically ensure these states exist in a system.

Regarding the work from [Yao et al., 2020], it is important to mention that the technique defends a system from **actuator-enablement attacks**. Therefore, sensor erasure, sensor insertion and actuator disablement attacks are not considered for this technique.

For the technique presented in the paper, there are system specific considerations that need to be taken into account, such are:

- **Set of Controllable and Observable Events** - As previously exposed for the works of [Yin, 2015], the same characteristic is a requirement for this technique.
- **Events influenced by the Attacker** - To simulate the active attack in the system, certain events $A_v \subseteq A_c$ a set of *vulnerable events*. These may be observable or not, and the attacker is able to enable a vulnerable event even if it has been disabled by the supervisor. The set of attacked actuator attacks is defined by $A^a_{c,v} = \{\theta^a : \theta \in A_v\}$, and thus $A_a = A \cap A^a_{c,v}$ is defined. The usage of θ^a represents the occurrence of θ after being originally disabled by the supervisor but enabled by the attacker.
- **Systems' Behavior** - We assume that the system can present certain types of behavior. In particular, the system presents the following types of behaviours:
  - **The desirable behavior**, described by the normal language $K_{des} = D(L(S_P/G)) = D(L(H \mid\mid G))$, which is the language generated by a system G under supervision of S_P S_P/G without an attack.
  - **The tolerable behavior**, described by the language of strings that fall out of K_{des} but also out of the unsafe language, defined by:

$$K_{tol} = \{s \in L(G_a) : \forall t \leq s, \delta_a(x_0, t) \neq X_{bad}\}$$

  - **Types of states**: It is considered that, X_D, X_T and X_F represent, desirable states, tolerable states and unsafe states, respectively.
- **Defendable Events** - The mitigation module of this system has the ability to disable all defendable events. The set of defendable events is assumed to be A_d, having: $A_d \subseteq A_v \subseteq A_c$.

The starting system must comply with all the requirements from both lists of characteristics. The proposed technique aims to preserve Current-State (CS) Opacity in a system under the influence of an active attacker. The principle behind the technique is that a supervisor enforcing CS Opacity can be influenced by an active attacker, potentially revealing the system's secret. This technique will involve enforcing CS Opacity followed by applying the mitigation technique against actuator enablement attacks.

From the analysis of both lists of requirements, it becomes easier to grasp the necessary details to implement both techniques. To better understand the changes needed to create a unified technique that incorporates both approaches, we will start with the example depicted in Figure 3.5 in Subsection 3.3.1.

Considering the characteristics of both algorithms, the technique needs to be applied to a system that has a-priori the set of vulnerable events defined, and the desirable behavior of it.

Immediate observations regarding the application of the mitigation technique, after the enforcement of opacity is done, warrant attention. For instance, as evident in Figure 3.8, particularly with respect to sub-figures *(b)* and *(d)*, the desirable behavior of system G and its representation post-mitigation, respectively. In sub-figure *(b)*, states $\{9, 10\}$ exist in the system's desirable representation but are eliminated after the mitigation technique is applied. Initially, it was considered crucial to keep the Secret State within the system during mitigation, as its removal could jeopardize the primary objective of maintaining opacity. However, as the technique evolved, it was concluded that the removal of the Secret State does not pose a significant problem. Therefore, the Secret State does not require special consideration or protection during mitigation.

Evolution of Procedures and System Adjustments

This subsection regards some changes that were considered at an initial development phase, for the successful application of the technique. Since this material and considerations lead to the creation of the final technique, it's of great importance to include them.

To transition from enforcing CS Opacity to implementing the attack mitigation technique, the resulting system obtained from the former will be regarded as the desired behavior of the system. From the results of the procedure *FIND-AIC-O*, the Supervisor will always opt for the transformation that keeps most of the functionalities of the original system.

The transformation will then be regarded as the desirable functioning of the system. As mentioned, the mitigation technique considers three types of states, desirable, tolerable, and faulty. This is something that is intricate in the algorithm developed in the literature, so it's useful if maintained for the proposed technique of this dissertation. However, the grading of these states based on risk levels may not be as useful, where a faulty state is considered a state of risk for the system. For this dissertation, the approach will be changed regarding how the state may

reveal opacity, following the nomenclature of:

- **Desirable State:** A state present in the desirable behavior of the system.
- **Tolerable State:** A state that is not included in the desirable behavior of the system, but none of the transitions that depart from it violate CS opacity.
- **Disclosing State:** A state that is not included in the desirable behavior of the system, and one of the transitions that depart from it will violate CS opacity.

The decision to utilize the algorithm from [Yao et al., 2020] derives from its suitability for integration into a system featuring three distinct state types. By accommodating these adaptations, the algorithm remains applicable without necessitating the addition of risk states to the original system. This alignment with the dissertation's objective underscores its significance.

Additionally, it is pertinent to address the treatment of the secret state within the system. As previously outlined, the secret state is predefined before the implementation of this security technique, and during the application of the technique, this state can be removed from the system.

4.2.1 First Application of the security technique and its Discoveries

The following subsection aims to explain how the technique started to be implemented and the general idea behind said technique and the requirements. To make the application of this phase of the technique easier to understand, a formal application regarding the technique to a specific system will be done in the end. With this application, the drawbacks will be exposed, as well as the modifications needed to create and reach the final's technique form.

When implementing the technique, the following steps will be executed to achieve a system that maintains current-state opacity, even when confronted with an active attacker. It is crucial to emphasize that this technique does not necessitate the detection of an active attacker. Instead, it is proactively applied to a system before deployment. Consequently, the final system configuration will serve as the new system representation, ensuring its resilience against active attackers while upholding opacity.

The first step involves applying the Bipartite Transition System, which categorizes the system using Information States. This is accomplished by employing a bipartite structure comprising two types of states. Transitioning from a Z-state to a Y-state signifies an observable transition. In contrast, transitioning from a Y-state to a Z-state indicates an unobservable reach and preserves the set of enabled events from the preceding Y-state. This implies that $I(z)$ captures the set of states reachable from a preceding Y-state through enabled unobservable event sequences, while $\Gamma(z)$ records the control decision made in the preceding Y-state.

To advance with the application of the technique, Algorithm 1 from [Meira-Góes et al., 2023] must be utilized. This algorithm requires two inputs: the system under evaluation and the *OP* function. The *OP* function is a binary function that operates on the Information states of the system. If an information state belongs to the set of secret states X_S, the function returns 0; otherwise, it returns 1.

Algorithm 1 FIND-AIC-O

```
1:
2: procedure FIND-AIC-O(G, OP)
3:     AICO.Y ← {y_0}, AICO.Z ← ∅, AICO.h ← ∅
4:     DoDFS(G, y_0, AICO, OP)
5:     Prune(AICO)
6:     AICO ← Accessible(AICO)
7: end procedure
8:
9: procedure DoDFS(G, y, AICO, OP)
10:    for γ ∈ Γ do
11:        z ← h_YZ(y, γ)
12:        if OP(I(z)) = 1 then
13:            AICO.h ← AICO.h ∪ {(y, γ, z)}
14:            if z ∉ AICO.Z then
15:                AICO.Z ← AICO.Z ∪ {z}
16:                for e ∈ γ ∩ E_o do
17:                    y' ← h_ZY(z, e)
18:                    AICO.h ← AICO.h ∪ {(z, e, y')}
19:                    if y' ∉ AICO.Y then
20:                        AICO.Y ← AICO.Y ∪ {y'}
21:                        DoDFS(G, y', AICO, OP)
22:                    end if
23:                end for
24:            end if
25:        end if
26:    end for
27: end procedure
28:
29: procedure PRUNE(AICO)
30:    while exists Y-state in AICO that has no successor do
31:        Delete all such Y-states in AICO and delete all their predecessor Z-states
32:    end while
33: end procedure
```

After applying algorithm 1, the output is the AIC-O of the system, indicating the desirable behavior while maintaining the system's Current-State opacity when there is a passive attacker present. Subsequently, this initial step identifies the preferred behavior of the system, ensuring opacity in its Current-State. During the application of this algorithm, procedure **DoDSF** will explore all the combinations of Control Decisions a supervisor can make, regarding all the Y-states, and the **Prune** procedure will eliminate all the changes required, to obtain the AIC-O.

Algorithm 2 Online Mitigation M

Input: AE-attacked model $G_a = (X, \Sigma_a, \delta_a, x_0)$ with $\Sigma_{uc,a}$, observation mapping P_C
Output: control decision γ at each instant
$\alpha \leftarrow \epsilon$
$\hat{\mathcal{E}}(\alpha) \leftarrow \{x_0\}$
$\gamma \leftarrow \text{ChooseAction}(\hat{\mathcal{E}}(\alpha))$
defense by disabling events not in γ
$\mathcal{E}(\alpha) \leftarrow \text{UR}_\gamma(\hat{\mathcal{E}}(\alpha))$
while new event $\sigma \in \Sigma_o$ is observed **do**
    $\alpha \leftarrow \alpha\sigma$
    $\hat{\mathcal{E}}(\alpha) \leftarrow \text{NX}_\sigma(\hat{\mathcal{E}}(\alpha))$
    $\gamma \leftarrow \text{ChooseAction}(\hat{\mathcal{E}}(\alpha))$
    defense by disabling events not in γ
    $\mathcal{E}(\alpha) \leftarrow \text{UR}_\gamma(\hat{\mathcal{E}}(\alpha))$
end while

Algorithm 3 CHOOSEACTION

Input: estimate $\hat{\mathcal{E}}(\alpha)$, controllable events $\Sigma_{a,c}$, and attacked system $\tilde{G}_a = \left(X, \Sigma_a, \tilde{\delta}_a, x_0\right)$
Output: control action γ
$\gamma \leftarrow \Sigma_{uc,a}$
$i \leftarrow 0$
EventList $\leftarrow \Sigma_{a,c}$
while EventList $\neq \emptyset$ and $i <$ |EventList| **do**
    $\sigma \leftarrow$ EventList$[i]$
    if $\text{UR}^+_{\gamma\cup\{\sigma\}}(\hat{\mathcal{E}}(\alpha)) \cap X_{Dis} \neq \emptyset$ **then**
        EventList $\leftarrow$ EventList$\backslash\{\sigma\}$
    else
        if $\exists x \in \text{UR}_\gamma(\hat{\mathcal{E}}(\alpha)) : \delta(x, \sigma) \in X_D$ **then**
            $\gamma \leftarrow \gamma \cup \{\sigma\}$
            EventList $\leftarrow$ EventList$\backslash\{\sigma\}$
            $i \leftarrow 0$
        else
            $i \leftarrow i + 1$
        end if
    end if
end while

Considering the application of the first algorithm, it is feasible to consider that the system protects its secret against the existence of a passive attacker. The remaining application of the technique is applied when the passive attacker gains some influence over the system and gets control over some of the events. When this situation is created in the system, the application of Algorithms 2 and 3 will ensure that the system can still be secured.

When applying algorithms 2 and 3, the requirements previously mentioned need to be present in the system. Therefore, when the set of vulnerable and defendable events is defined, the application of said algorithm can be done. The definitions and organization of the sets of states is partitioned as follows, and the formal definition of sets is the equal to the one present in Algorithm 3.

- **Desirable State:** A state present in the desirable behavior of the system, included in the set X_D.
- **Tolerable State:** A state that is not included in the desirable behavior of the system, but none of the transitions that depart from it violate CS opacity. These are included in the set of states X_T.
- **Disclosing State:** A state that is not included in the desirable behavior of the system, and one of the transitions that depart from it will violate CS opacity. The set is represented by X_{Dis}.

From the application of the Online Mitigation M algorithm, the output will be a limited version of the system that will sustain an active attack and won't reveal the secret state of the system. It is important to reinforce the idea that this application is limited to the set of vulnerable, defendable and the desirable behavior returned from Algorithm. The results obtained will be different if any of these change.

Example for the application of the technique

To apply the technique, the following system can be considered. Considering system G present in chapter 3, more specifically in Subsection 3.3.1 where the analysis of the work done in [Yin, 2015] is done. If we consider system G with the same characteristics, the result obtained in the AIC-O from Figure 3.7 will be desirable behaviour of the system. From this, we know that the system is composed by the following sets of events and states.

- **Controllable and Uncontrollable Events:** $A_c = \{a, b, c\}$, $A_{uc} = \{o_1, o_2\}$. No special distinction using colours is needed.
- **Observable and Unobservable Events:** $A_o = \{o_1, o_2\}$, $A_{uo} = \{a, b, c\}$. No special distinction using colours is needed.
- **Vulnerable Events:** $A_V = \{a, b, c\}$. Identified using red.
- **Defendable Events:** $A_D = \{b\}$. Identified using green.
- **Desirable States:** $X_D = \{0, 3, 4, 5, 6\}$. Identified using black.
- **Tolerable State:** $X_T = \{1\}$. Identified using dark red.
- **Disclosing State:** $X_{Dis} = \{2\}$. Identified using blue.
- **Secret State:** $X_S = \{5\}$. Identified using red.

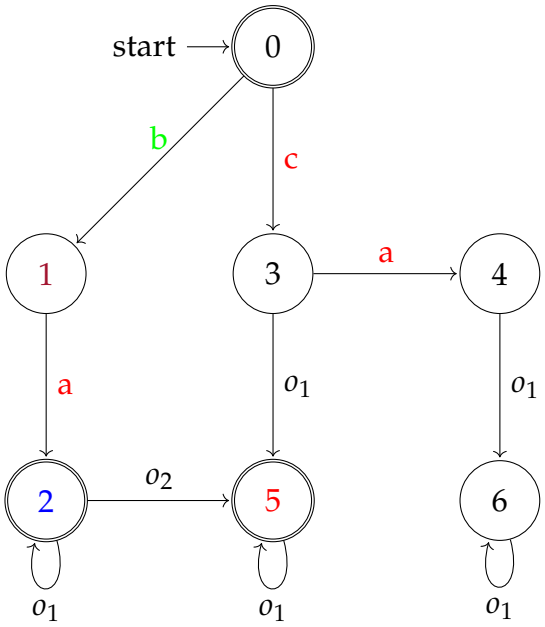


Figure 4.1: System G with the classification of States and Events

Figure 4.1 is the representation of system G considering the desirable behaviour obtained from the AIC-O and the set of events defined above. This representation includes the utilization of colours to improve the interpretation and the meaning of specific states and events, these colours are in accordance with the previous enumeration of sets of states and events.

From this representation, the application of algorithms 2 and 3 is easy. Starting from state 0, the algorithm will explore state 1 and state 2. When the algorithm evolves to state 1, a Tolerable state in the System, the system can only evolve with the occurrence of vulnerable event a. This means that since b is defendable, the system will immediately use the mitigation module M and defend event b, making it impossible to occur. The next state to examine is state 3, which is in the desirable behaviour and that is the same for the rest of the system, meaning that the system will not need to deactivate or defend any other events. This means that the final representation of the system, under the influence of an active attack and with the sets previously defined will have the following representation.

As observable in the figure above, system G can function in a limited way under the influence of an active attacker. The edge in orange means that the event associated with it has been defended by Mitigation Module M.

Applying the technique to a real Cyber-Physical System

Utilizing these three algorithms within the technique demonstrates that it is possible to defend a system from an Actuator-Enablement Attack without revealing the system's secret. As shown in Figure 4.2, the results indicate that the system can be effectively protected while maintaining the confidentiality of the secret state.

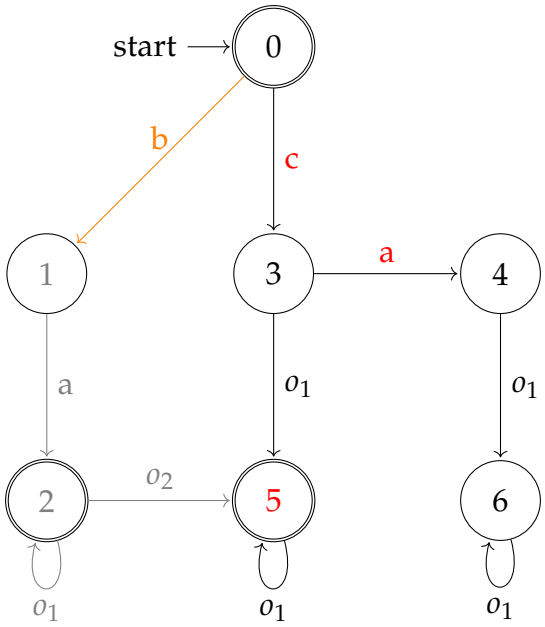


Figure 4.2: System G defended against an Active Attacker using the Mitigation Module M

Since system G is not associated with a real Cyber-Physical System, some assumptions made for the technique do not have significant repercussions. However, when applied to the simpler Hydra System, mentioned in Chapter 5, some issues with this approach become evident.

The first and most notable implication is that the technique, at this stage, requires the system's current state to be state 0 when an attacker is detected. In a system like Hydra's, this assumption is not feasible because the system changes states based on the occurrence of cyber and physical events. For example, if the system's current state is state 4 when an attacker is detected, it would result in the immediate loss of all water from its tanks, which is not practical.

Another issue is that algorithms 1, 2, and 3, when combined, unnecessarily traverse the entire system more than once. This can be resolved by utilizing the output of procedure **DoDFS** and making a slight modification to it when applied within Algorithm 1. The output can then be used to determine defensibility, which will be explored in the next subchapter. This approach demonstrates how Algorithms 2 and 3 can be omitted, thereby reducing the time complexity of the technique's application. This is ought to be explained in the next subsection.

4.2.2 Changing the Approach for the Technique

As mentioned in the previous subsection, the proposed technique has some flaws related to the current state of the system and the temporal complexity of the algorithm. Therefore, these issues will now be mitigated.

With this new approach, the assumptions regarding the current state of the system when a new attacker is detected can be relaxed. This allows the system to

be characterized based on the limitations and type of attacker present. The following list characterizes the three modes of operation that a system can now be operating in:

- **Normal:** The system is configured to be functioning in its original configuration. There is no passive or active attacker present, so the system will not reveal the secret to an external observer.

- **Passive Attacker:** The system is now configured so that its secret will not be revealed to the passive observer. This is done through the implementation of a Supervisor S, by enforcing Current-State Opacity.

- **Active Attacker:** The system will enter its most restrictive behaviour, with the activation of Mitigation Module M. By having both Supervisor S and Mitigation Module M active, the system will mitigate an active attack's repercussions, such as revealing the Secret State of the System.

Regarding the second problem related with the proposed technique, as mentioned, there are redundant passes through the system, to defend it. This way, algorithm 1 needs to be changed, to fulfill the premises. The first change is regarding Algorithm 1, more specifically, procedure **DoDFS**. Given that the procedure is useful because it can traverse the whole system, the only thing to be changed in it, is line 12. As previously explained, function OP will return 1 if $I(Z) \in X_S$, since the objective is to traverse the whole system, the if condition associated is therefore removed. Due to the removal of this line in the code, the Prune procedure will not be necessary, since the system will not need to be analysed for the existence of Y-states without a successor Z-state, since it will not be possible due to the changes made.

From the application of this algorithm, the output of procedure DoDFS will be sufficient to successfully defend the system, therefore Algorithms 2 and 3, can be removed from the technique, and consequently the grading given for all types of states, such as , **desirable**, **tolerable**, and **disclosing**. The application of the technique will only need the existence of the Mitigation Module M, that will decide its actions based on the systems current state and the sets of **controllable** and **defendable** events.

This way, using system G that was used in the previous subsection 4.2.1, and keeping the same characteristics, the output from the refined proposed technique will be the following.

Algorithm 4 FIND-AIC-O

```
1:
2: procedure FIND-AIC-O(G, OP)
3:     AICO.Y ← {y0}, AICO.Z ← ∅, AICO.h ← ∅
4:     DoDFS(G, y0, AICO, OP)
5: end procedure
6:
7: procedure DoDFS(G, y, AICO, OP)
8:     for γ ∈ Γ do
9:         z ← hYZ(y, γ)
10:        AICO.h ← AICO.h ∪ {(y, γ, z)}
11:        if z ∉ AICO.Z then
12:            AICO.Z ← AICO.Z ∪ {z}
13:            for e ∈ γ ∩ Eo do
14:                y' ← hZY(z, e)
15:                AICO.h ← AICO.h ∪ {(z, e, y')}
16:                if y' ∉ AICO.Y then
17:                    AICO.Y ← AICO.Y ∪ {y'}
18:                    DoDFS(G, y', AICO, OP)
19:                end if
20:            end for
21:        end if
22:    end for
23: end procedure
```

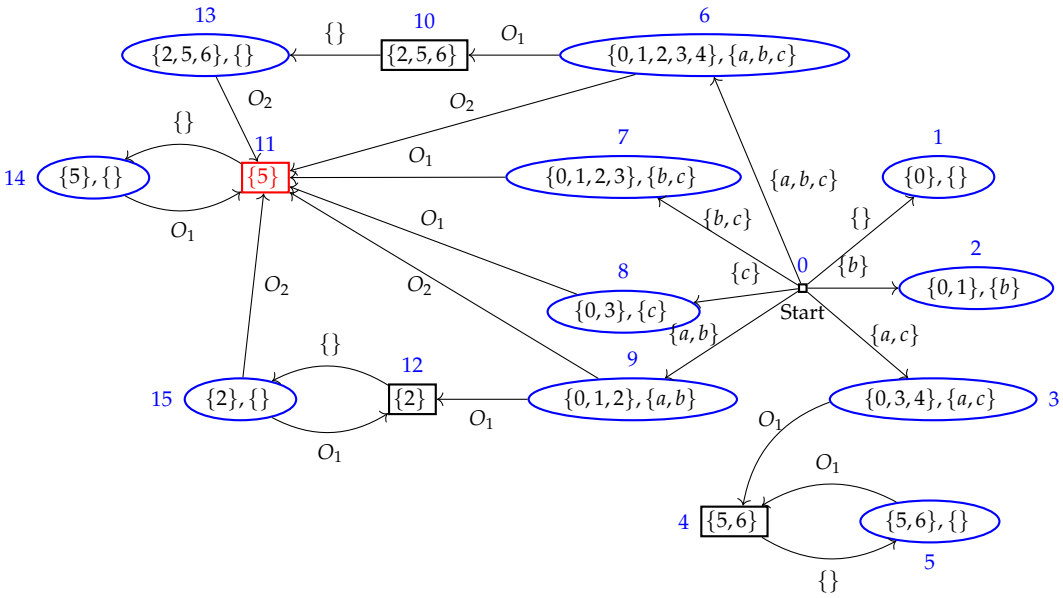


Figure 4.3: Application of the refined Technique to System G

When comparing the results obtained in Figure 4.3 above with the results in Figure 3.4, the only difference is the existence of a new Z-state, derived from the Y-state that contains the system's secret state, state 5. This change is due to the modifications made in procedure **DoDFS**, which ensure that the output includes the Y-state with the secret state and the remainder of the system that would oth-

erwise not be visited.

Assumption 1. *To ensure the technique's feasibility based on the current state of the system, it must be assumed that there is at least one control decision preventing the system from evolving towards revealing the secret. In other words, the secret state cannot be part of the path dictated by any control decision. If no such control decision exists, neither the Supervisor nor the Mitigation Module M can protect the system.*

Although not directly connected with the defense technique, Algorithm 5 includes a possible procedure description that can be used to identify the set of Current-States that follow Assumption 1. This algorithm will remove all the states that are not deemed possible for the systems current-state when an attacker is detected.

Algorithm 5

1: **procedure** IDENTIFY POSSIBLE CURRENT-STATES($AICO$)
2: **while** exists a Y-state in $AICO$ that is in the set of Secret-States or a Y-state that has no successor **do**
3: **if** state $\in X_S$ **then**
4: Delete all such Y-states in $AICO$, delete the successor Z-state with control decision $\{\}$ and delete all the predecessor Z-states;
5: **else**
6: Delete all such Y-states in $AICO$ and delete all their predecessor Z-states;
7: **end if**
8: **end while**
9: **end procedure**

Given the output in Figure 4.3, Assumption 1, and Algorithm 5, the system system's reaction to the attacks is described below.

When in the presence of a **Passive Attacker**, the AIC-O provides a map for a system administrator to issue control decisions. Based on Assumption 1, the set of possible current states is $X_{CS} = \{0, 1, 2, 3, 4, 5\}$. To obtain this set of states using Algorithm 5, the first state to be marked as a state to be avoided is state 11, followed by states 13, 14, 15, 7, and 8. States 2 and 10 are the marked next, as these will lack a successor Z-states that is not marked, followed by states 10 and 12, which are the respective predecessor Z-states of these Y-states. A visual representation of the system can be observed in Figure 4.4

The remaining states will uncontrollably reveal the secret state. The possible control decisions for the supervisor are: $\{\{\}, \{b\}, \{a, c\}\}$, which ensure that the system remains safe from revealing its secret.

In the presence of an **Active Attacker**, the approach considers the control decisions used before the attacker was active. The next step is identifying vulnerable and defendable events. A vulnerable event that is not defendable must be considered active in the control decision, while a defendable event can be disabled. Given the set of defendable events $A_D = \{c\}$ and the set of vulnerable events $A_V = \{b, c\}$:

Considering the possible current states X_{CS}, previously defined, event c has to be defended, as shown in Figure 4.5. Control decisions $\{\{a, b, c\}, \{b, c\}\}$ will lead to

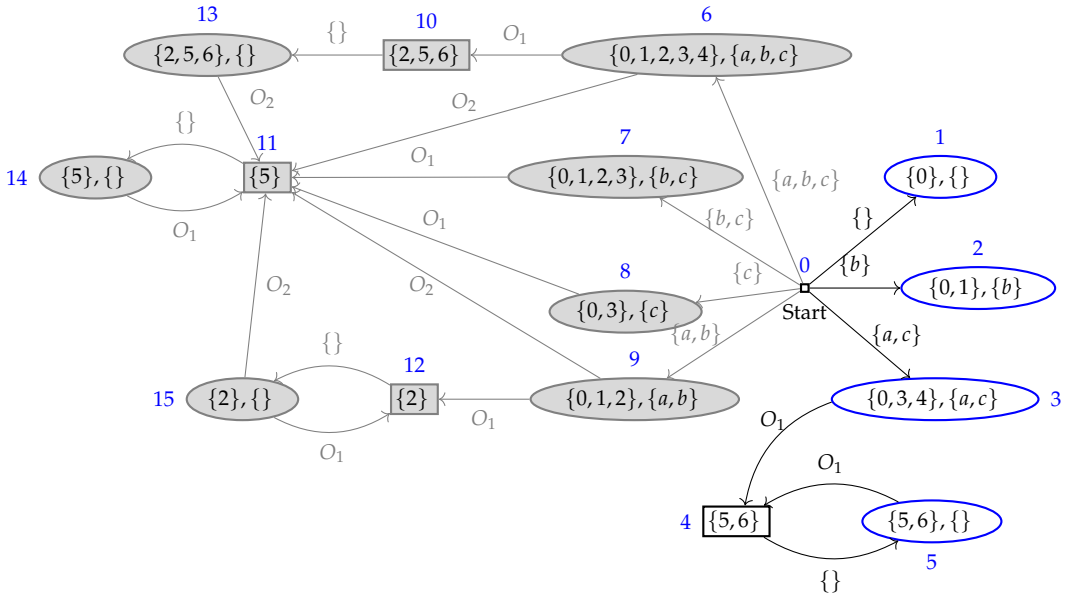


Figure 4.4: System G with marked states

marked states if event c is not defended when the system is in state 0. This is the only state that issues any control decisions in this system. In this particular case, although the system is left with an active control decision b, which is under the influence of the active attacker, nothing detrimental can happen to the system by leaving it active, since the transition is to state 2, which is a final Z-state.

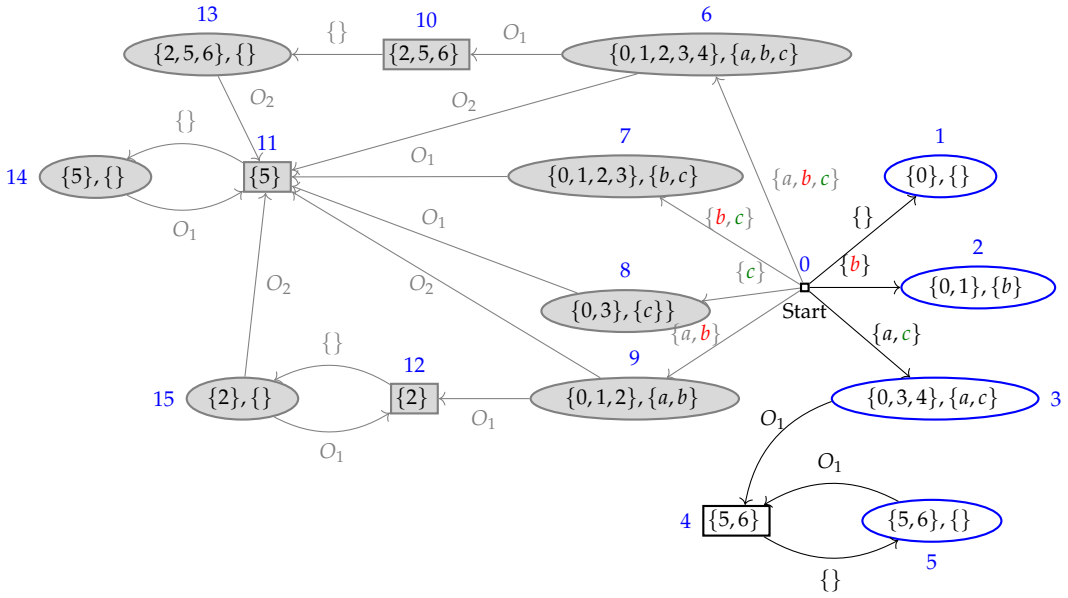


Figure 4.5: System G with vulnerable events in red and defendable events in dark green.

This application successfully proves that it is possible to successfully secure the system, while relaxing the assumptions for the systems current state to be the initial state of the system, and reducing the time complexity substantially.

The next section will formally present the technique as this subsection was its deduction and a simple proof of concept. A practical application of this technique is provided in chapter 5, where it is shown how it can be applied to a cooling system as the Hydra adaptation in Figure 5.3.

4.3 Securing a DES with Opacity Against Active Attackers

4.3.1 Overview

The technique developed for the dissertation, as previously mentioned, aims to secure a DES system that is not inherently opaque through the usage of current-state opacity against the influence of active attackers. Such technique is a novelty in the field of opacity, as there are not many approaches that study the integration of an active attacker when studying the opacity, as explained in Chapter 3.

4.3.2 Integrating the Technique

Preliminaries

To apply the technique, the starting ground will be to have the following requirements in the system. There is a system, that can be modeled using a DFA, denoted as $G = (X, A, \lambda, X_0)$, with X being the finite set of events, A denoting the alphabet of events, and λ representing the transition function. Here, $\lambda(x, \theta) = y$ signifies that there is a transition labeled by event θ from state x to y. Additionally, X_0 is a subset of or equal to X ($X_0 \subseteq X$), serving as the set of initial states of the system. As the technique will make system G be controlled by a *supervisor*, that will dynamically enable/disable events of the system so that the post-technique specification is achieved. Therefore, the alphabet A is partitioned into two disjoint subsets: $A = A_c \cup A_{uc}$, A_c representing the set of controllable and A_{uc} the set of uncontrollable events, the rules presented in Chapter 2 are followed rigorously throughout the implementation of the technique. The system is also assumed to be partially observed, having alphabet A partitioned into two other disjoint subsets, $A = A_o \cup A_{uo}$, A_o representing the set of observable and A_{uc} the set of unobservable events. The natural projection $P : A^* \rightarrow A_o^*$, is defined as

$$P(\epsilon) = \epsilon \text{ and } P(s\sigma) = \begin{cases} P(s)\sigma, & \text{if } \sigma \in A_o, \\ P(s), & \text{if } \sigma \in A_{uo} \end{cases}$$

, since a supervisor can only base decisions on observations, a partial-observation supervisor can be defined as a function $S_P : P(L(G)) \rightarrow \Gamma$. The notation S_P/G is used to represent the controlled system and the language generated is denoted by $L(S_P/G)$.

The model of the Actuator Enablement Attack (AE-attack) presupposes that $A_v \subseteq A_c$ comprises a set of vulnerable actuator events, which may be either observable or unobservable. This implies that an attacker possesses the capability to enable an event, even if the supervisor has disabled it. The set of attacked actuator events is delineated by $A^a_{c,v} = \sigma^a : \sigma \in A_v$, and $A_a = A \cup A^a_{c,v}$ encapsulates all attacked events. Here, σ^a symbolizes the occurrence of event σ initially disabled by the supervisor but subsequently enabled by the attacker.

For the technique it is also assumed that A_d is the set of *defendable events*, where $A_d \subseteq A_v \subseteq A_c$. Meaning that the events in A_d are vulnerable in the original closed-loop system, but the existence of the mitigation module may these to be defended, meaning that the controllability of events can be partitioned into three categories:

- $\mathbf{A_c} \setminus \mathbf{A_v}$ - events that can be deactivated directly by the supervisor in the usual manner, as they are not susceptible to attack.
- $\mathbf{A_v} \setminus \mathbf{A_d}$ - events that cannot be consistently disabled; in other words, for any $\sigma \in A_v \setminus A_d$, σ^a can always be enabled by the attacker, even when σ is disabled by the supervisor.
- $\mathbf{A_d}$ - Events that can be effectively disabled (or defended) by the mitigation module, despite potentially at a higher cost.

Mathematically, the mitigation module is represented by M.

When applying the technique, the sets previously defined need to be established and the systems current-state needs to comply with Assumption 1. It is then possible to apply the technique.

As explained in the previous section, the system has 3 modes of operation, a brief overview of each is:

- **Nominal:** Original system configuration with no attackers present, ensuring secret remains unrevealed.
- **Passive Attacker:** System configured with Supervisor S to enforce Current-State Opacity, preventing secret disclosure to passive observers.
- **Active Attacker:** System activates Mitigation Module M alongside Supervisor S to mitigate the impact of active attacks, such as preventing disclosure of the Secret State.

The algorithm to be applied to the system is depicted in Algorithm 4, and the output will be the AIC-O of the system. Using the AIC-O, the system's administrator can adjust control decisions based on the type of attacker and the predefined sets of events.

It is important to emphasize that this technique does not need to be applied to the system when an attack is detected; it only needs to be enforced. This means

that to obtain the AIC-O, the person in charge of defending the system can apply Algorithm 4 while the system is operating under normal conditions, as the sets of events are predefined. This way, when a passive attacker is detected, the supervisor can take the control decisions that avoid states that would reveal the secret. These are the states not included in the set of possible current states, X_{CS}, obtained by applying Algorithm 5 to the AIC-O, meaning that the system cannot transition to them.
When the attacker becomes active, the approach remains similar. By knowing the sets of vulnerable and defendable events, the system will always enable control decisions involving non-defendable vulnerable events, as these cannot be deactivated. It is important to note that the system's defendability against an active attacker depends on the sets of defendable and vulnerable events. For instance, in the scenario shown in Figure 4.3, if the set of vulnerable events is $A_V\{a, b, c\}$ and $A_D\{c\}$ is the set of defendable events, it will not be possible to defend the system. In state 0, the possible control decisions are, $\{\{a, b, c\}, \{a, b\}\}$ leading the system to naturally transition to states that will reveal the secret.

Chapter 5

Case Study

5.1 Hydra System

5.1.1 Introduction

The HYDRA testbed serves as a compact model of an automated water distribution system, incorporating a physical plant with structures, tanks, and connecting pipes. Additionally, it features a control system equipped with sensors, actuators, and microcontrollers. It's essential to note that the HYDRA testbed diverges from a system regulation model, where physical quantities aim to achieve predicted reference values continuously. Instead, it operates as an event-driven system, with occurrences taking place at irregular intervals that are not predetermined. Due to the unpredictable timing of events, the HYDRA testbed can be effectively described using DES.

This testbed, originally developed for a project by a PhD student, will undergo examination in the presence of an active attacker. The study will specifically focus on the utilization of opacity to safeguard certain aspects of the system under the described attack conditions. Notably, all the figures used for this section were generously provided by the student as part of the project.

5.1.2 Physical Model

The model consists of three tanks, each with the same capacity. A reservoir (S) at the base level simulates an infinite source of water.

- Placed at the highest level, Tank 1 (T1) has a connecting pipe allowing gravity-driven water flow to Tank 2 (T2).
- Connected through a pipe at the same height, T2 and T3 behave like communicating vessels, influencing water flow based on the water level in each tank.

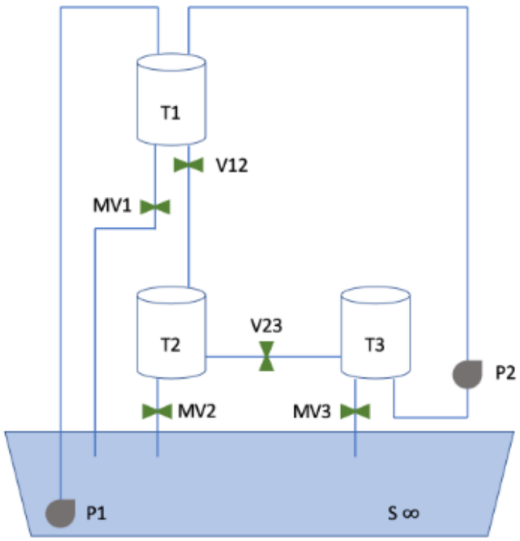


Figure 5.1: Hydra System Physical Configuration

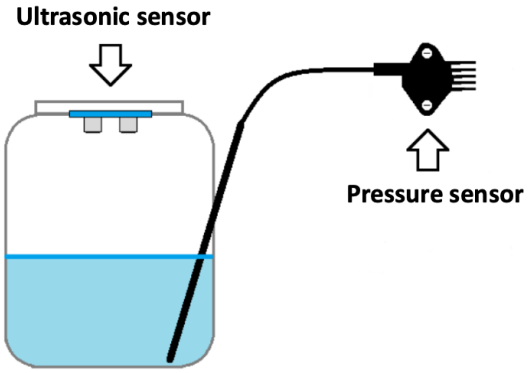


Figure 5.2: Sensors in the Water Tanks

- The final tank, T3, has a connecting pipe enabling water flow from T3 to T1, facilitated by an external pump.

Each tank is equipped with two types of sensors:

- An ultrasonic sensor placed on top of the tank measuring the water level.
- A pressure sensor measuring the air pressure created by the fluid in which the sensor is immersed.

The passage of water from T1 to T2 is regulated by an electromechanical valve (**V12**). Another electromechanical valve (**V23**) regulates water flow between T2 and T3.

Both V12 and V23 can be opened or closed mechanically.

Tank T1 receives water through two ways:

- A pump (**P1**) inside reservoir S.

- A pump (**P2**) taking water from tank T3.

Both P1 and P2 can be turned on or off.

There is also a flow meter between T3 and T1 measuring the flow rate of water passage.

Finally, each tank is equipped with pipes with manual valves (**MV1**, **MV2**, **MV3**) that can be opened or closed manually. These are used in safety cases or as fault simulations.

Figures 5.1 and 5.2 are provided to help visualize how the system functions.

5.1.3 General Architecture

The general architecture of the system is provided to establish a clear definition, considering its detachment from the rest of the dissertation.

Sensors transmit measurements from the physical plant to an Arduino labelled "Sensori", which organizes the data into a unified string structured as follows:

$$ultr1 * ultr2 * ultr3|pres1 * pres2 * pres3|media1 * media2 * media3|flow \quad (5.1)$$

The data is then transmitted serially to the Raspberry labelled "Sensori." This Raspberry device is responsible for extracting information from the received string, encrypting the data, and subsequently retransmitting it within a Modbus TCP packet. This process ensures that the values travel securely over the network. The encrypted data are received by another Raspberry labelled "Decrypting," which possesses the capability to decipher the information. The decrypted data are then forwarded to the PLC by encapsulating them into a new Modbus TCP packet.

In order to transmit data, the Raspberry Sensori functions as a Modbus client, establishing a connection with the server running on the Raspberry Decrypting/Encrypting. The Raspberry Decrypting also establishes a new client-server connection with the PLC, in which it operates as a client.

Initially, the Raspberry Decrypting/Encrypting queries the PLC regarding the operations to be performed. The PLC responds by sending information on the states to be assumed by the pumps and motorized valves, represented by Boolean values [1,0].

Subsequently, the Raspberry Decrypting encrypts the data and transmits it to the Raspberry Actuators. The latter is capable of deciphering the data and forwarding it to the Arduino Actuators, which is responsible for issuing commands to the physical plant.

5.1.4 Different Configurations

From the given physical system, two distinct configurations have been devised, representing alternative projects that can be loaded onto the PLC software.

3 Tanks Configuration: The primary objective of this configuration is to maintain a continuous flow of water within the three tanks while endeavoring to distribute it uniformly. This process unfolds in two main phases:

- **Set-up Phase:** In this initial stage, water is extracted from reservoir S and initiated into the distribution across the three tanks.
- **Cyclical Phases:** The subsequent phases involve repetitive cycles where a consistent quantity of water circulates among the multiple tanks.

2 Tanks Configuration: This configuration, involves the utilization of only tanks T1 and T3. Notably, its distinctive feature lies in the faster dynamics it exhibits compared to the 3 tanks configuration. As a result, it proves more suitable for assessing the system's responsiveness to potential cyber-attacks.

5.2 Adapting the Hydra System - A simpler approach

To create a simpler environment in which the Hydra System could be applied, the following environment was created. Considering Figure 5.1, and changing so that T2 and T3 are merged into a single tank, T2, that has twice the capacity of T1. In this situation, the following description of the system normal functioning is considered.

Identifier	*Meaning*
T1	Tank 1
T2	Tank 2, twice the size of T1
V1	Valve 1, connects T1 to T2
P1	Pump 1, connected to T1 from the outside
P2	Pump 2, connected to T1 from T2

Table 5.1: Information about components.

The system is deployed to grant the water supply for a cooling system. It is assumed that the external source of water is limited, so the usage of P1 is restricted. The set of states is $X = \{0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17\}$, the set initial states is $X_0 = \{0\}$, the alphabet $A = \{a, b, c, d, e, f, g, h, i, j\}$, and the transition function depicted in Figure 5.3. The set of controllable events is

$A_c = \{a, c, d, f, h, j\}$, $A_o = \{b, e, g, i\}$ is the set of observable events, and the secret states are in set $X_S = \{5, 11\}$. The selection of secret states was based on the characteristics of the system in these states. In states 5 and 11, T_1 is at the lowest capacity allowed in the system, 10%, while V_1 remains open. Consequently, the tank is losing water in both states, making these the riskiest states of the system. Therefore, these states were chosen.

To better understand the system's normal functioning, the following description outlines its typical evolution, emphasizing the sequence of steps over the various states of the system. The meaning of each state is also defined in Table 5.4.

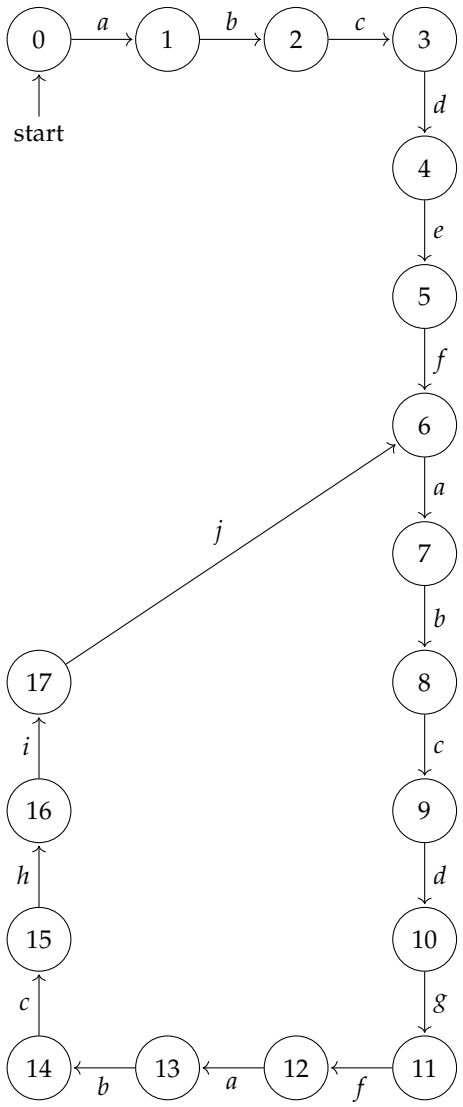


Figure 5.3: DES of a simpler Hydra System.

State	T_1	T_2	V_1	P_1	P_2	e_i
0	0	0	0	0	0	a
1	0	0	0	1	0	b
2	100	0	0	1	0	c
3	100	0	0	0	0	d
4	100	0	1	0	0	e
5	10	45	1	0	0	f
6	10	45	0	0	0	a
7	10	45	0	1	0	b
8	100	45	0	1	0	c
9	100	45	0	0	0	d
10	100	45	1	0	0	g
11	10	90	1	0	0	f
12	10	90	0	0	0	a
13	10	90	0	1	0	b
14	100	90	0	1	0	c
15	100	90	0	0	0	h
16	100	90	0	0	1	i
17	10	45	0	0	1	j
6	10	45	0	0	0	-

Figure 5.4: Meaning of the states of the Hydra Adaptation.

The system operates with two tanks, Tank 1 (T_1) and Tank 2 (T_2), both initially empty. The sequence begins with Pump 1 (P_1) activating in State 1, filling T_1 until it reaches full capacity by State 2. P_1 deactivates in State 3, followed by Valve 1 (V_1) opening in State 4, allowing water from T_1 to flow into T_2 by State 5. Once

90% of T_1's water has transferred to T_2, V_1 closes in State 6, stabilizing T_1 at 10% and T_2 at 45% capacity.

The cycle repeats with P_1 activating again in State 7, filling T_1 to 100% in State 8, and then deactivating in State 9. V_1 opens in State 10, allowing T_1 to empty to 10% while T_2 increases to 90% by State 11. V_1 then closes in State 12, stabilizing T_1 at 10% and T_2 at 90% in readiness for the next cycle.

The second phase involves P_1 re-engaging in State 13, refilling T_1 to 100% by State 14, and then deactivating in State 15. P_2 opens in State 16, allowing Pump 2 (P_2) to return water from T_2 to T_1, with the system stabilizing in State 17 at T_1: 10%, T_2: 45%, and turning it off, bringing the system to State 6, and repeating the operational sequence.

Overall, the system cycles through a series of steps. P_1 fills T_1, and then V_1 transfers water from T_1 to T_2. This process ensures that T_1 and T_2 reach the correct levels before stabilizing. Each cycle involves controlled movements of water through pump activation, valve openings, and closures, maintaining a consistent pattern between the two tanks.

5.3 Defending the Hydra System with the Proposed Approach

This section is dedicated to the application of the security technique to the Hydra system that has been described in previous section 5.2. As explained in Chapter 4, an attacker does not need to be detected, for Algorithm 4 to be applied. Based on this, the following subsection will explain how a system administrator would react to the attackers, and how Algorithms 4 and 5 are applied and the usefulness.

The application of this technique in a system like the adapted Hydra system is important because the technique is based purely on a theoretical approach. This system will demonstrate the limitations of such a technique. In this specific system, all events associated with the sets of controllable and observable events are applicable in a real-world context. Events related to physical changes in the system are in the set of controllable events, while events related to sensor readings are in the set of observable events.

In a large-scale Hydra system, if there is no visual contact with the system's pumps and valves, it is possible to turn these on or off, but the physical process remains unobservable. In contrast, sensor readings cannot be controlled when the values reach the desired threshold, but these values are observable, for example, through a Human Machine Interface (HMI).

For this technique, the main idea is to have a real system, create a DFA model of the system, define the sets of events, and apply algorithms 4 and 5. By applying the technique, a system like a Supervisor will know the control decisions to have enabled and disabled, based on the system's current-state and the type of active attacker. The Mitigation Module M that is referenced in Chapter 4, can be viewed

like a physical switch for the deactivation of a controllable event, such as turning on a Pump or opening a valve.

5.3.1 Nominal Functioning

It is considered that the system is functioning normally when there are no attackers present. The system's responsible personnel, having all the necessary information, may apply the technique to obtain the AIC-O of the simpler Hydra system. To achieve this, Algorithm 4 needs to be applied using the previously defined sets of events and states, these being:

- The set of states is $X = \{0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17\}$.
- The initial state set is $X_0 = \{0\}$.
- The alphabet is $A = \{a, b, c, d, e, f, g, h, i, j\}$.
- The set of controllable events is $A_c = \{a, c, d, f, h, j\}$.
- The set of observable events is $A_o = \{b, e, g, i\}$.
- The secret states are in the set $X_S = \{5, 11\}$.

After obtaining the AIC-O of the system, there is nothing for the system or the administrator to do, since no attackers have been detected.

5.3.2 Passive attacker in the system

When a passive attacker is detected in the system, Supervisor S must immediately enforce opacity. This is achievable due to the existence of the AIC-O and Assumption 1, which guarantee the supervisor's control through the system's control actions. The system's current state must be within the set of events output by Algorithm 5; otherwise, the Assumption is violated. The algorithm is also useful when combined with the Supervisor, as it identifies states that need to be avoided.

For example, in the Hydra System, the algorithm will identify secret-state 7, state 8, and state 6 for avoidance, followed by secret-state 15, state 16, and state 14. Since state 3 and state 11 have other Y-state successors, they remain active in the system. Thus, all states are potentially current states of the system, except those specifically marked for avoidance. Using this information, the Supervisor S deactivates control decisions leading to these states, as illustrated in Figure 5.6.

When the supervisor is active in the system and the attacker is passive, it ensures that the system's secrets will not be violated. Depending on the current state, certain functionalities can be preserved, as the application of the technique is not overly restrictive.

From this system state, the system can either return to its original behavior or the attacker can escalate to an active type.

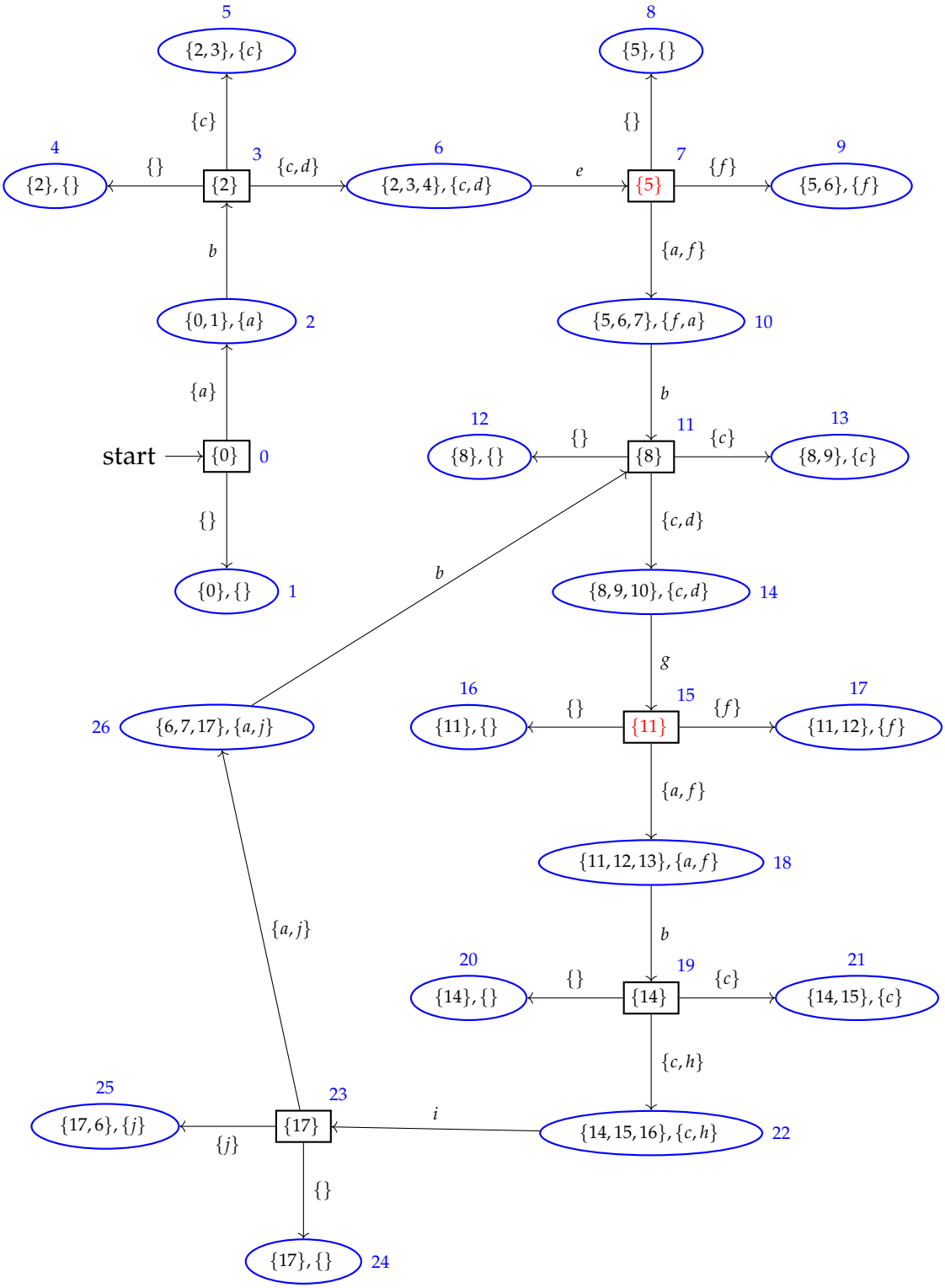


Figure 5.5: AIC-O Hydra System

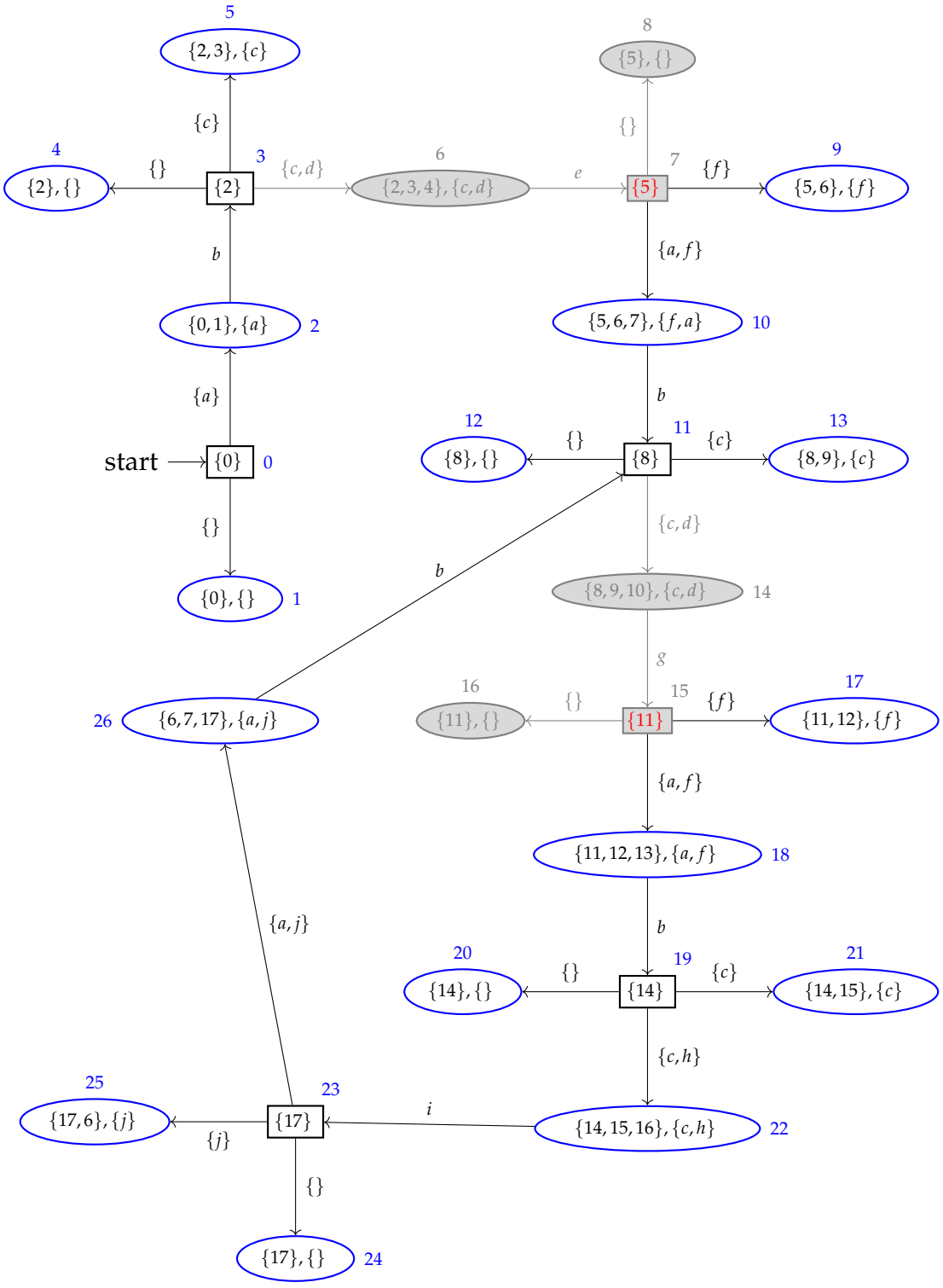


Figure 5.6: AIC-O of the Hydra System with marked States and Transitions

5.3.3 Active attacker in the system

For the purposes of this technique, it is assumed that the system's attacker may gain influence while initially passive, eventually becoming active. As previously explained, when such an event occurs, two new sets of events come into consideration: E_V for vulnerable events and E_D for defendable events.

When the Supervisor enforces opacity, some control decisions may be compromised depending on disabled events in the presence of the active-attacker. Therefore, with both sets of events defined, the procedure is as follows:

1. Every event that belongs to the set of vulnerable events E_V but does not belong to the set of defendable events E_D must always be considered enabled, because its disablement cannot be assured.
2. Any defendable event that needs to be deactivated becomes a task for the mitigation module. These events are specifically identified as those in E_D that are intended to be disabled to maintain system security, although to great cost to the system.

As previously explained, both sets E_V and E_D are defined for the technique. There is no mechanism that detects which specific events are in these sets. For this example, consider $E_V = \{a, c, d\}$ and $E_D = \{d\}$. The first step involves treating the vulnerable events as active. In this specific case, when applied to the AIC-O in Figure 5.6, the transitions leading to marked states are in danger because events c and d are now active.

It is known that event d is defendable, although at great cost to the system. Therefore, this event must be defended to prevent the secret from being revealed to the active attacker. Once these defensive actions are employed into the system, it will keep this configuration until the active attacker is removed from it, resuming then the normal operation.

5.4 Conclusions regarding the Case-Study and Future Work

This application of the technique successfully ensures that the active attacker cannot reveal the secret, thus maintaining the security and opacity in this system. Although the system's functionality is reduced, it is ensured that opacity is used against both passive and active attackers, which is a something new for the field, and the current-state of the system is not limited to the initial-state of the system, something that in a Physical system like the Hydra is not feasible .

The usage of the AIC-O is of great importance because it provides a map for the system's administrator to understand the control decisions being issued. A clear limitation of the technique arises when an active attacker is detected in the system. Due to the lack of literature on Intrusion-Detection Systems with DFA

and Opacity, and the fact that most literature defines the sets of vulnerable events, defining these sets was also mandatory for applying the technique. Although this approach is not feasible in practice, it remains an important and relevant field for future work. Future developments should aim to create a comprehensive technique that does not require predefined sets of events and can react to new types of attackers that have not been previously identified, such as using other sets of vulnerable events.

Another interesting future approach concerns the definition of defendable events and the relaxation of Assumption 1. The definition of a defendable event is currently done without any specific requirements or assumptions beforehand, which may lead to situations where the technique cannot effectively defend the system. Understanding the rules that could improve the criteria for an event to be declared defendable, thereby reducing these instances, is a promising direction for future research. Additionally, while Assumption 1 is less restrictive than the original works on which the technique is based, it still imposes significant constraints on the normal evolution of the system. Exploring ways to relax this assumption without compromising security could also enhance the technique's practical applicability.

5.5 Summary

Throughout this chapter, the focus was on applying a theoretical security technique to the Hydra test-bed. The Hydra system, initially introduced with its physical model, architecture, and various configurations, served as the backdrop for demonstrating the technique's applicability. The emphasis was on defending the system against both passive and active attackers using predefined sets of controllable and observable events.

The technique's effectiveness was explored in scenarios where normal system functioning, detection of passive attackers, and response to active threats were analyzed.

The chapter concluded by outlining the limitations encountered, such as the reliance on predefined event sets and assumptions about attacker behavior. Future work was suggested to enhance the technique's adaptability and responsiveness to emerging threats, aiming for more robust and autonomous defense mechanisms in dynamic system environments.

Chapter 6

Methodology and Planning

6.1 Introduction

This chapter delves into the systematic approach employed to address the research problem. It unfolds by detailing a phased work plan and providing an update on the progress made to date. The identification of risks takes center stage, where we consider their probabilities, potential impacts, and proposed mitigation. The meticulous planning and methodology outlined here lay a robust foundation for the ensuing exploration and analysis phases of the study. As this is being written for the intermediary delivery, this chapter is susceptible to change, nonetheless, there is an estimation of how the work is going to be divided, and the expected progress.

6.2 Methodology

6.2.1 Introduction

In this section, a methodology for addressing the identified problem in the dissertation is provided. The chosen research paradigm and methodology for this work is Design Science Research (DSR). A concise definition of how this methodology operates is presented, followed by its application to this dissertation. The utilization of DSR is expected to bring greater organization to the steps involved and enhance the understanding of the identified problem.

6.2.2 Design Science Research

Design Science Research (DSR) is a problem-solving paradigm that seeks to enhance human knowledge by systematically creating and applying innovative artifacts, [Brocke et al., 2020]. In essence, DSR strives to address real-world problems and enhance the environment in which these artifacts are implemented. This

research approach emphasizes the importance of practical solutions and the iterative development of tangible contributions to both knowledge and application domains. As outlined in [Brocke et al., 2020], the implementation of Design Science Research (DSR) projects has relied on various process models, with particular distinction given to the framework proposed in [Peffers et al., 2007]. This Design Science Research Methodology (DSRM) identifies six fundamental steps, each can briefly be defined as:

- **Problem identification and motivation** - This step defines the research problem and justifies the need for a solution, motivating both the researcher and the audience. Resources required include understanding the problem's current state and its importance.
- **Definition of the objectives for a solution** - The solution's objectives are derived from the problem definition and an understanding of what is possible and feasible. These objectives can be either quantitative, specifying improvements over current solutions, or qualitative, describing how a new artifact is expected to address previously unexplored problems. The rational inference of objectives from the problem specification is essential.
- **Design and development** - An artifact is created, representing any designed object wherein a research contribution is included in the design. This involves outlining the artifact's desired functionality and architecture, followed by the actual creation of the artifact.
- **Demonstration** - This activity demonstrates using the artifact to solve instances of the problem, such as through experimentation, simulation, case studies, or proofs.
- **Evaluation** - The evaluation assesses how effectively the artifact supports a solution to the problem. It entails comparing solution objectives to observed results from using the artifact. Evaluation formats vary based on the problem venue and the artifact. Researchers can decide whether to iterate for improvement or proceed to communication, leaving further enhancements for subsequent projects.
- **Communication** - All aspects of the problem and the designed artifact are communicated to relevant stakeholders. The choice of communication methods depends on research goals and the audience, which may include practicing professionals.

The steps given were adapted from [Brocke et al., 2020].

This process will be employed in the methodology of this dissertation. Subsection 6.2.3 provides detailed integration.

6.2.3 Applying DSR in the dissertation

From the initial division of objectives in the introduction of this work, there is already a preliminary alignment with the steps outlined in this methodology. How-

ever, a more detailed integration can be specified through a comprehensive list of steps, similar to the one provided in the previous subsection.

- **Problem identification and motivation** - The problem defined for this dissertation is the usage of security by opacity against active attacks in DES. DES can be used to describe many systems that are used regularly on a daily-basis, and active attacks are the most predominant nowadays. The application of opacity in this context aims to bridge a literature gap, given the current lack of investigation into its usage in this specific scenario.
- **Definition of the objectives for a solution** - The primary goal of the dissertation is to formulate the proposed technique, aiming for a best-case scenario where it has a quantifiable impact on current security techniques. Additionally, the exploration will consistently emphasize the qualitative aspect, given the limited literature on this specific problem.
- **Design and development** - The development of the proposed-technique to protect a system from a specific active attack, using opacity.
- **Demonstration** - This topic is addressed through the usage of the Hydra System to create a case study, where the proposed technique will be employed and tested.
- **Evaluation** - Upon applying the proposed technique in the case study, conclusions regarding its utility and effectiveness will be drawn. This entails assessing whether the technique can effectively safeguard a representation of a real system. Some limitations may arise due to the limited amount of literature on the topic, potentially making it more challenging to address difficulties when applying the technique to the system.
- **Communication** - Given that this is an academic dissertation, that is being done with coordinators from the University of Coimbra and University of RomaTre, bi-weekly meetings are held to discuss progress and address defined objectives. These meetings involve collaborative discussions among all parties to make informed choices and ensure comprehensive development.

Chapter 7

Conclusion

In conclusion, this study has successfully introduced and applied a novel security technique using opacity against active attackers within a DFA. This technique, distinct from previous approaches, addresses the challenge of maintaining system confidentiality and integrity in the face of Actuator-Enablement attacks, a type of active threat. By adapting theoretical foundations into practical implementations, Algorithms 4 and 5 demonstrate how system administrators can effectively manage control decisions against both passive and active attacks, using the AIC-O framework to strategically map out responses.

The efficacy of the developed technique was validated through practical experiments within the Hydra system, showcasing its ability to enhance defendability against various attack scenarios. Notably, the study highlights the importance of defining defendable events within the AIC-O framework, particularly in scenarios involving active attackers. While limitations were identified, such as the reliance on predefined sets of events, these findings underscore areas for future research to explore more adaptive techniques capable of dynamically responding to emerging threats.

Furthermore, this dissertation underscores the need for refining assumptions to enhance system defendability. While the developed technique forms a robust foundation, future advancements should focus on reducing constraints on system evolution and establishing clearer criteria for event defendability. These refinements are crucial for advancing intrusion detection systems and ensuring system integrity in dynamic cybersecurity scenarios.

In summary, this research bridges theoretical insights with practical applications to develop a pioneering security technique tailored for DES, specifically DFAs, using opacity to safeguard against both passive and active attacks. Moving forward, continued refinement and exploration of adaptive security measures will further contribute to the field, effectively addressing evolving cybersecurity challenges.

References

Badouel, E., Bednarczyk, M., Borzyszkowski, A., Caillaud, B., and Darondeau, P. (2007). Concurrent secrets. Discrete Event Dynamic Systems, 17:425–446.

Barcelos, R. J. and Basilio, J. C. (2023). Ensuring utility while enforcing current-state opacity. IFAC-PapersOnLine, 56(2):4595–4600.

Basilio, J., Hadjicostis, C., and Su, R. (2021). Analysis and Control for Resilience of Discrete Event Systems: Fault Diagnosis, Opacity and Cyber Security.

Battisti, F., Bernieri, G., Carli, M., Lopardo, M., and Pascucci, F. (2018). Detecting integrity attacks in iot-based cyber physical systems: a case study on hydra testbed. In 2018 Global Internet of Things Summit (GIoTS), pages 1–6.

Brocke, J. v., Hevner, A., and Maedche, A. (2020). Introduction to Design Science Research, pages 1–13.

Bryans, J. W., Koutny, M., Mazaré, L., and Ryan, P. Y. (2008). Opacity generalised to transition systems. International Journal of Information Security, 7(6):421 – 435.

Bryans, J. W., Koutny, M., and Ryan, P. Y. (2005). Modelling opacity using petri nets. Electronic Notes in Theoretical Computer Science, 121(SPEC. ISS.):101 – 115.

Carvalho, L. K., Wu, Y.-C., Kwong, R., and Lafortune, S. (2018). Detection and mitigation of classes of attacks in supervisory control systems. Automatica, 97:121–133.

Cassandras, C. and Lafortune, S. (2010). Introduction to Discrete Event Systems, page 800.

Falcone, Y. and Marchand, H. (2015). Enforcement and validation (at runtime) of various notions of opacity. Discrete Event Dynamic Systems, 25:531–570.

Fritz, R. and Zhang, P. (2023). Detection and localization of stealthy cyber attacks in cyber-physical discrete event systems. IEEE Transactions on Automatic Control.

Hélouët, L., Marchand, H., and Ricker, L. (2018). Opacity with powerful attackers. IFAC-PapersOnLine, 51(7):464–471.

Lafortune, S., Lin, F., and Hadjicostis, C. N. (2018). On the history of diagnosability and opacity in discrete event systems. Annual Reviews in Control, 45:257–266.

Mazaré, L. (2004). Using unification for opacity properties. Proceedings of the 4th IFIP WG1, 7:165–176.

Meira-Góes, R., Wintenberg, A., Matsui, S., and Lafortune, S. (2023). Mdesops: An open-source software tool for discrete event systems modeled by automata. IFAC-PapersOnLine, 56(2):6093–6098.

Oliveira, S., Leal, A. B., Teixeira, M., and Lopes, Y. K. (2023). A classification of cybersecurity strategies in the context of discrete event systems. Annual Reviews in Control, 56:100907.

Partovi, A., Jung, T., and Hai, L. (2020). Opacity of discrete event systems with active intruder.

Peffers, K., Tuunanen, T., Rothenberger, M., and Chatterjee, S. (2007). A design science research methodology for information systems research. Journal of Management Information Systems, 24:45–77.

Ramadge, P. and Wonham, W. (1989). The control of discrete event systems. Proceedings of the IEEE, 77(1):81–98.

Rosa, L., Alves, P., Cruz, T., Simões, P., and Monteiro, E. (2015). A comparative study of correlation engines for security event management. In Iccws 2015-The Proceedings of the 10th International Conference on Cyber Warfare and Security, page 277.

Rosa, L., Cruz, T., de Freitas, M. B., Quitério, P., Henriques, J., Caldeira, F., Monteiro, E., and Simões, P. (2021). Intrusion and anomaly detection for the next-generation of industrial automation and control systems. Future Generation Computer Systems, 119:50–67.

Saboori, A. and Hadjicostis, C. N. (2011a). Verification of k-step opacity and analysis of its complexity. IEEE Transactions on Automation Science and Engineering, 8(3):549–559.

Saboori, A. and Hadjicostis, C. N. (2011b). Verification of infinite-step opacity and complexity considerations. IEEE Transactions on Automatic Control, 57(5):1265–1269.

Saboori, A. and Hadjicostis, C. N. (2013). Verification of initial-state opacity in security applications of discrete event systems. Information Sciences, 246:115–132.

Tong, Y., Cai, K., and Giua, A. (2018). 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), pages 1053–1058.

Tong, Y., Ma, Z., Li, Z., Seatzu, C., and Giua, A. (2016). Supervisory enforcement of current-state opacity with uncomparable observations. pages 313–318.

Tong, Y., Wang, Y., and Giua, A. (2022). A polynomial approach to verifying the existence of a threatening sensor attacker. IEEE Control Systems Letters, 6:2930–2935.

Yao, J., Li, S., and Yin, X. (2024). Sensor deception attacks against security in supervisory control systems. Automatica, 159.

Yao, J., Yin, X., and Li, S. (2020). On attack mitigation in supervisory control systems: A tolerant control approach. In 2020 59th IEEE Conference on Decision and Control (CDC), pages 4504–4510.

Yin, X. (2015). A new approach for enforcing opacity via supervisory control for partially-observed discrete-event systems.

Zheng, S., Shu, S., and Lin, F. (2023). Modeling and control of discrete event systems under joint sensor-actuator cyber attacks. IEEE Transactions on Control of Network Systems.

Appendices

Appendix A

Planning

A.1 Planning for work after the Intermediate Delivery

The work plan for this dissertation involved several key steps:

Starting with a foundational phase, the initial focus is on acquiring comprehensive knowledge about DES and understanding their characteristics. This serves as a crucial building block, facilitating a more straightforward exploration of opacity.

Following this phase, the next step is a detailed literature review. The objective is to gather state-of-the-art information on cybersecurity in DES, with a specific emphasis on active attacks and the implementation of security through opacity.

Having a solid understanding of the fundamental concepts, the subsequent phase involves the development of a security technique. This technique aims to effectively defend DES against the identified types of attacks.

With the security technique in place, the dissertation progresses to a practical application through a case study. The Hydra System is chosen as the subject for this study. The application of the developed security technique to this real-world scenario allows for a thorough analysis of results and the drawing of conclusions regarding the system's security and resilience against the specified attack.

In essence, this approach ensures a logical and comprehensive progression, from basic knowledge acquisition to the practical application and analysis of a developed security technique within the context of the Hydra System.

A.1.1 Planning for the next phase

Considering that the next phase of work relies heavily on in-person collaboration, the work plan can be divided as follows:

- **During January to mid-February** - Focus on the intermediate delivery and presentation. After completing these, return to the literature review. Identify the type of cyber-attack that best aligns with the dissertation, focusing on active attacks against DES.
- **Mid-February to mid-March** - Around this period, in-person work will commence, facilitating a smoother flow of progress. Develop a more detailed understanding of the Hydra System, starting with a representation using a DES. Based on this representation and the previously studied attacks, create a simple representation of a specific attack on the system, initiating the development of a proposed technique for its defense.
- **Mid-March to mid-April** - During this period, fully develop the proposed technique so that the case study can begin execution. Depending on progress, ideally, start drafting conclusions regarding the usefulness and effectiveness of the technique. This phase also allows for considering and implementing possible upgrades or changes based on the ongoing progress and findings.
- **Mid-April to the end of May** - As this marks the final period for in-person work, there will be a priority in completing as much of the work that requires physical presence as possible.
- **June** - Dedicate time to work on the dissertation document, aiming to consolidate all achieved objectives. Ensure that the document is comprehensive and ready for presentation.

Throughout the specified months, it is understood that documentation for the dissertation document is being recorded. This approach ensures greater precision in capturing the occurrence of events as they happen, rather than relying on retrospective documentation at the end.

In Appendix B, there is a Gantt Chart that provides a condensed division of work for the tasks that were completed, along with their duration.

Appendix B

Gantt Chart with a Condensed Division of Work

Dissertation Gantt
Read-only view, generated on 26 Jun 2024
2023
Oct 2023
Nov 2023
Dec 2023
Jan 2024
Feb 2024
Mar 2024
Apr 2024
May 2024
Jun 2024
Jul 2024
18 25 02 09 16 23 30 06 13 20 27 04 11 18 25 01 08 15 22 29 05 12 19 26 04 11 18 25 01 08 15 22 29 06 13 20 27 03 10 17 24 01 08 15 22 29
Masters Thesis - João Melo
Write/Update the Thesis Document
Background and introduction to DES theory
Literature Review
Adapt the HYDRA system
Combine the techniques
Define the Proposed Technique
Define the simpler Hydra system
Apply the Technique Theoretical
Update the Technique - Change Assumptions and Algorithm
Apply Updated technique to the Simpler Hydra System