1-1-2013

# Opacity Of Discrete Event Systems: Analysis And Control

Majed Mohamed Ben Kalefa
*Wayne State University,*

# OPACITY OF DISCRETE EVENT SYSTEMS: ANALYSIS AND CONTROL

by

**MAJED MOHAMED BEN KALEFA**

**DISSERTATION**

Submitted to the Graduate School

of Wayne State University,

Detroit,Michigan

in partial fullfilment of the requirements

for the degree of

**DOCTOR OF PHILOSOPHY**

2013

MAJOR: Electrical Engineering

Approved by:

_____

Advisor Date

_____

_____

_____

_____

## *DEDICATION*

*To my family.*

## ACKNOWLEDGEMENTS

I am grateful for guidance and support of Prof. Feng Lin, without which this research effort would never have matured to its present state.

I would like to thank the committee members with their patience in reading the dissertation and also being cautiously supportive of my desire to produce new knowledge.

My gratitude also goes to the staff and faculty of Wayne State University for great facilities, friendships, and research opportunities.

I am grateful to my brothers and sisters (Kheria, Haji, Tariq, Raheel, Zaema, Ismail, Asem, Hanan, and Faiq) who have had a positive influence not only on my upbringing but also in my maturation.

I am grateful to my good wife Abeer, and my children, Salem, Saline and Cizar. I feel that I receive the recognition of this work, but they have paid the price.

# *TABLE OF CONTENTS*

# LIST OF FIGURES

# 1 Introduction

## 1.1 Motivation

Security and privacy of computer systems and communication protocols have become the most challenging objectives of modern hardware and software design. In many applications like healthcare systems, power distribution systems and e-voting systems the leak of the information flow can be very dangerous. In the past few years, various notions concentrate on characterizing the information flow from the system to an external observer. Opacity is one of the notions defined to study some properties of security and privacy. It has been shown that for instance, properties like anonymity and secrecy can be studied as special cases of opacity. Opacity describes the inability for an external observer to know what happened in a system; therefore, it aims at determining whether a given critical subset of a system's behavior is kept opaque to an external observer. In the context of a discrete event system with partial observation, for example, anonymity is defined as the ability to hide a set of particular actions among other actions. Through that, anonymity can be investigated in terms of opacity in the framework of a discrete event system where the occurrence of events is strongly opaque if an observer cannot determine which events have occurred for all executions of the system. Consider secrecy, a secret is defined as a particular subset of trajectories and the observer should not find out that a trajectory of the system belongs to this subset. In this context, a secret is said to be strongly opaque with respect to the system if every execution of the secret is confused with other execution from the observer's point of view. Consider the case of the e-voting system, which is physically supervised by independent electoral authorities. A fundamental challenge with any voting mechanism is assuring that is not possible for an external observer to discover the vote of an elector. In the context of opacity, this can be defined as it is ensuring strong opacity property. Also, detecting errors and fraud is important in the e-voting mechanism, therefore ensuring that every elector votes without fraud is also a concern of opacity properties. The interest in

verifying security properties is growing, therefore analyzing opacity properties in discrete event systems become a significant tool to serve the developers for ensuring security and privacy properties.

Motivated by enforcing opacity properties in many application areas, we investigate supervisory control for opacity. For a given system that does not satisfy opacity properties, we may need to investigate if a supervisor can be used to restrict the system's behavior so that the supervised system satisfies opacity properties. The supervisor can disable some controllable events to restrict the behavior of the system. Therefore, supervisor control problems for opacity involve controllability, observability (or normality). We formally define the opacity control problem (SOCP), the weak opacity control problem (WOCP), and no opacity control problem (NOCP). Solutions to SOCP in terms of the largest sublanguage that is controllable, observable (or normal), and strongly opaque are characterized. Similar characterization is available for solutions to NOCP.

## 1.2 Related Work and Contributions

The developments in this dissertation are related to existing work in opacity and its application [33]. Opacity introduced by [33] is presented through the following definitions: strong opacity and weak opacity. Given a general observation mapping, a language is strongly opaque if all strings in the language are confused with some strings in another language, and it is weakly opaque if some strings in the language are confused with some strings in another language. A language is not opaque if it is not weakly opaque. It has been showed that anonymity [44, 46, 55] and secrecy [8, 23] can all be investigated in terms of opacity [33]. Furthermore, it has been shown that three important properties of discrete event system, observability, diagnosability, and detectability, can all be reformulated as special cases of opacity [33].

Opacity requires that the system's secret behavior is opaque to an external observer who is able to observe some events that occur in the system. Therefore opacity can be a useful property to prove the security of protocol. The notion of similarity [18, 37, 38] was

used to prove the property of the security of protocol. In other words, an external observer is not able to distinguish a run of protocol where the property is satisfied from a run where it is not.

The notion of opacity in computer security and other information flow security has been extended in [23] to discrete event systems. In [23] three variants of the opacity have been introduced: the initial opacity, the final opacity and total opacity. The initial opacity focuses on the initial information that needs to be kept secret. On the other hand, the final opacity defines the situation where the final state needs to be kept secret. The total opacity is more general because it considers that both the initial state and the final state need to be kept secret. This approach showed how the opacity is related to other concepts used for information flows security like anonymity and non-interference.

In [24, 25] the opacity property was investigated in the framework of Petri nets. Based on the observation of the locale states of the system as well as the execution of the traces, opacity was defined as property of certain states of the secure. For a given an observation mapping and a secure property (states), opacity is defined as the interest of finding out if the outside observer can determine the secure states.

One of the earliest approaches of the opacity is presented by [50, 51]. In this framework opacity was defined based on the following assumption. The states of the system are partitioned to a secret and non-secret states. This approach is known as the *state-based* approach. In this approach opacity requires that secret states need to be kept secret from an observer who observes events under the natural projection. That is, for any string reaches the secret state, there must exist at least one other different string that has the same projection and reaches another state that is not secret.

The *language-based* approach presented in [8] investigates the system that its secret is defined by language generated. Opacity requires that none of multiple observers with different observation capabilities are able to determine whether the actual behavior of the system belongs to a secret language. Also, it has shown opacity can be enforced by supervisory control possibly by disabling the least possible of strings to restrict the ability of the

observer to find out that the actual language belongs to the secret language.

In the last few years, some opacity studies focussed on demonstrating opacity properties enforced by supervisory control. Opacity enforced by a controller has been studied in [9, 19–22] where opacity is defined by keeping a secret (language) opaque with respect to the language generated by the system. Disabling some controllable events can lead to the restriction of the language generated. The observation mapping is reduced to the natural projection in these studies.

In this dissertation, we will use the definition of opacity proposed in [33]. The reason for this is that the system behaviour can be analyzed and studied from the occurrences of events. To investigate opacity properties we need to analyze the observation mapping by observing more or less events or controlling a subset of events by enabling or disabling events. Also, the two definitions of opacity give us more flexibility to satisfy opacity if needed. If strong opacity cannot be satisfied, then we modify the language to satisfy weak opacity.

The research goals of this dissertation is to study, and to investigate opacity in the discrete event system. This work considers theoretical analysis of opacity and its applications.

The first contribution of the dissertation is to study opacity properties in discrete event systems. The second contribution of the dissertation is to evaluate opacity theories by considering opacity under union, and opacity under intersection to establish a structural information for computing sublanguages and superlanguages. The third contribution is to provide formulas for sublanguages and superlanguages. We find the largest sublanguages and smallest superlanguages that satisfies the properties of opacity. We provide an extended version of the computed sublanguages and superlanguages in the decentralized framework of discrete event system, in order to provide a foundation for the case of many observers are connected to the system. Finally, the dissertation contributes to the construction of a minimally restrictive supervisor for opacity. We consider opacity under the controllability and observability restrictions. We show how a supervisor limits the system's behavior within a specified behavior that satisfies controllability, observability and opacity requirements. We characterize solutions to the strong opacity control problem in terms of the largest

sublanguage that is controllable, observable (or normal), and strongly opaque. We show that the supremal controllable and normal sublanguage exists. Similar characterization is available for solutions to no opacity control problem. We also show that there is no solution to the weak opacity control problem. Generally, the result is a supervisor that achieves the desired behavior to enforce opacity possibly by disabling, a subset of controllable events, in a way that system behavior is minimally restricted.

## 1.3 Methodology

Discrete Event System is a discrete-state, event-driven system of which the state evolution depends entirely on the occurrence of asynchronous discrete events, where an event is to be understood as an action without temporal duration. Examples of discrete event systems include process engineering, communication systems, networks, digital circuits, traffic networks, and manufacturing systems. Generally the dynamics of a discrete event system can by be described on the basis of its observed behavior. Internally the dynamics described by the set of all possible sequences of events, called traces, that the system can execute starting from its initial state.

In general, from the viewpoint of the properties and the specification of discrete event systems, it is more convenient to be represented and analyzed in the formal language setting. Therefore, this dissertation is mainly developed on the basis of the formal language. However, some examples and proofs are illustrated on the basis of the automata.

For investigating and studying opacity in the framework of discrete event systems we investigate first the properties of opacity based on the definitions given by [33]. By studying strong opacity, weak opacity, and no opacity we formulate languages need to be enlarged or reduced to satisfy opacity properties. In other words, the information needs to be observed or not observed to satisfy opacity properties. Thus, we will be considering the sublanguages that represent the desired behavior to satisfy strong opacity, weak opacity and no opacity. Also we provide formulas for computing sublanguages and superlanguages.

As one of the applications that can be studied and investigated using opacity is

anonymity properties. We consider anonymity protocol as an application for this dissertation. In general anonymity is a notion that arises in activities where the identity of users involved need to be kept secret. We investigate anonymity protocols using opacity properties in the framework of discrete event systems. We give initially an analysis of the verification of protocol using strong opacity and no opacity properties, and then we synthesize anonymity protocol using opacity properties.

In order to modify the behavior of a system to satisfy opacity some events should be controlled, or some events should be observed, therefore we investigate opacity theory related to controllability and observability theories. We compute the superlanguages and sublanguages under controllability, observability and opacity restrictions.

## 1.4 Thesis Outline and Publications

The dissertation is divided to 7 chapters including an introduction in Chapter 1 . A brief outline of the topics discussed in these chapters are given as follows. Chapter 2 focuses on the introduction of the automata and a formal language model for discrete event systems proposed by Ramadge, Wonham and Lin in [45] [31] and the presentation of its suitability for dealing with control-theoretic issues. Notions of supervisory control, controllability, observability and normality are defined and discussed. In Chapter 3, the definition of the opacity in the framework of discrete event system is presented. The main theorem of Chapter 3, establishes the properties of opacity. Chapter 4 presents the formulas for calculating the sublanguages and superlanguages. In Chapter 5, we extend opacity of the discrete event system to decentralized systems. We provide computation of the sublanguages and superlanguages in the framework of decentralized discrete event systems . Chapter 6 provides an introduction to anonymity and the dinning cryptographers problem. The analysis and the synthesis of the dinning cryptographers protocol are presented. Chapter 7 contains the results of the supervisory control of opacity. We present a solution to strong, weak, and no opacity problems.

Many of the results in the dissertation have been published in a journal and in the

proceedings of conferences. More specifically, the results of Chapter 4 appeared in [3] and an extended version was published in [5]. The results in Chapter 7 appeared in [4].

# 2 Basic Notions and Definitions of Discrete Event Systems

Discrete event systems are systems which are discrete in both time and state space. The changes in the system states occur based on event transitions. Each event occurs marks a change of state in the system. Discrete event systems provide useful models for complex control systems, such manufacturing systems, networks, computer databases, communication protocols, etc.

Discrete event system theory started with supervisory control theory over twenty years ago. The must important concepts in supervisory control are controllability introduced by Ramadge and Wonham [45] and observability introduced by Lin and Wonham [31]. The notion of controllability introduced by Ramadge and Wonham is a necessary and sufficient condition for the existence of a supervisor that achieves the desired controlled behavior of a given discrete event system under the complete observation of events. However, if the system is under partial observation of events, then an additional condition of observability introduced by Lin and Wonham should be considered. This condition is necessary and sufficient for the existence of such a partial observation supervisor. Controllability theory and observability theory were studied extensively by many other researchers. Several important problems in discrete event systems such as supervisory control and observation problem were solved because of these studies.

Other properties of discrete event systems, such as diagnosability [29, 34–36, 58, 59] and detectability [48, 49] have also been studied. The notion of diagnosability is a necessary and sufficient condition for the existence of a diagnoser that can perform failure diagnostics. These conditions can be verified using detection algorithms on the diagnosers and the system.

The detectability problem in a discrete event system has been investigated in [48, 49]. Detectability aims to estimate or to determine the current state of the system based on the observation mapping. It has presented computable criterions for checking necessary and

sufficient conditions for detectability. It has also presented the way to construct an observer, whose roles are to estimate the states of a system after a sequence of observation is very important in some applications. It is shown in [48] that in some medical applications, determining the state of the system is an important task. Detectability and periodic detectability, both in a strong sense and in a weak sense has been defined. A discrete event system is strongly detectable if the current state and subsequent states of the system after a finite number of observations for all trajectories of the system can be determined. A discrete event system is weakly detectable if the current state and the subsequent states of the system after a finite number of event observations for some trajectories of the system can be determined.

The general formalism to describe discrete event systems is the formal languages [17,56]. The language is a theoretic-based representation of the systems whose behaviour can be described by sequences of events. These sequences of events are called strings.

Discrete event systems are often represented by state machine (automata), where transitions between states are labeled by events. Also a set of regular languages is used to represent supervisory control for discrete-event systems. Regular language operators such as choice, concatenation, and Kleene-closure have been defined in the setting of languages to allow modeling of complex systems in terms of simpler ones. In this work we present the theoretical analysis in the regular language framework. We present also some of the theoretical results in the automata framework.

As we mentioned earlier, this chapter focuses on the representation of the discrete event system. Therefore, the organization of the chapter is as follow. Formal language definitions and regular expressions are presented in Sections 2.1. The Section 2.2 introduces finite automata which will be relevant in the dissertation. Supervisory control of discrete event systems is presented in Section 2.3.

## 2.1 Languages

In computer science, the formal language is widely used to define the data formats and the syntax of programming languages. Therefore, they are playing an important role in the development of compilers. In addition, formal languages are also used in logic and in mathematics to represent the syntax of formal theories and algorithms. The behavior of discrete event systems is studied in this dissertation by formal languages. In this section we give a comprehensive introduction to formal languages and automata presented in [17, 56].

### 2.1.1 Basic Definitions and Notation

Let $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_m\}, m \in \mathbb{N}^+$ be a finite nonempty set of distinct symbols. $\Sigma$ used to denote the *alphabet*.

**Definition 2.1.1.** *[52] A word (or string) over an alphabet $\Sigma$ is a finite sequence of elements from an* alphabet $\Sigma$ *. A string $s$ can be written as the arbitrary concatenations $s = \sigma_{i_1}\sigma_{i_2}...\sigma_{i_k}$ of symbols $\sigma_{i_1}, \sigma_{i_2}, ..., \sigma_{i_k} \in \Sigma$ with $i_1, ..., i_k \in \{1, ..., m\}$, where $k \geqslant 1$ is the length of the string $s$.*

The set of all finite strings over the alphabet $\Sigma$ is written as $\Sigma^*$. The * operation is called the *Kleene-closure*. The empty string (string with no symbols) is $\epsilon$, where $\epsilon \notin \Sigma$.
Definition 2.1.1 basically presents the concept of language over an*alphabet* $\Sigma$.

**Definition 2.1.2.** *A* language *over an alphabet $\Sigma$ is a subset of $\Sigma^*$. [56]*
*This definition includes the empty set $\emptyset$ which is a language that has no strings and the* kleene-closure $\Sigma^*$.

As strings are the elements of a language. Therefore, the usual operations on sets like union, intersection, difference and complement with respect to $\Sigma^*$, can be applied to languages. In addition to those already presented operations, there are other important operations. *Kleene-closure*, concatenation, and the *prefix-closure* are the fundamental operations in generating of strings. The concatenation is a binary operation on $\Sigma^*$, where the

concatenation of $\alpha$ and $\beta$, is written $\alpha\beta$. A string $s \in \Sigma^*$ is a proper prefix of $t \in \Sigma^*$ if $s$ is an initial substring of t. The set of prefixes of elements of a language $L \subseteq \Sigma^*$ is denoted by $\bar{L}$. This language is called the prefix-closure of $L$, and $L$ is said to be prefix-closed if $L = \bar{L}$. The Kleene closure of a set $L$ denoted by $(L^*)$ is the set of all strings of finite length, whose elements are elements of set L.

The technique for constructing languages is to use set operations to construct complex sets of strings from simple ones. Example 2.1.1 shows Operations on formal languages.

**Example 2.1.1.** *Let $\Sigma = \{\alpha, \beta, \gamma\}$ be an alphabet and consider the languages $L = \{\alpha\beta + \alpha\gamma\}$, $K = \{\beta\gamma\beta\}$. $s = \beta$ and $t = \gamma\beta$ are strings over $\Sigma$. The concatenation of $s$ and $t$ is given by $st = \beta\gamma\beta$. The prefix-closure of $K$ is $\bar{K} = \{\epsilon, \beta, \beta\gamma, \beta\gamma\beta\}$. The kleene-closure of $\Sigma$ is $\Sigma^* = \{\epsilon, \alpha, \beta, \gamma, \alpha\alpha, \alpha\beta, \alpha\gamma, \beta\alpha, \beta\beta, \beta\gamma, ....\}$.*

In expressions involving several operations, one should apply the operations with the following order: (1) Closure, (2) Concatenation, and (3)Union, intersection, and set the difference. Also regarding the empty string $\epsilon$ we present the following observations.

- $\epsilon \notin \{\emptyset\}$
- $\{\epsilon\}$ is a nonempty language containing only the empty string
- If $L = \emptyset$, then $\bar{L} = \emptyset$, and if $L \neq \emptyset$ then $\epsilon \in \bar{L}$
- $\emptyset^* = \{\epsilon\}$ and $\{\epsilon\}^* = \{\epsilon\}$

In [30,31] the natural projection is introduced. We recall the definition and its inverse image function.

**Definition 2.1.3.** ***(Projection)*** *[6]*
*Let $\Sigma_s \subseteq \Sigma$. The projection $P : \Sigma^* \longrightarrow \Sigma_s^*$ is*

$$P(s\sigma) = \begin{cases} P(s)\sigma & \text{if } \sigma \in \Sigma_s, \\ P(s) & \text{Else} \end{cases}$$

*for $s \in \Sigma^*$ and $\sigma \in \Sigma$. The map $P$ is called the natural projection of $\Sigma^*$ onto a* smaller *set of events $\Sigma_s^*$. Clearly for $s,t \in \Sigma^*$ we have $P(st) = P(s)P(t)$ and $P(\epsilon) = \epsilon$,i.e. $P$ is concatenating.*

The natural projection P deletes events from the string that belong to $\Sigma$ and do not belong to $\Sigma_s$. The corresponding inverse natural projection $P^{-1} : \Sigma_s^* \longrightarrow 2^{\Sigma^*}$ of a given string $t \in \Sigma_s^*$ returns the set of strings that are projected on $t$.

**Definition 2.1.4.** ***(Inverse Projection)*** *[6]*
*Let $\Sigma_s \subseteq \Sigma$. The inverse projection $P^{-1} : \Sigma_s^* \longrightarrow 2^{\Sigma^*}$ is*

$$P^{-1}(t) := \{s \in \Sigma^* | P(s) = t\}$$

*for $t \in \Sigma_s^*$*

The projection P and the inverse projection $P^{-1}$ are generalized to languages by simply applying them to all the strings in the language [34]. For $L \in \Sigma^*$ and $L_s \in \Sigma_s^*$:

$$\begin{aligned} P(L) &:= \{t \in \Sigma_s^* : (\exists s \in L)[P(s) = t]\} \\ P^{-1}(L_s) &:= \{s \in \Sigma^* : (\exists t \in L_s)[P(s) = t]\} \end{aligned}$$

The next example shows how to apply the natural projection and its inverse to languages.

**Example 2.1.2.** *Let $\Sigma = \{\alpha, \beta, \gamma\}$ and consider the languages $L_1 = \{\alpha\beta, \alpha\beta\alpha\gamma\}$, $L_2 = \{\alpha\gamma\}$. The alphabet $\Sigma_s = \{\alpha, \beta\}$ is considered to be the alphabet of the natural projection $P : \Sigma^* \to \Sigma_s^*$. The natural projection of the languages are $P(L_1) = \{\alpha\beta, \alpha\beta\alpha\}$ and $P(L_2) = \{\alpha\}$ respectively. The inverse projection of $L_1$ and $L_2$ is given by $P^{-1}P(L_1) = \{\gamma^*\alpha\gamma^*\beta\gamma^*, \gamma^*\alpha\gamma^*\beta\gamma^*\alpha\gamma^*\}$ and $P^{-1}P(L_2) = \{\gamma^*\alpha\gamma^*\}$. The concept of the natural projection is illustrated in Figures 2.1 and 2.2. In Figure 2.1 the lines present strings, and the crosses present the concatenated symbols. The way to read a string is from the left to the right. The projected string $P(L_1)$ of the language $L_1$ and its strings in $L_1$ are linked by dashed lines. Figure 2.2 shows the concept of the natural projection and its inverse applied to the language. The projection $P(L_1)$ of*

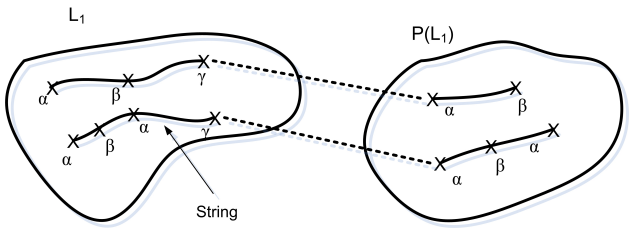



*Figure 2.1: The illustrated language $L_1$ and its projection $P(L_1)$*

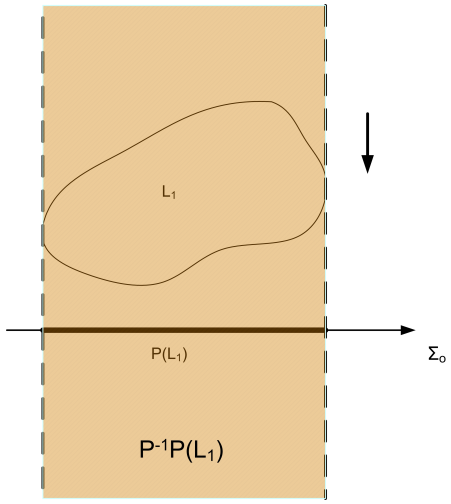



*Figure 2.2: Illustration of the $P(L_1)$ and the inverse $P^{-1}P(L_1)$*

*the language $L_1$ is presented by a bold line over the axis. The colored (shadow) area shows the inverse projection $P^{-1}P(L_1)$ of $L_1$.*

Another operation frequently performed on strings and languages which can be presented using the inverse projection, is called *Parallel Composition* or synchronous product.

**Definition 2.1.5.** ***Parallel Composition [34]***
*Let $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$, and $\Sigma = \Sigma_1 \cup \Sigma_2$. The natural projection $P_i$ is given by $P_i : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^*$ for $i = 1, 2$, the parallel composition of $L_1$ and $L_2$ is:*

$$L_1 \parallel L_2 := P_1^{-1}(L_1) \cap P_2^{-2}(L_2)$$

### 2.1.2 Regular Language

Within the formal languages, the languages are classified to four formal languages [17, 34]. This classification is known the Chomsky hierarchy. One type of languages called the regular language, which is the simplest of four Chomsky formal languages. The regular expression is one of the ways to describe regular languages.

**Definition 2.1.6.** ***(Regular Language)*** *[41]*
*The language is said to be regular over $\Sigma$ if it belongs to the set of regular languages defined recursively as follow:*

- ***Basis Clause:*** *$\emptyset, \epsilon$ and $\alpha$ for any symbol $\alpha \in \Sigma$ are regular languages*
- ***Inductive Clause:*** *If $L_1$ and $L_2$ are regular languages, then $L_1 \cup L_2, L_1L_2$ and $L_1^*$ are regular languages.*
- ***Extremal Clause:*** *Nothing is a regular language unless it is obtained from the above two clauses.*

**Example 2.1.3.** *Let $\Sigma = \{\alpha, \beta\}$. Then since $L_1 = \{\alpha\}$ and $L_2 = \{\beta\}$ are regular languages, $L_1 \cup L_2 = (\{\alpha\} \cup \{\beta\})$ and $L_1L_2 = \{\alpha\}\{\beta\}$ are regular languages. Also since $\{\alpha\}$ is regular, $\{\alpha\}^*$ is a regular language.*

### 2.1.3 Regular Expression

Regular expressions are used to represent regular languages. They are also used substantially to represent the defined operations as concatenation, Kleene-closure, and the union on languages.

**Definition 2.1.7.** ***(Regular expressions)*** *[41]*
*The regular expressions over $\Sigma$ are defined recursively as below:*

- ***Basis Clause:*** *$\emptyset, \epsilon$ and $\alpha$ for any symbol $\alpha \in \Sigma$ are regular expressions.*

- ***Inductive Clause:*** *If $s_1$ and $s_2$ are regular expressions corresponding to languages $L_1$ and $L_2$, then $(s_1+s_2)$, $(s_1s_2)$ and $s_1^*$ are regular expressions corresponding to languages $L_1 \cup L_2, L_1L_2$ and $L_1^*$ respectively.*

- ***Extremal Clause:*** *Nothing is a regular expression unless it is obtained from the above two clauses.*

### 2.1.4 Conventions On Regular Expressions [34, 52]

Generally, an infinite number of strings for potentially complex languages can be easily represented by regular expressions. Regular expressions is a compact way of describing regular languages with a possibly infinite number of strings. Regular expressions represent the following defined operations.

- The concatenation of $r$ and $s$, $rs$ is written as $rs$.

- The union of $r$ and $s$, $r \cup s$ is written as $r+s$.

- The regular expression $(r+(s(t^*)))$ is written as $r+st^*$

**Example 2.1.4.** *Let $\Sigma = \{\alpha, \beta, \gamma\}$ be a set of events. The regular expression $(\alpha\beta+\gamma)\alpha^*$ denotes the language*

$$L = \{\alpha\beta, \gamma, \alpha\beta\alpha, \gamma\alpha, , \alpha\beta\alpha\alpha, \gamma\alpha\alpha, ...\}$$

## 2.2 Finite Automata

In this section we present a class of machines called finite automata. Finite automata is used to describe regular languages. As we show that the regular languages can be presented by regular expressions. Another kind of representing regular languages is the finite automata. We present the equivalencies and conversions between finite automata and expression languages. It is shown in [6] that for every regular language, a finite automaton can be constructed which can recognize the language. We formally present both types of finite automata, the deterministic as well as in nondeterministic finite automata.

**Definition 2.2.1.** ***(Deterministic automata)*** *[6]*

*A deterministic automaton, denoted by $G$, is a six-tuble*

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

*Where:*

1. *$X$ is the set of states.*
2. *$\Sigma$ is the finite set of events.*
3. *$f : X \times \Sigma \to X$ is the transition function: $f(x, \sigma) = y$ means that there is a transition labeled by event $\sigma$ from state $x$ to state $y$. It is only defined for a subset of $\Sigma$ in any state.*
4. *$x \in X$. $\Gamma : X \to 2^{\Sigma}$ is the active event function.*
5. *$\Gamma(x)$ is the set of events $\sigma$ for which $f(x, \sigma)$ is defined.*
6. *$x_0$ is the initial state.*
7. *$X_m \subseteq X$ is the set of marked states.*

To show the concept of the finite automata we use the following example.

**Example 2.2.1.** *Let $G = (X, \Sigma, f, x_0, X_m)$ be a deterministic finite state automaton. Figure 2.3 illustrates the automaton. States are presented by the nodes and events between the states represented by the arrows. The alphabet of $G$ is $\Sigma = \{\alpha, \beta, \gamma, \mu\}$, the states set is $X = \{0, 1, 2\}$, $x_0 = 0$ is the initial state and $X_m = \{0\}$ is the marked state. The transition function $f$ states that $f(0, \alpha) = 1$ and $f(1, \beta) = 0$ for example.*

The function $f$ can be extended to the domain $X \times \Sigma^*$, defining the function $f$ recursively as follows:

- $f(x, \epsilon) = x$ for each $x \in X$.

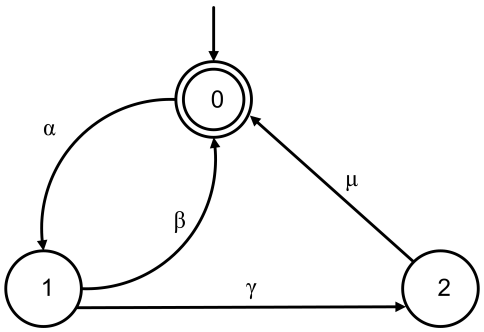



*Figure 2.3: Automaton Graph of G*

- $f(x, s\sigma) = f(f(x, s), \sigma)$ for each $s \in \Sigma^*$ and $\sigma \in \Sigma$.

Returning to the automaton in Example 2.2.1, we have, for example, that:

$$\begin{aligned} f(0, \epsilon) &= 0 \\ f(0, \alpha\gamma\mu) &= f(f(0, \alpha\gamma), \mu) = f(f(f(0, \alpha), \gamma), \mu) = f(f(1, \gamma), \mu) = f(2, \mu) = 0 \end{aligned}$$

### 2.2.1 Languages Generated by Automata

A generative model of a language is used either to recognize or to generate strings. The full set of strings that can be generated is called the language of the automaton. In other words, we consider all strings which can be followed in an automaton.

**Definition 2.2.2.** ***(Language generated and language marked)*** *[34]*
*Let $G = (X, \Sigma, f, x_0, X_m)$ be an automaton. The language generated by $G$ is*

$$\mathcal{L}(G) := \{s \in \Sigma^* | f(x_0, s)\}$$

*The language marked by $G$ is*

$$\mathcal{L}_m(G) := \{s \in \Sigma^* | f(x_0, s) \in X_m\}$$

The generated language $\mathcal{L}(G)$ contains all strings in the automaton starting from the initial state. The language marked includes all strings starting from the initial state to the marked states. The language marked is also called the language recognized by the automaton,

therefore a given automaton $G$ generates the language $\mathcal{L}(G)$ and recognizers the language $\mathcal{L}_m(G)$.

**Example 2.2.2.** *Consider the automaton $G$ in Figure 2.3, the language marked $\mathcal{L}_m(G) = \{\alpha\beta, \alpha\gamma\mu, \alpha\beta\alpha\gamma\mu, \alpha\beta\alpha\beta\alpha\gamma\mu, \alpha\beta\alpha\beta\alpha\beta\alpha\gamma\mu, ...\}$ and the $\mathcal{L}(G) = \{\epsilon, \alpha, \alpha\gamma, \alpha\gamma\mu, \alpha\beta, \alpha\beta\alpha, \alpha\beta\alpha\gamma, ...\}$.*

Obviously, there are two languages $\mathcal{L}(G)$ and $\mathcal{L}_m(G)$ can be represented by an automaton $G$. On the other hand there exist many ways to construct an automata that generate, or mark, a given language.

**Definition 2.2.3.** ***(Language-equivalent automata)*** *[34]*
*Automata $G_1$ and $G_2$ are said to be language-equivalent if*

$$\mathcal{L}(G_1) = \mathcal{L}(G_2) \text{ and } \mathcal{L}_m(G_1) = \mathcal{L}_m(G_2)$$

*That is, if they both recognize the same language generated $\mathcal{L}(G)$ and marked language $\mathcal{L}_m(G)$.*

**Example 2.2.3.** *Let $G_1 = (X_1, \Sigma_1, f_1, x_{0,1}, X_{m,1})$ and $G_2 = (X_2, \Sigma_2, f_2, x_{0,2}, X_{m,2})$ shown in Fig. 2.4 be deterministic automaton. $G_1$ and $G_2$ are language-equivalent, as they generate the same language and they mark the same language.*

**Definition 2.2.4.** ***(Nondeterministic automata)***
*A nondeterministic finite automaton is a 5-tuple $G_{nd} = (X, \Sigma \cup \epsilon, f_{nd}, x_0, X_m)$, where*

1. *$X$ is the set of states.*
2. *$x_0 \subseteq X$ is the set of initial states.*
3. *$f_{nd}$ is the transition function $f_{nd} : X \times \Sigma \cup \{\epsilon\} \to 2^X$.*
4. *$X_m \subseteq X$ is the set of marked states.*

*The important difference between the deterministic and nondeterministic automata is that $f(x, \sigma)$ is a (possible empty) set of states rather than a single state. That is, every state in a deterministic automata always has one $\sigma$-transition for every $\sigma \in \Sigma$, but in a nondeterministic automata, there could be states that have none, one, two or more $\sigma$-transitions.*

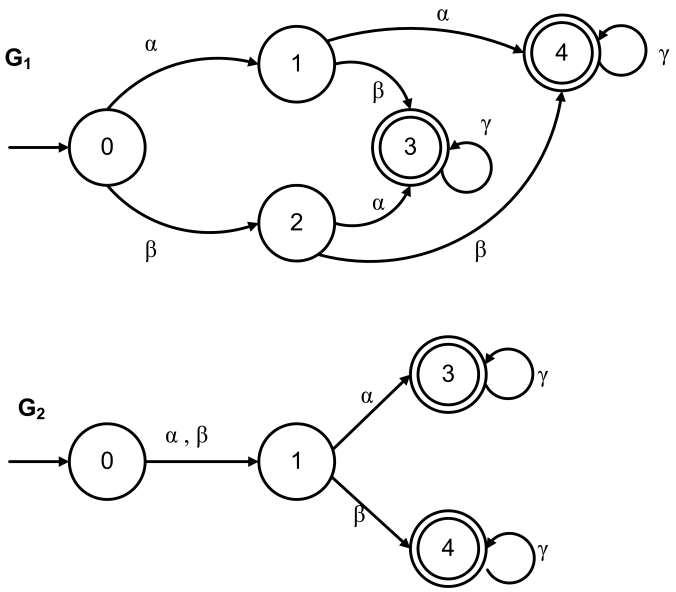



*Figure 2.4: Language-equivalent automata (Example 2.2.3)*

**Theorem 2.2.1.** ***(Deterministic Automata for a Nondeterministic Automata)*** *Let $L$ be a language recognized by a nondeterministic finite automaton, then there exists a deterministic finite automaton that recognizes $L$ [1].*

### 2.2.2 Regular Language and Finite-State Automata

Finite automata are used to recognize languages, therefore for every regular language, a finite automaton can be constructed which can recognize the language. However, there are languages which can not be recognized by finite automata.

**Definition 2.2.5.** ***(Finite Automata and Regular Languages)*** *[17, 52]*
*A language $L \in \Sigma^*$ is said to be regular, if it can be recognized by a finite automaton.*

**Definition 2.2.6.** *A language $L$ over $\Sigma$ is recognized by a finite automaton if there exists a finite automaton such that $L$ is the language recognized by this finite automaton.*

On the other hand, languages that are not regular require devices other than finite automata to recognize them. John Myhill and Anil Nerode have studied testing methods for

regularity. The idea behind these test methods is presented in [39, 60].

**Definition 2.2.7.** ***(distinguishing string)***
*Let $G = (X, \Sigma, f, x_0, X_m)$ be a deterministic finite automaton. The string $s \in \Sigma^*$ is a distinguishing string for states $x_1, x_2 \in X$ if exactly one of $f(x_1, s)$ and $f(x_2, s)$ is in $X_m$.*

Intuitively, a distinguishing string s for states $x_1$ and $x_2$ in the automaton $G$ is a string which is mapped to an accepted state from exactly one of $x_1$ and $x_2$. In other words, the existence of a distinguishing string for a pair of states $x_1,\ x_2 \in X$ proves that $x_1$ and $x_2$ are not equivalent. Based on the concept of indistinguishability, Myhill and Nerod give a criterion for (non)regularity of a language. This criterion is given by the following theorem.

**Theorem 2.2.2.** ***(Non-regularity test based on Myhill-Nerode's)*** *[39]*
*The language $L$ over alphabet $\Sigma$ is nonregular if and only if there is an infinite subset of $\Sigma^*$, whose strings are distinguishable with respect to $L$.*

### 2.2.3 Operation on Automata

As other tools, finite automata requires a set of operation that can be used to analyze automata. These operations are presented in this section.

#### Accessible part

States that never can be reached are clearly unnecessary. Also, transitions associated with such states are unnecessary. $Ac(G)$ denotes the operation for deleting these unnecessary states and transitions. The $Ac(G)$ operation has no effect on $\mathcal{L}(G)$ or $\mathcal{L}_m(G)$.

#### Coaccessible part

A state x of an automaton G is said to be coaccessible if there is a string s that takes from x to a marked state, that is $f(x, s) \in X_m$. We denote the operation of deleting all the states of L that are not coaccessible by $CoAc(G)$. The $CoAc$ operation may shrink $\mathcal{L}(G)$ but does not affect $\mathcal{L}_m(G)$. If $G = CoAc(G)$ then $L$ is said to be coaccessible. If an automaton

is nonblocking then it also have to be coaccessible. If there is no path from every state to a marked state then it can't be nonblocking.

**Trim operation**

An automaton that is both accessible and coaccessible is said to be trim. The trim operation is defined as

$$Trim(G) := CoAc(Ac(G)) = Ac(CoAc(G))$$

It does not matter in which order $Ac$ and $CoAc$ is applied.

**Complement**

Suppose we have a trim automaton $G = (X, \Sigma, f, x_0, X_m)$ that marks the language $\mathcal{L}_m(G) \subseteq \Sigma^*$. We can build another complement automaton that marks $\Sigma^* \setminus \mathcal{L}_m(G)$, which we denote $G^{comp}$.

1. Add an unmarked state $x_d$, called "dump" or "dead" state.
2. Complete the transition function $f_G$ of $G$ and make it a total function, $f_G^{tot}$, by assigning all undefined $f_G(x, \sigma)$ in $G$ to $x_d$. Furthermore $f_G^{tot}(x_d, \sigma) = x_d$ for all events $\sigma \in \Sigma$.
3. Mark all unmarked states (including $x_d$), and unmark all marked states.

$$\begin{aligned} G^{comp} &= (\{X \cup x_d\}, \Sigma, f^{tot}, x_0, \{X \cup x_d\} \setminus X_m) \\ \mathcal{L}(G^{comp}) &= \Sigma^* \text{ and } \mathcal{L}_m(G^{comp}) = \Sigma^* \setminus \mathcal{L}_m(G), \text{ as desired} \quad . \end{aligned}$$

**Composition operations**

Combining tow or more automata is considered the most important operation on the automata, because it plays an important role in problems of synthesis and decomposition of automata. "The most important and the most frequently used compositions of automata are the direct product, superposition and feedback" [40]. We define two operations:

1. Parallel composition, denoted $\|$. Also called synchronous composition.

2. The product, denoted $\times$. Also called a completely synchronous composition.

To illustrate the both operations let $G_1 = (X_1, \Sigma_1, f_1, x_{0,1}, X_{m,1})$ and $G_2 = (X_2, \Sigma_2, f_2, x_{0,2}, X_{m,2})$. The parallel composition of $G_1$ and $G_2$ is given by:

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

with

$$f_{1\|2}((x_1, x_2), \sigma) := \begin{cases} (f_1(x_1, \sigma_1), f_2(x_2, \sigma_2)) & \text{if } \sigma \in \Gamma_1(x_1) \cap \Gamma_2(x_2), \\ (f_1(x_1, \sigma_1), x_2) & \text{if } \sigma \in \Gamma_1(x_1) \setminus \Sigma_2, \\ (x_1, f_2(x_2, \sigma_2)) & \text{if } \sigma \in \Gamma_2(x_2) \setminus \Sigma_1, \\ \text{undefined} & \text{else} \end{cases}$$

The product of $G_1$ and $G_2$ is the automaton

$$G_1 \times G_2 := Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, f_{1x2}, (x_{0,1}, x_{0,2}), X_{m,1} \times X_{m,2})$$

with

$$f_{1\times 2}((x_1, x_2), \sigma) := \begin{cases} (f_1(x_1, \sigma_1), f_2(x_2, \sigma_2)) & \text{if } \sigma \in \Gamma_1(x_1) \cap \Gamma_2(x_2), \\ \text{undefined} & \text{else} \end{cases}$$

In the Parallel composition, a common event, i.e.,an event in $\Sigma_1 \cap \Sigma_2$, can only be executed if two automata both execute it simultaneously. Thus, the two automata are "synchronized"on the common events. For this reason, this operation is also called *synchronous composition*. The other events, i.e., those in $(\{\Sigma_1 \setminus \Sigma_2\} \cup \{\Sigma_2 \setminus \Sigma_1\})$ are not the subject to such constrain and can be executed whenever possible. Also note that if $\Sigma_1 = \Sigma_2$ the parallel composition reduces to a product.

The parallel composition and the product of automata are more shown in the following example.

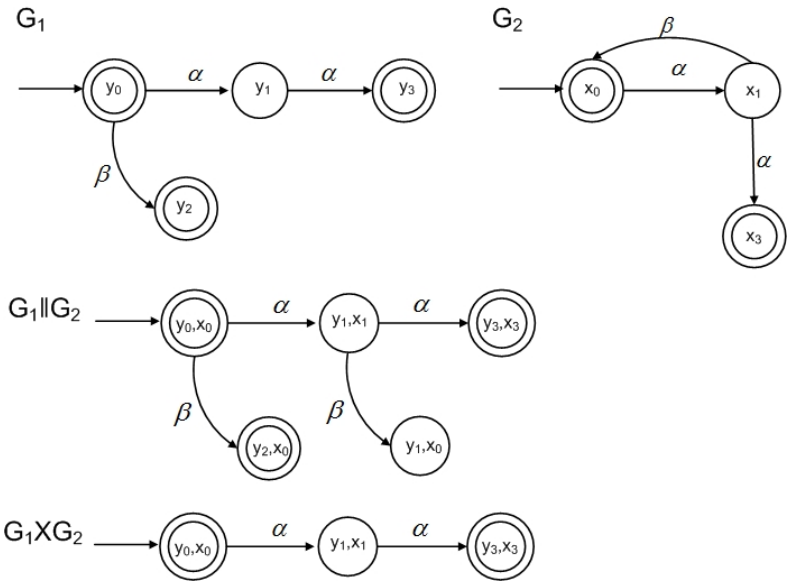



*Figure 2.5: Parallel composition automata and Product automata of (Example 2.2.4)*

**Example 2.2.4.** *Consider the following automata $G_1$ and $G_2$ in Figure 2.5, the set of common events is $\{\alpha, \beta\}$. As in case of $G_1 \parallel G_2$, the states of the product are denoted by pairs. In the case of $G_1 \parallel G_2$ the state is marked only if it is marked in $G_1$ as well as in $G_2$. The transitions in $G_1 \parallel G_2$ can be executed without the participation of $G_1$ and $G_2$ together. A transition in $G_1 \times G_2$ can be executed only and only if it is a common event in $G_1$ and $G_2$. Further on,all states in $G_1 \parallel G_2$ and $G_1 \times G_2$ are reachable from the initial state $(y_0, x_0)$.*

## 2.3 Supervisory Control of Discrete Event Systems

In this section concepts and results for control of discrete event systems are presented. It will be discussed in this section the control using complete observations and control using partial observations. The theory of supervisory control of discrete event systems was introduced by Ramadge and Wonham [45] for designing controllers so that the controlled system satisfies certain desired qualitative constraints. Many extensions of the basic supervisory control problem such as control with partial observations presented by Lin [31]and

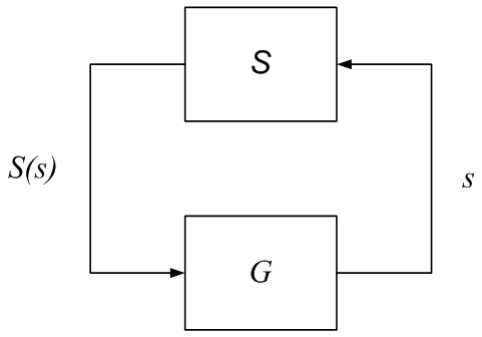



Figure 2.6: The closed-loop of supervisory control

decentralized control have been studied.

### 2.3.1 Representation of Controlled Discrete Event Systems

In this section we present the concepts of a controlled discrete event system using finite automata and in a language framework. Control of discrete event systems is based on the principle of feedback. Therefore, the observation of the system is used by a controller to generate the input to the system such that the closed-loop system meets control specifications. Feedback with full observation is illustrated in Figure 2.6. $G$ represents the uncontrolled system and $S$ is the supervisor (controller). For discrete event systems, supervisory control refers to design of a controller to satisfy desired specifications. Typically, the control problem requires constructing a supervisor $S$ for the system $G$ such that the controlled behaviour of $G$ follows some given specifications. Based on the definition 2.2.2, the language $\mathcal{L}_m(G)$ describes the desirable nonblocking language of the system. $\mathcal{L}(G)$ presents the language which can be generated by the system. Furthermore, the set $\Sigma$ is partitioned events two classes of events. The first class is called controllable events ($\Sigma_c$), i.e. events which can be prevented from occurring. These events can be enabled (allowed to occur) or disabled (prevented from occurring) in particular instances of time during the system's evolution. The events in the second class denote the uncontrollable events ($\Sigma_{uc}$) which can not be disabled. Enabling and disabling certain events in a particular state are determined by a control pattern for that

state.

**Definition 2.3.1.** ***(Control Pattern and Set of control Patterns) [6, 52]***
*Let $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$ be an event set with the controllable events $\Sigma_c$ and the uncontrollable events $\Sigma_{uc}$. A control pattern is a set $\lambda$ with $\Sigma_{uc} \subseteq \lambda \subseteq \Sigma$. The set of all control patterns is given by*

$$\Lambda := \{\lambda | \Sigma_{uc} \subseteq \lambda \subseteq \Sigma\} \subseteq 2^{\Sigma}$$

The control pattern of the discrete event system is described as follows. The controller $S$ is controlling the transition function of $G$ by enabling or disabling the controllable events of $G$.

**Definition 2.3.2.** ***(Supervisor $S$)***
*Formally, a supervisor $S$ is a function from the language generated $\mathcal{L}(G)$ by $G$ to the power set of $\Sigma$.*

$$S : \mathcal{L}(G) \to \Lambda$$

*Where $S(s)$ represents the set of enabled events after the occurrence of a string $s \in \mathcal{L}(G)$.*

Figure 2.6 shows an illustration of the interaction between the supervisor $S$ and the control system in a feedback loop. Events occurring in the system $G$ are observed by the supervisor, but the supervisor can only disable the controllable events after any string $s \in \mathcal{L}(G)$. Therefore the supervisor $S$ is admissible if for all $s \in \mathcal{L}(G)$

$$\{\Sigma \cap \lambda(f(x_0, s))\} \subseteq S(s)$$

where $\Lambda(f(x_0, s))$ represents the control pattern of the supervisor. That means the supervisor is not capable to disable uncontrollable events based on the definition of control patterns. So given $G$ and admissible $S$ , the supervisory controller restrict the behavior of the system $G$ to a desired behavior $S/G$. It follows that the generated and the marked languages are the subsets of $\mathcal{L}(G)$ and $\mathcal{L}_m(G)$ respectively. Note that those languages contain strings that remain a subject to control in the presence of the controller. Formally, the language

generated and the language controlled by $S$, $\mathcal{L}(S/G)$ and $\mathcal{L}_m(S/G)$ respectively are defined as follows:

**Definition 2.3.3.** ***(Control System) [6]***

*Let $G$ be a system, let $S$ be a supervisor. The language generated by $S/G$ is defined as*

1. $\epsilon \in \mathcal{L}(S/G)$

2. $s\sigma \in \mathcal{L}(S/G) \Leftrightarrow s \in \mathcal{L}(S/G)$
$\wedge\, s\sigma \in \mathcal{L}(G)$
$\wedge\, \sigma \in S(s)$

*The language marked by $S/G$, denoted by $\mathcal{L}_m(S/G)$ is defined as*

$$\mathcal{L}_m(S/G) = \mathcal{L}(S/G) \cap \mathcal{L}_m(G)$$

The next example illustrates the operation and the concept of supervisory control.

**Example 2.3.1.** *Let $G$ as illustrated in Figure 2.7. Also define the uncontrollable event set $\Sigma_{uc} = \{\beta_1, \beta_2\}$ , and the controllable event set $\Sigma_c = \{\alpha_1, \alpha_2\}$. Assuming that we don't want the system to get to state $x = 3$. So we design a supervisor which disables the both events $\{\alpha_1, \alpha_2\}$ which bring the system to that state. We define a supervisor $S$ for the control system $G$ for strings $s \in \mathcal{L}(G)$ as*

$$S(s) := \begin{cases} \{\beta_1, \beta_2\} & \text{if } s = \{\alpha_1\alpha_2, \alpha_2\alpha_1\}, \\ \{\alpha_1, \alpha_2\} & \text{otherwise} \end{cases}$$

*That is if $s = \alpha_1$ and $\sigma = \alpha_2$ then we have $s\sigma = \alpha_1\alpha_2 \in \mathcal{L}(G)$, but $s\sigma \notin \mathcal{L}(S/G)$. This means $S$ disables the event $\alpha_2$ after $\alpha_1$ has occurred, and similarly disables $\alpha_1$ after $\alpha_2$. The resulting closed-loop behavior $S/G$ is shown in Figure 2.7.*

### 2.3.2 Controllability

In the previous section, the concept of the control system and supervisors were introduced. The main objective of the supervisor is to modify the open-loop behavior of the

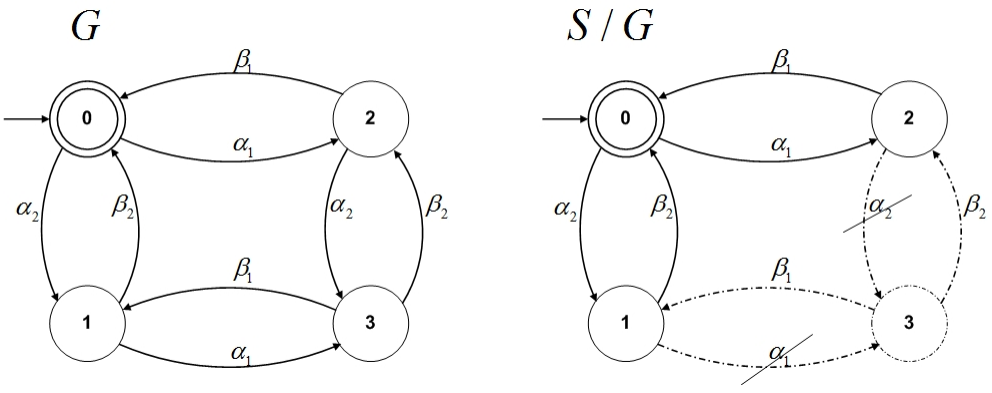



*Figure 2.7: The control system $G$ and the supervisor $S/G$*

system so that it satisfies some specified constraints. By describing the system with the generator $G$ and the language generated $\mathcal{L}(G)$ and specifying a supervisor by an automaton $S$ , the desired system properties are formulated by $\mathcal{L}(S/G)$ which is determined by coupling $S$ and $G$ together. $\mathcal{L}(S/G)$ is defined as the subset of possible strings which can be executed by the supervisor. The language $\mathcal{L}(S/G)$ represents the desired behavior of the system. The desired behavior of the system can be realized using the controllability theory.

The concept of controllability is close to the concept of feedback control. This concept plays an important role in theoretical and practical aspects of modern control theory. This concept can be formulated formally as follows.

**Definition 2.3.4.** ***(Controllability) [45]***

*Consider a discrete event system $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ where $\Sigma_{uc} \subseteq \Sigma$ is the set of uncontrollable events. Let $\mathcal{L}(G) = \overline{\mathcal{L}(G)}$ be a prefix-closed language. Let $K$ and $\mathcal{L}(G)$ be languages over an event set $\Sigma$. The language $K$ is said to be controllable w.r.t,$\mathcal{L}(G)$ and the set of uncontrollable events $\Sigma_{uc}$ if*

$$\overline{K}\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}$$

In a more intuitive way, the controllability condition can be written as follows. $K$ is controllable w.r.t $\mathcal{L}(G)$ if an event of uncontrollable events occurs, then the path along which that event occurred must remain in $K$. That is

$$\text{for all } s \in \overline{K}, \text{ for all } \sigma \in \Sigma_{uc}, s\sigma \in \mathcal{L}(G) \Rightarrow s\sigma \in \overline{K}$$

In [6, 45] it has been shown that if the desirable behavior of a system which is described by the language $K$, where $K$ is controllable w.r.t $\mathcal{L}(G)$ and prefix closed, then there exists a supervisor $S$ such that $\mathcal{L}(S/G) = \overline{K}$. In addition to the controllability property of a control system, the *nonblocking behavior* is also meaningful for the control system.

**Definition 2.3.5.** ***(Nonblocking Control System) [6]***
*Given a plant $G$ and $\mathcal{L}_m(G), \mathcal{L}(G)$. Let $K \subseteq \mathcal{L}_m(G)$, be the system's desired behavior language. The supervisor $S$ is nonblocking if*

$$\overline{\mathcal{L}_m(S/G)} = \mathcal{L}(S/G)$$

*Otherwise, if $\mathcal{L}(S/G) \neq \overline{\mathcal{L}_m(S/G)}$ then we say $S$ is blocking.*

As we noted in the previous sections, the supervisory control problem refers to design a supervisor that achieves desired specifications. These specifications require that the system satisfy a certain progress property, which is equivalent to the system being nonblocking. Therefore, the supervisor must be fair in a way that it never blocks the system with respect to any of the specification languages [10]. The following theorem presents the *Nonblocking Controllability Theorem*, which is an extended version of the controllability theorem to nonblocking supervisors.

**Theorem 2.3.1.** ***(Nonblocking Controllability Theorem) [45]***
*Consider a discrete event system $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ where $\Sigma_{uc} \subseteq \Sigma$ is the set of uncontrollable events. Let the language $K \subseteq \mathcal{L}_m(G)$, where $K \neq \emptyset$. There exists a nonblocking supervisor $S : \mathcal{L}(G) \to \Lambda$ such that*

$$\mathcal{L}_m(S/G) = K$$
$$\mathcal{L}(S/G) = \overline{K}$$

*if and only if the tow following conditions hold:*

1. *Controllability: $K$ is controllable with respect to $\mathcal{L}(G)$.*

*2.* $K = \overline{K} \cap \mathcal{L}_m(G)$.

A proof of Theorem 2.3.1 is given in [45]. Clearly that the above theorem is applicable if both conditions are satisfied. If the language $K$ is not controllable with respect to $\mathcal{L}(G)$, then we need to investigate the sublanguage of $K$ to achieve controllability. The class of a controllable sublanagauges of a given language $K$ are defined as follows.

**Definition 2.3.6.** ***Supremal Controllable Sublanguage***
*Let* $K = \overline{K} \subseteq \mathcal{L}(G)$ *be a prefix-closed language and* $\Sigma_{uc} \subseteq \Sigma$ *be the set of uncontrollable events. The class of controllable sublanguage of* $K$ *is*

$$C_{in}(K) = \{L \subseteq K : \overline{L}\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \overline{L}\}$$

The set $C_{in}(K)$ is closed under an arbitrary union. There exist a unique largest language in $C_{in}(K)$ and it is given by:

$$K^{\uparrow c} = \bigcup_{L \in C_{in}(K)} L$$

Where $K^{\uparrow c}$ is controllable and a subset of $K$. $K^{\uparrow c}$ is called the *supremal controllable sublanguage* of $K$.

**Definition 2.3.7.** ***Infimal Closed and Controllable Superlanguage***
*Let* $K = \overline{K} \subseteq \Sigma^*$ *be a prefix-closed language and* $\Sigma_{uc} \subseteq \Sigma$ *be the set of uncontrollable events. The class of controllable sublanguage of* $K$ *is*

$$CC_{out}(K) = \{L \subseteq \Sigma^* : K \subseteq L \subseteq \mathcal{L}(G) \wedge L = \overline{L} \wedge \overline{L}\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \overline{L}\}$$

The set $CC_{out}(K)$ is closed under an arbitrary intersection. There exist a unique smallest language in $CC_{out}(K)$ and it is given by:

$$K^{\downarrow c} = \bigcap_{L \in CC_{out}(K)} L$$

Where $K^{\downarrow c}$ is controllable and a subset of $K$. $K^{\downarrow c}$ is called the *Infimal closed and controllable superlanguage* of $K$. The main focus of the previous sections was the understanding of

controllability in the framework of discrete event system. We have presented the control of discrete event systems based on the principle of feedback. The supervisor observes all the system events and used to generate the input of the system. So the closed-loop system meets control specifications. Now in case if the supervisor does not observe all events that the system executes. The next section describes the control system under a partial observation of the system.

## 2.4 Control Under Partial Observation

We consider the situation, when the supervisor is not able to observe the occurrences of all events. In this case the existence of the supervisor is no longer guaranteed by controllability alone. Therefore, the concept of observability was introduced in [31]. In such a situation the event set $\Sigma$ is partitioned into tow disjoint subsets

$$\Sigma \quad = \quad \Sigma_o \dot{\cup} \Sigma_{uo}$$

Where

- $\Sigma_o$ is the set of events that can be seen by the supervisor (observable events).
- $\Sigma_{uo}$ is the set of events(unobservable events) that the supervisor can not see.

With the set of observable events $\Sigma_o$ a projection $P$ is associated such that

$$P : \Sigma^* \to \Sigma_o^*$$

Due to the presence of the projection $P$ in the feedback loop of the controlled system under partial observation, the feedback loop of supervisory control in this case is shown in Figure 2.8. As it is depicted in Figure 2.8, the supervisor produces the control on the bases of the projection $P$ of the string $s$. If there exist $(s, \grave{s}) \in \mathcal{L}(G)$ such that $P(s) = P(\grave{s})$, then the supervisor will execute the same control action, $S_P(P(s))$. Therefore, the definition of the partial-observation supervisor is given as the function

$$S_P : P(\mathcal{L}(G)) \to 2^{\Sigma}$$

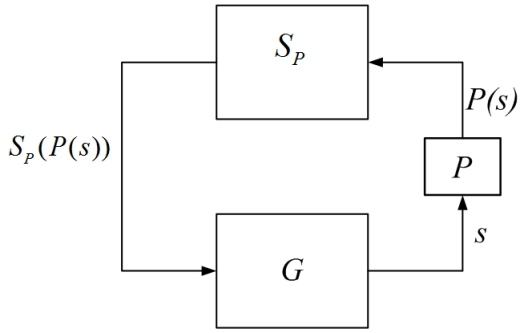



Figure 2.8: The closed-loop of supervisory control under partial observation.

Where $S_P$ is the supervisor. The supervisor can execute a controllable event only after the occurrence of an observable event. Furthermore, the language generated by the supervisor $\mathcal{L}(S_P/G)$ and language marked $\mathcal{L}_m(S_P/G)$ are defined in the same way as $\mathcal{L}(S/G)$ and $\mathcal{L}_m(S/G)$.

### 2.4.1 Observability

In the previous section we presented the case of the supervisor being not able to observe all events generated by the system. The fact that some of the events might not be observable to the supervisor has to be taken into account while designing the supervisor. Given the language $K \subseteq \mathcal{L}(G)$ describing the desired behavior of the controlled system, does there exist a supervisor $S_P$ such that $\mathcal{L}(S_P/G) = K$? The existence of such a supervisor is closely related to the concept of observability [31].

**Definition 2.4.1.** ***Observability [31]***
*Let $K \subseteq \mathcal{L}(G)$, and $\Sigma_o \subseteq \Sigma$ be the set of observable events with $P : \Sigma^* \to \Sigma_o^*$ be the corresponding natural projection. $K$ is said to be observable with respect to $\mathcal{L}(G)$ and $P$ if*

$$(\forall s, \grave{s} \in \Sigma^*) P(s) = P(\grave{s}) \Rightarrow (\forall \sigma \in \Sigma)(s\sigma \in \overline{K} \wedge \grave{s} \in \overline{K} \wedge \grave{s}\sigma \in \mathcal{L}(G) \Rightarrow \grave{s}\sigma \in \overline{K})$$

Intuitively, observability requires that two strings look the same, they must be consistent in control action [31]. In other words, the language $K$ is observable if the projection

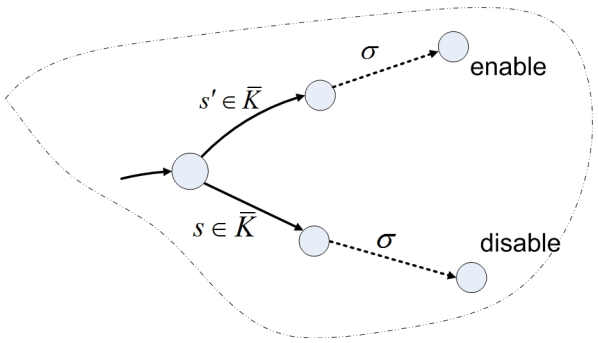



Figure 2.9: Observability Concept

$P$ has to retain sufficient information for a supervisor to decide whether after the enabling or disabling of a particular event the produced string is in $K$. Figure 2.9 shows a graphical representation of the two strings look the same to the supervisor, because they have the same projection. Therefore the supervisor cannot decide to enable $\sigma$ or to disable it. In this case the observability theorem is used to investigate if the system contains such situations or not. As is mentioned the supervisor $S_p$ exists if and only if the language $K$ is controllable and observable. This leads to the controllability and observability theorem presented in [31].

**Theorem 2.4.1.** ***(Controllability and Observability Theorem) [31]***
*Consider $G$ with controllable event set $\Sigma_c$ and observable event set $\Sigma_o$. Let a language $K \subseteq \mathcal{L}_m(G)$, where $K \neq \emptyset$. There exists a nonblocking P-supervisor $S_p$ for $G$ such that*

$$\mathcal{L}_m(S_p/G) = K$$
$$\mathcal{L}(S_p/G) = \overline{K}$$

*If and only if the following three conditions are satisfied*

1. *$K$ is controllable with respect to $\mathcal{L}(G)$ and $\Sigma_{uc}$.*
2. *$K$ is observable with respect to $\mathcal{L}(G)$ and $P$.*
3. *$K$ is $\mathcal{L}_m(G)-$closed.*

A proof of the theory is given in [31].

The notion of controllability and observability theory plays an important role in the framework of the discrete event system. However, it is well known that the natural projection of a discrete event system might create more complexity in the term of the computation. In hierarchical control of discrete-event systems [12, 13, 26] the concept of the observer introduced, where it was shown to be very important by ensuring the preservation of the nonblocking property from a high level control synthesis to its low level implementation. A detailed discerption of the way to construct an observer automaton is given in [6].

**Example 2.4.1.** *Let $G$ and $K$ as depicted in Figure 2.10. Let the controllable event set $\Sigma_c = \{\alpha_1, \alpha_2\}$, and the observable event set $\Sigma_o = \{\beta_1, \alpha_2\}$. Is the language $K$ controllable with respect to $\mathcal{L}(G)$ and $\Sigma_{uc}$? Is $K$ observable with respect to $\mathcal{L}(G)$ and $P$? The uncontrollable events should not take the plant $G$ outside the language of $K$. We see here clearly that controllability for $K$ depends on the set $\{\alpha_1, \alpha_2\}$ which are controllable. Therefore, $K$ is controllable with respect to $\mathcal{L}(G)$ and a set of uncontrollable events $\Sigma_{uc}$.*

*To test the observability let construct an observer as follows: In $K$, we cannot distinguish between state (1) and (2) based on the unobservable events. When $\beta_1$ occurs, we know we are at state (4). Afterwards, when $\alpha_2$ occurs, then we do not know in which state of the states (1,2,3,5). If $\beta_1$ occurs then we will be in either in state (4) or state (6). Same if $\alpha_2$ occurs we are back to one of the states (1,2,3,5). Now we compare the conflicting states in $G$ and check whether there is any mismatch between in disabling and enabling controllable events. We see that at states (2,4),(1,4),(1,6),(3,4) and (3,6) the controllable events are disabled by $K$ and enabled by $G$, and we can conclude that $K$ is observable.*

In case that the language $K$ does not satisfy the observability condition there is again the possibility to modify $K$ such the sublanguage of $K$ achieves observability. The class of observable sublanagauges of a given language $K$ are defined as follows.

**Definition 2.4.2.** ***The Infimal Observable and Closed Superlanguage***

*Let $K \subseteq \mathcal{L}(G)$ be a prefix-closed language and $\Sigma_{uo} \subseteq \Sigma$ be a set of unobservable events. The class of observable sublanguage of $K$ is*

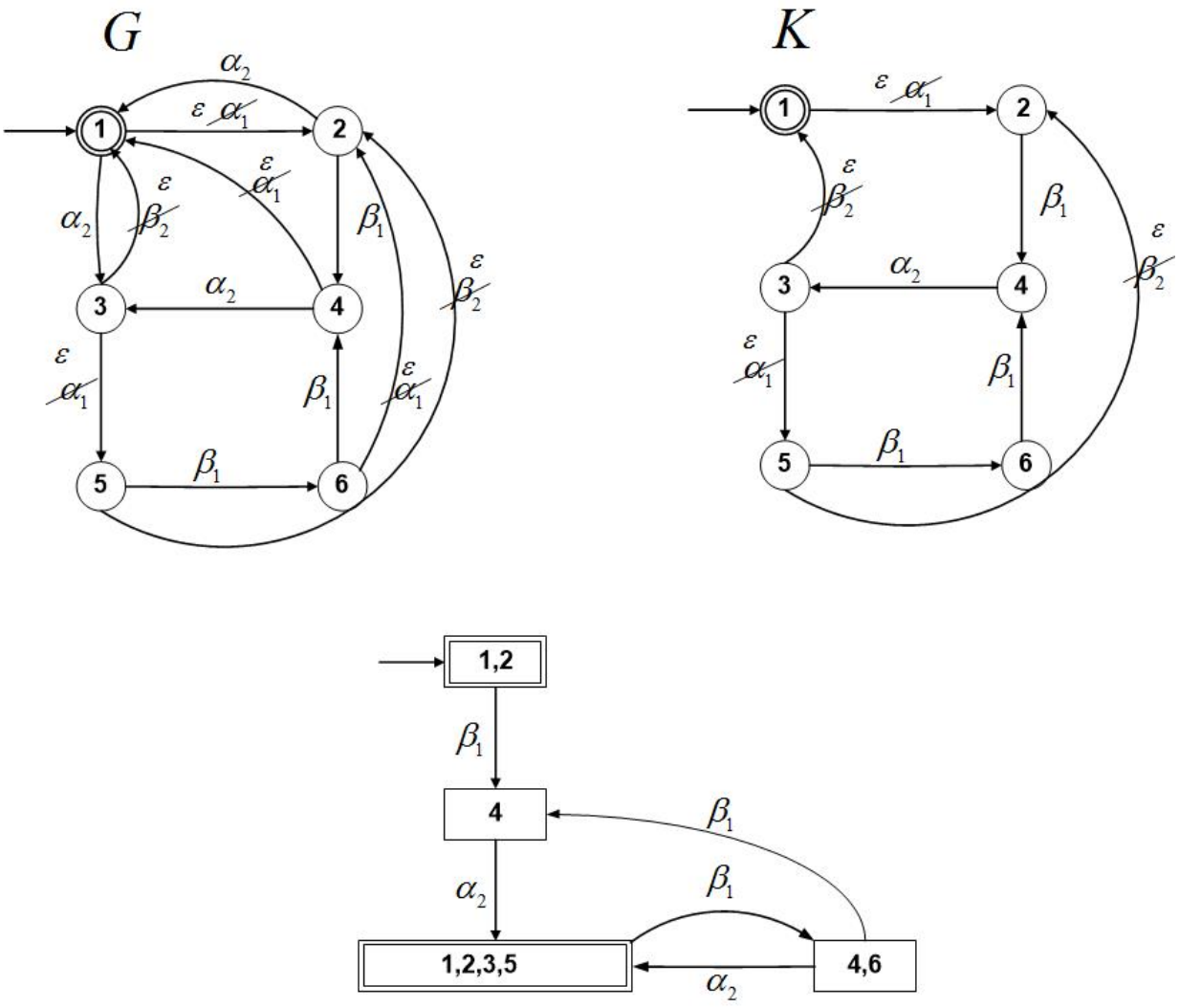



*Figure 2.10: Automaton of $G$, $K$ and the observer of $K$*

$$CO_{out}(K) = \{L \subseteq \mathcal{L}(G) : K \subseteq L \wedge L = \overline{L} \wedge\ L \textit{ is observable}\}$$

*The set $CO_{out}(K)$ is closed under an arbitrary intersection. There exist a unique smallest language in $CO_{out}(K)$ and it is given by:*

$$K^{\downarrow o} \quad = \quad \bigcap_{L \in CO_{out}(K)} L$$

*$K^{\downarrow o}$ is called the* infimal observable and closed superlanguage *of $K$.*

In [6] properties of observability are introduced. It is shown that the property of observability is preserved under intersection, but not preserved under union. In view of the results presented in [6], the supremal observable sublanguage of a given language may not exist, but a maximal element can always be found. Also in view of the properties of the observability and controllability, the infimal element exists and defined as follows.

**Definition 2.4.3.** ***Infimal Closed, Controllable and Observable Superlanguage***
*Let $K \subseteq \mathcal{L}(G)$ be a prefix-closed language and $\Sigma_{uc} \subseteq \Sigma$ be the set of uncontrollable events. Let $\Sigma_{uo} \subseteq \Sigma$ be the set of unobservable events. The class of closed, controllable and observable superlanguage of $K$ is*

$$CCO_{out}(K) = \{L \subseteq \mathcal{L}(G) : K \subseteq L \wedge L = \overline{L} \wedge\ L \textit{ is controllable and observable}\}$$

*The set $CCO_{out}(K)$ is closed under an arbitrary intersection. There exist a unique smallest language in $CCO_{out}(K)$, and it is given by:*

$$K^{\downarrow c} \quad = \quad \bigcap_{L \in CC_{out}(K)} L$$

*$K^{\downarrow co}$ is called the* Infimal closed, controllable and observable superlanguage *of $K$.*

As we pointed out earlier that because the lack closure of observability under union, we cannot find a supremal element. Consequently, the supremal observable and controllable sublanguage may also not exist. Therefore, a new language property called normality is presented [6].

### 2.4.2 Normality

The property of normality was introduced to solve some supervisory control and observation problems. The property of normality is closed under arbitrary union, therefore present a unique solution of basic supervisory control and observation problem.

**Definition 2.4.4.** *A language $K \subseteq \mathcal{L}(G)$ is normal with respect to $\mathcal{L}(G)$ and natural projection $P$ if*

$$\overline{K} = P^{-1}P(\overline{K}) \cap \mathcal{L}(G)$$

The Figure 2.11 shows an illustration of two cases. In one $K$ is not normal and the other $K$ is normal. Clearly we can see from the Figure 2.11 that $K$ in (a) is not normal because $P^{-1}P(\overline{K}) \cap \mathcal{L}(G)$ is bigger than $\overline{K}$, and in (b) $P^{-1}P(\overline{K}) \cap \mathcal{L}(G) = \overline{K}$.

**Theorem 2.4.2.** ***(Normality and Observability) [6]***
*If $K \subseteq \mathcal{L}(G)$ is normal with respect to $\mathcal{L}(G)$ and $P$, then $K$ is observable with respect to $\mathcal{L}(G)$ and $P$. However, the converse statement is not true.*

A proof of the theorem is given in [6].

With the property of normality we present a class of a normal sublanagauges of a given language $K$ as follows.

**Definition 2.4.5.** ***Supremal Normal Sublanguage***
*Let $K \subseteq \mathcal{L}(G)$ be a prefix-closed language and $\Sigma_{uo} \subseteq \Sigma$ be a set of unobservable events. The class of normal sublanguage of $K$ is*

$$N_{in}(K) = \{L \subseteq \mathcal{L}(G) : K \subseteq L \wedge L = \overline{L} \wedge \ L \text{ is normal}\}$$

*The set $N_{in}(K)$ is closed under an arbitrary union. There exist a unique largest language in $N_{in}(K)$ and it is given by:*

$$K^{\uparrow N} \ = \bigcup_{L \in N_{in}(K)} L$$

*$K^{\uparrow N}$ is called the* supremal normal sublanguage *of $K$.*

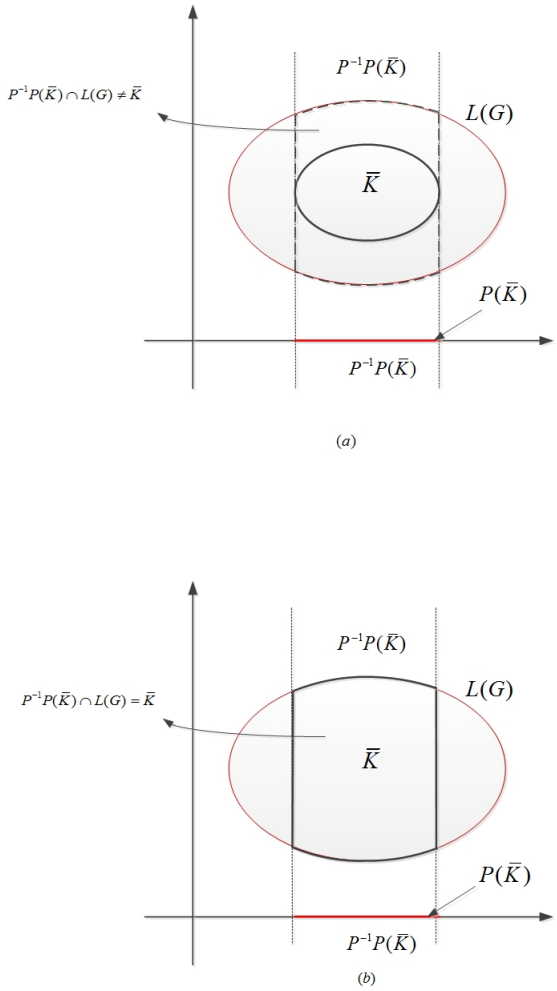



Figure 2.11: (a)$K$ is not normal and (b) $K$ is normal

**Definition 2.4.6.** ***Supremal controllable and normal sublanguage***

*Let $K \subseteq \mathcal{L}(G)$ be a prefix-closed language, $\Sigma_{uo} \subseteq \Sigma$ be a set of unobservable events and $\Sigma_{uc} \subseteq \Sigma$ be a set of uncontrollable events. The class of controllable and normal sublanguage of $K$ is*

$$CN_{in}(K) = \{L \subseteq \mathcal{L}(G) : K \subseteq L \wedge L = \overline{L} \wedge \ L \text{ is controllable and normal}\}$$

*The set $CN_{in}(K)$ is closed under an arbitrary union. There exist a unique largest language in $CN_{in}(K)$ and it is given by:*

$$K^{\uparrow CN} \quad = \quad \bigcup_{L \in CN_{in}(K)} L$$

*$K^{\uparrow CN}$ is called the* supremal controllable and normal sublanguage *of $K$.*

Both previous definitions lead to an interesting theorem which considers the equivalency of normality and observability.

**Theorem 2.4.3.** ***Equivalence of normality and observability*** *[6]*

*Assume that $\Sigma_c \subseteq \Sigma_o$. If $K$ is controllable with respect to $\mathcal{L}(G)$ and $\Sigma_{uc}$ and observable with respect to $\mathcal{L}(G)$ and $P$, then $K$ is normal with respect to $\mathcal{L}(G)$ and $P$.*

In other words, if a language is normal and controllable, then it is observable and controllable under the condition that $\Sigma_c \subseteq \Sigma_o$.

The main focus of the previous sections was the understanding of basic control-theoretic issues such as controllability, observability and normality in the framework of discrete event systems introduced by Ramadge, Wonham and Lin in [45] [31].

# 3 Discrete Event Systems and Opacity

Opacity is a general notion that arises in activities where the users involved wish to keep their activities secret. In this context, the system behavior is opaque if the evolution of its string through a set of strings remains opaque to an observer, who is observing activity in the system. In [33] a new framework to study opacity was proposed. This framework defines strong opacity, weak opacity, and its negation. Given a general observation mapping $\theta$, a language is strongly opaque if all strings in the language are confused with some strings in another language and it is weakly opaque if some strings in the language are confused with some strings in another language. We say a language is not opaque if it is not weakly opaque. It is also shown that the three known properties in discrete event systems, namely, observability, diagnosability, and detectability, can be studied as a special cases of opacity.

In this chapter, we present the opacity in the centralized framework. The centralized setting presented in this chapter is described as follow. There exists only one agent who observes the system's activities through the general observation mapping $\theta$. First, we give a brief definition of the general mapping $\theta$. Then a discussion about the classification of opacity followed with the algorithms for checking opacity presented in [33]. Finally, we give a brief presentation of the properties of opacity. The discussion in this chapter is formal.

As it was noted earlier the theoretical developments in this dissertation are mainly language-based and automata-based. The system to be monitored or to be controlled is modeled as an automaton ($G$)

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

## 3.1 General Observation Mapping

To investigate opacity, we consider a general observation mapping as in [33]:

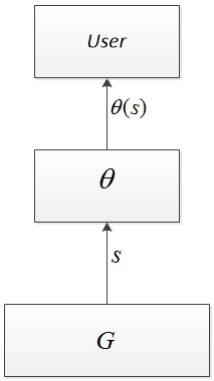



Figure 3.1: Framework for centralized opacity

$$\theta : \Sigma^* \longrightarrow \Sigma^*.$$

$\theta$ is interpreted as follows: If a string of events $s$ occurs in the system, an agent or controller will see $\theta(s)$. The natural projection $P : \Sigma^* \longrightarrow \Sigma_o^*$ is a special case of observation mapping. In general, however, $\theta$ can be any observation mapping, not restricted to the natural projection. An observation mapping $\theta$ can be extended from strings to languages as follows.

For a language $L \subseteq \Sigma^*$, its mapping under $\theta$ is defined as,

$$\theta(L) = \{t \in \Sigma^* : (\exists s \in L) t = \theta(s)\}.$$

For a language $J \subseteq \Sigma^*$ , its inverse mapping is defined as

$$\theta^{-1}(J) = \{t \in \Sigma^* : \theta(t) \in J\}.$$

The considered framework based on one user observation is described by Figure 3.1.

## 3.2 Definitions of Opacity

The strong opacity requires that the system's secret behavior is opaque to an external observer who is observing the events that occur in the system through a general observation mapping $\theta$. No opacity requires that the system's behavior should not be opaque to an observer. Opacity is defined in [33] as follows.

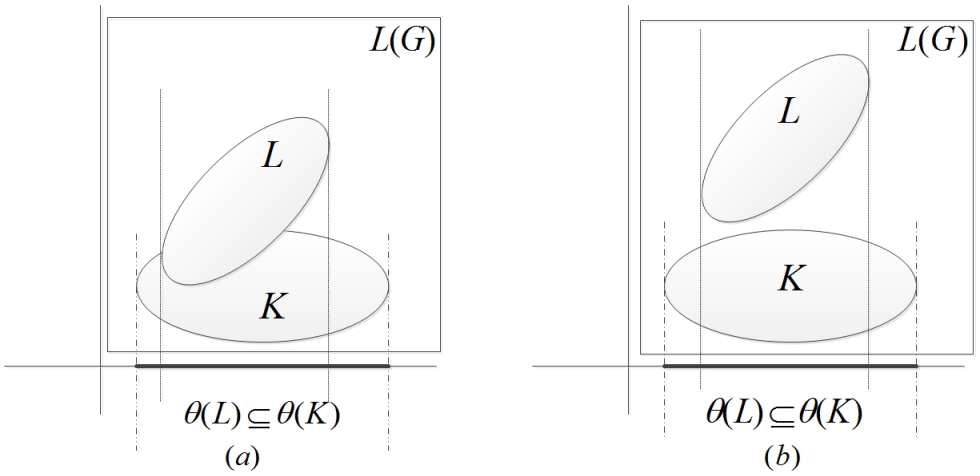



Figure 3.2: Illustration of the strong opacity

**Definition 3.2.1.** ***Strong Opacity***

*Given two languages $L, K \subseteq \mathcal{L}(G)$, $L$ is strongly opaque with respect to $K$ and $\theta$ if*

$$L \subseteq \theta^{-1}\theta(K).$$

In other words, $L$ is strongly opaque with respect to $K$ and $\theta$ if every string in $L$ is confused with some strings in $K$ under observation $\theta$. Both $L$ and $K$ are sublanguages of $\mathcal{L}(G)$, describing two properties or conditions of system $G$. Figure 3.2 presents a graphical representation of a strong opacity. In $(a)$ $L \cap K \neq \emptyset \wedge \theta(L) \subseteq \theta(K)$, and $(b)$ presents the case where $L \cap K = \emptyset$ but still $\theta(L) \subseteq \theta(K)$.

**Definition 3.2.2.** ***Weak Opacity***

*Given two languages $L, K \subseteq \mathcal{L}(G)$, $L$ is weakly opaque with respect to $K$ and $\theta$ if*

$$L \cap \theta^{-1}\theta(K) \neq \emptyset.$$

In other words, $L$ is weakly opaque with respect to $K$ and $\theta$ if some string in $L$ is confused with some strings in $K$ under observation $\theta$. Figure 3.3 presents a graphical representation of the weak opacity with two cases. In case $(a)$ $L \cap K = \emptyset$ and $\theta(L) \cap \theta(K) \neq \emptyset$. Case $(b)$ shows $L \cap K \neq \emptyset$ and $\theta(L) \cap \theta(K) \neq \emptyset$.

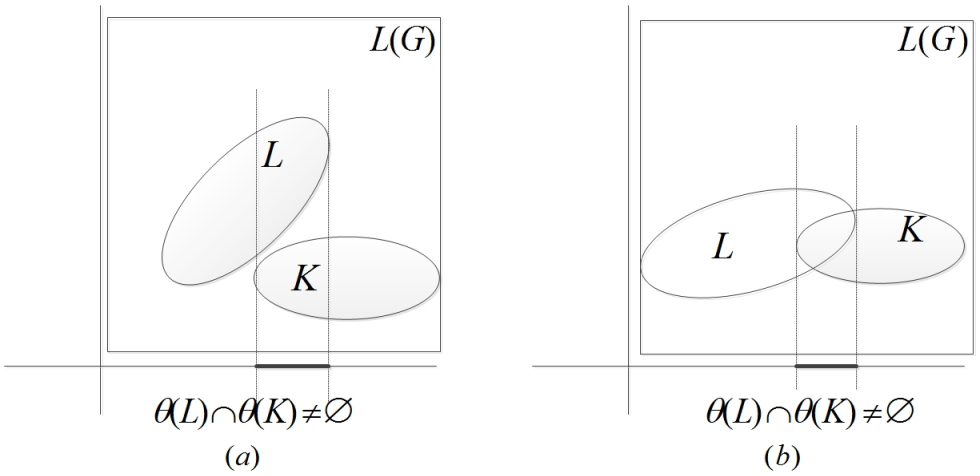



Figure 3.3: Illustration of the weak opacity

**Definition 3.2.3.** ***No Opacity***

*Given two languages $L, K \subseteq \mathcal{L}(G)$, $L$ is not opaque with respect to $K$ and $\theta$ if*

$$L \cap \theta^{-1}\theta(K) = \emptyset$$

.

In other words, $L$ is not opaque with respect to $K$ and $\theta$ if it is not weakly opaque with respect to $K$ and $\theta$, that is, no strings in $L$ are confused with some strings in $K$ under observation $\theta$. Figure 3.4 presents a graphical representation of a no-opacity. It shows that $L \cap K = \emptyset$ and $\theta(L) \cap \theta(K) = \emptyset$.

### 3.2.1 Algorithms for Checking Opacity

To check opacity, two theorems are given in [33]. The first theorem says that for any observation mapping $\theta$ and languages $L, K \subseteq \mathcal{L}(G)$,

$$L \subseteq \theta^{-1}\theta(K) \Leftrightarrow \theta(L) \subseteq \theta(K).$$

With this theorem, we are able to check the strong opacity by checking whether $\theta(L) \subseteq \theta(K)$. The second theorem says that for any observation mapping $\theta$ and languages $L, K \subseteq \mathcal{L}(G)$,

$$L \cap \theta^{-1}\theta(K) = \emptyset \Leftrightarrow \theta(L) \cap \theta(K) = \emptyset.$$

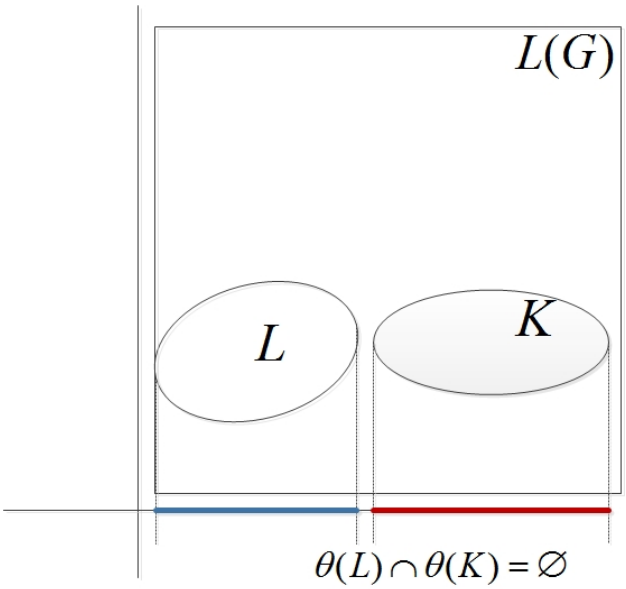



Figure 3.4: Illustration of No-opacity

Therefore, checking weak opacity can be done by checking whether $\theta(L) \cap \theta(K) \neq \emptyset$.

With these results, we can investigate the properties of different opacity classifications and find opaque superlanguages and sublanguages.

## 3.3 Properties of Opacity

In this section, we investigate the properties of various opacities of discrete event systems. We consider opacity under union, and opacity under the intersection. We first present the following two lemmas.

**Lemma 3.3.1.** *For any observation mapping $\theta$ and languages $L_i \subseteq \mathcal{L}(G), i = 1, 2, \cdots$, we have*

$$\theta(\cap_i L_i) \subseteq \cap_i \theta(L_i)$$

*Proof.* For any $t \in \Sigma^*$,

$$t \in \theta(\cap_i L_i)$$

$\Leftrightarrow (\exists s) t = \theta(s) \wedge s \in (\cap_i L_i)$

$\Leftrightarrow (\exists s) t = \theta(s) \wedge (s \in L_1 \wedge s \in L_2 \cdots)$

$\Rightarrow ((\exists s) t = \theta(s) \wedge s \in L_1) \wedge ((\exists s) t = \theta(s) \wedge s \in L_2) \cdots$

$\Leftrightarrow t \in \theta(L_1) \cap \theta(L_2) \cdots$

$\Leftrightarrow t \in \cap_i \theta(L_i)$ □

**Lemma 3.3.2.** *For any observation mapping $\theta$ and languages $L_i \subseteq \mathcal{L}(G), i = 1, 2, \cdots$ we have,*

$$\theta(\cup_i L_i) = \cup_i \theta(L_i)$$

*Proof.* For any $t \in \Sigma^*$,

$t \in \theta(\cup_i L_i)$

$\Leftrightarrow (\exists s) t = \theta(s) \wedge s \in (L_1 \cup L_2 \cdots)$

$\Leftrightarrow (\exists s) t = \theta(s) \wedge (s \in L_1 \vee s \in L_2 \cdots)$

$\Leftrightarrow ((\exists s) t = \theta(s) \wedge s \in L_1) \vee ((\exists s) t = \theta(s) \wedge s \in L_2) \cdots$

$\Leftrightarrow t \in \theta(L_1) \cup \theta(L_2) \cdots$

$\Leftrightarrow t \in \cup_i \theta(L_i)$ □

Using Lemmas 3.3.1 and 3.3.2, we can derive the following properties for strong opacity, weak opacity, and no opacity.

**Theorem 3.3.1.** *For any observation mapping $\theta$ and languages $L_i \subseteq \mathcal{L}(G), i = 1, 2, \cdots$, and $K \subseteq \mathcal{L}(G)$, if $L_i$ is strongly opaque with respect to $K$, then the intersection $\cap_i L_i$ is also strongly opaque with respect to $K$.*

*Proof.* By the definition, $L_i$ is strongly opaque with respect to $K$ means

$$\theta(L_i) \subseteq \theta(K) \Rightarrow \cap_i \theta(L_i) \subseteq \theta(K)$$

From Lemma 3.3.1:

$$\theta(\cap_i L_i) \subseteq \cap_i \theta(L_i)$$

Hence

$$\theta(\cap_i L_i) \subseteq \theta(K)$$

that is, $\cap_i L_i$ is strongly opaque with respect to $K$. □

**Theorem 3.3.2.** *For any observation mapping $\theta$ and languages $L_i \subseteq \mathcal{L}(G)$,$i = 1,2,\cdots$, and $K \subseteq \mathcal{L}(G)$, if $L_i$ is strongly opaque with respect to $K$, the union $\cup_i L_i$ is also strongly opaque with respect to $K$.*

*Proof.* By the definition, $L_i$ is strongly opaque with respect to $K$ means

$$\theta(L_i) \subseteq \theta(K) \Rightarrow \cup_i \theta(L_i) \subseteq \theta(K)$$

From Lemma 3.3.2:

$$\theta(\cup_i L_i) = \cup_i \theta(L_i)$$

Hence

$$\theta(\cup_i L_i) \subseteq \theta(K)$$

that is, $\cup_i L_i$ is strongly opaque with respect to $K$. □

**Theorem 3.3.3.** *For any observation mapping $\theta$ and languages $L \subseteq \mathcal{L}(G)$, and $K_i \subseteq \mathcal{L}(G)$,$i = 1,2,\cdots$, if $L$ is strongly opaque with respect to $K_i$ and $\theta$, then $L$ is also strongly opaque with respect to $\cup_i K_i$.*

*Proof.* By the definition, $L$ is strongly opaque with respect to $K_i$,$i = 1,2,\cdots$ means

$$\theta(L) \subseteq \theta(K_i) \Rightarrow \theta(L) \subseteq \cup_i \theta(K_i)$$

From Lemma 3.3.2:

$$\theta(\cup_i K_i) = \cup_i \theta(K_i)$$

Hence

$$\theta(L) \subseteq \theta(\cup_i K_i)$$

that is, $L$ is strongly opaque with respect to $\cup_i K_i$. □

**Theorem 3.3.4.** *For any observation mapping $\theta$ and languages $L_i \subseteq \mathcal{L}(G)$,$i = 1,2,\cdots$, and $K \subseteq \mathcal{L}(G)$, if $L_i$ is weakly opaque with respect to $K$ and $\theta$, then the union $\cup_i L_i$ is also weakly opaque with respect to $K$.*

*Proof.* By the definition, $L_i$ is weakly opaque with respect to $K$ means

$$\theta(L_i) \cap \theta(K) \neq \emptyset \Rightarrow \cup_i (\theta(L_i) \cap \theta(K)) \neq \emptyset$$

From Lemma 3.3.2:

$$\begin{aligned}
&\theta(\cup_i L_i) = \cup_i \theta(L_i) \\
\Rightarrow\ &\theta(\cup_i L_i) \cap \theta(K) = (\cup_i \theta(L_i)) \cap \theta(K) \\
\Rightarrow\ &\theta(\cup_i L_i) \cap \theta(K) = \cup_i (\theta(L_i) \cap \theta(K)) \\
\Rightarrow\ &\theta(\cup_i L_i) \cap \theta(K) \neq \emptyset
\end{aligned}$$

that is, the union $\cup_i L_i$ is also weakly opaque with respect to $K$. □

**Theorem 3.3.5.** *For any observation mapping $\theta$ and languages $L_i \subseteq \mathcal{L}(G), i = 1, 2, \cdots$, and $K \subseteq \mathcal{L}(G)$, if $L_i$ is not opaque with respect to $K$ and $\theta$, then the union $\cup_i L_i$ is also not opaque with respect to $K$.*

*Proof.* By the definition, $L_i$ is not opaque with respect to $K$ means

$$\theta(L_i) \cap \theta(K) = \emptyset \Rightarrow \cup_i (\theta(L_i) \cap \theta(K)) = \emptyset$$

From Lemma 3.3.2:

$$\begin{aligned}
&\theta(\cup_i L_i) = \cup_i \theta(L_i) \\
\Rightarrow\ &\theta(\cup_i L_i) \cap \theta(K) = \cup_i \theta(L_i) \cap \theta(K) \\
\Rightarrow\ &\theta(\cup_i L_i) \cap \theta(K) = \cup_i (\theta(L_i) \cap \theta(K)) \\
\Rightarrow\ &\theta(\cup_i L_i) \cap \theta(K) = \emptyset
\end{aligned}$$

that is, the union $\cup_i L_i$ is not opaque with respect to $K$. □

**Theorem 3.3.6.** *For any observation mapping $\theta$ and languages $L_i \subseteq \mathcal{L}(G), i = 1, 2, \cdots$, and $K \subseteq \mathcal{L}(G)$, if $L_i$ is not opaque with respect to $K$ and $\theta$, then the intersection $\cap_i L_i$ is also not opaque with respect to $K$.*

*Proof.* By the definition, $L_i$ is not opaque with respect to $K$ means

$$\begin{aligned}
&\theta(L_i) \cap \theta(K) = \emptyset \\
\Rightarrow\ &\cap_i (\theta(L_i) \cap \theta(K)) = \emptyset
\end{aligned}$$

From Lemma 3.3.1:

$$\begin{aligned}
&\theta(\cap_i L_i) \cap \theta(K) \subseteq \cap_i (\theta(L_i) \cap \theta(K)) \\
\Rightarrow\ &\theta(\cap_i L_i) \cap \theta(K) = \emptyset
\end{aligned}$$

that is, $\cap_i L_i$ is also not opaque with respect to $K$. □

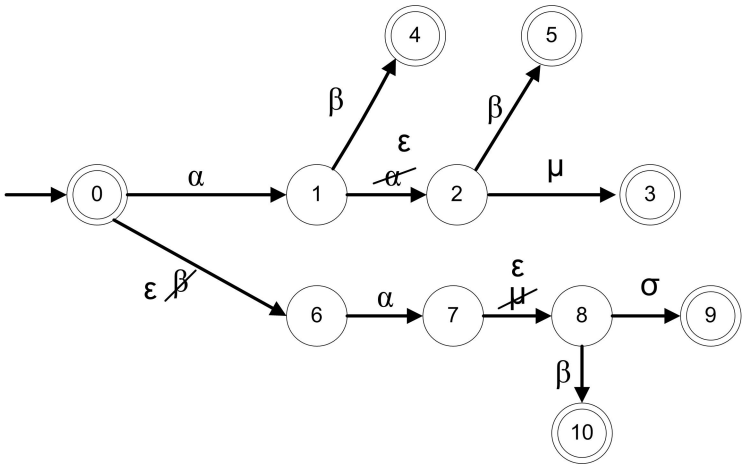



*Figure 3.5: System $G$ and observation mapping $\theta$ used in Remarks 3.3.1 and 3.3.2*

Generally speaking, opacity is closed under union but may not be closed under intersection. The following remarks provide counter-examples for cases where opacity is not closed under the intersection.

**Remark 3.3.1.** *For any observation mapping $\theta$ and languages $L, K_1, K_2 \subseteq \mathcal{L}(G)$, even if $L$ is strongly opaque with respect to $K_1$ and $\theta$; and $L$ is strongly opaque with respect to $K_2$ and $\theta$, $L$ might be not strongly opaque with respect to $(K_1 \cap K_2)$.*

*This can be seen from the following example. Consider the system in Figure 3.5, where $\Sigma = \{\alpha, \beta, \mu, \sigma\}$. Let $\theta$ be the observation mapping as defined in Figure 3.5, where the unobservable transitions are replaced by $\epsilon$.*

$$L = \alpha\beta$$
$$K_1 = \alpha\alpha(\beta + \mu)$$
$$K_2 = \beta\alpha\mu(\beta + \sigma)$$

*Clearly*

$$\theta(L) = \alpha\beta$$
$$\theta(K_1) = \alpha(\beta + \mu)$$
$$\theta(K_2) = \alpha(\beta + \sigma)$$

*By the definition $L$ is strongly opaque with respect to $K_1$ and $\theta$; and $L$ is strongly opaque with respect to $K_2$ and $\theta$, because*

$$\theta(L) \subseteq \theta(K_1)$$
$$\theta(L) \subseteq \theta(K_2)$$

*However, $L$ is not strongly opaque with respect to $(K_1 \cap K_2)$ and $\theta$ because*

$$(K_1 \cap K_2) = \emptyset$$
$$\Rightarrow \theta(K_1 \cap K_2) = \emptyset$$
$$\Rightarrow \theta(L) \nsubseteq \theta(K_1 \cap K_2).$$

**Remark 3.3.2.** *For any observation mapping $\theta$ and languages $L_1, L_2, K \subseteq \mathcal{L}(G)$, even if both $L_1$ and $L_2$ are weakly opaque with respect to $K$, $(L_1 \cap L_2)$ might be not weakly opaque with respect to $K$.*

*Consider the same system and observation mapping as in Remark 3.3.1. Let*

$$L_1 = \alpha\beta$$
$$L_2 = \beta\alpha\mu(\beta + \sigma)$$
$$K = \alpha\alpha(\beta + \mu)$$

*Clearly*

$$\theta(L_1) = \alpha\beta$$
$$\theta(L_2) = \alpha(\beta + \sigma)$$
$$\theta(K) = \alpha(\beta + \mu)$$

*By the definition, both $L_1$ and $L_2$ are weakly opaque with respect to $K$ and $\theta$, because*

$$\theta(L_1) \cap \theta(K) = \{\alpha\beta\} \neq \emptyset$$
$$\theta(L_2) \cap \theta(K) = \{\alpha\beta\} \neq \emptyset$$

*However $(L_1 \cap L_2)$ is not weakly opaque with respect to $K$ and $\theta$, because*

$$(L_1 \cap L_2) = \emptyset$$
$$\Rightarrow \theta(L_1 \cap L_2) = \emptyset$$
$$\Rightarrow \theta(L_1 \cap L_2) \cap \theta(K) = \emptyset.$$

# 4 Opaque Superlanguages and Sublanguages

In the previous chapter we show that opacity is closed under arbitrary union, but may not be closed under the intersection. Based on these properties, we discuss how to modify languages to satisfy the strong opacity, weak opacity, and no opacity by investigating the class of opaque sublanguage and the class of opaque superlanguage. In particular, we would like to find the unique smallest language as well as the unique largest language that satisfies the opacity properties. This would allow us to modify the language in order to satisfy opacity properties with a unique language. In this context, results presented in Chapter 3 are used to obtain general formulas for computing the opaque superlanguage and opaque sublanguage.

## 4.1 Preliminaries and Notations

The study of the "largest" and the "smallest" sublanguage of the specification languages in discrete event systems framework is very important. It is shown in the literature that the solutions for supervisory control problems are always characterized in terms of the largest sublanguage or the smallest superlanguage that meets the specifications. However, we do not just want to find any such language; the empty language, for instance, is always a solution, though not a very satisfactory one. What we want to find is the largest possible such language, if such a language exists.

The notion in finding the "largest" or the "smallest" possible objects in some set under a given ordering is phrased in the term of Zorn's Lemma. In this chapter we will state first some of set theory definitions as well as Zorn's lemma and use it later to prove some results in opacity control problems.

**Definition 4.1.1.** *A partial ordering on a non-empty set $\mathcal{M}$ is a binary relation on $\mathcal{M}$, denoted $\leq$, which satisfies following properties:*

- *For all* $m \in \mathcal{M}, m \leq m$
- *if* $m \leq \acute{m}$ *and* $\acute{m} \leq m$ *then* $m = \acute{m}$
- *if* $m \leq \acute{m}$ *and* $\acute{m} \leq t$ *then* $m \leq t$

**Definition 4.1.2.** *Let* $\mathcal{M}$ *be a non-empty partially ordered set. A non-empty subset* $R \subseteq \mathcal{M}$ *is said to be a totally ordered subset with respect to* $\leq$*, if*

$$(\forall r, t \in R)\ \ r \leq t \ \text{ or } t \leq r$$

**Definition 4.1.3.** *For any subset* $R \subseteq \mathcal{M}$*, an element* $m \in \mathcal{M}$ *is the upper bound of* $R$*, if*

$$(\forall r \in R)\ \ r \leq m$$

A *maximal element* $m$ of a partially ordered set $\mathcal{R}$ is an element which is not below any element to which it is comparable, that is, for all $r \in \mathcal{R}$ to which m is comparable, $r \leq m$. Equivalently, m is maximal element when the only $r \in \mathcal{R}$ satisfying $m \leq r$ is $r = m$. This does not mean $r \leq m$ for all $r$ in $\mathcal{R}$ since we don't insist that maximal elements are actually comparable to every element of $\mathcal{R}$. A partially ordered set could have many incomparable maximal elements.

A *minimal element* $m$ of a partially ordered set $\mathcal{R}$ is an element which is below any element to which it is comparable, that is, $m \in \mathcal{R}$ is a minimal if $m \leq r$ for all $r \in \mathcal{R}$ to which m is comparable.

**Theorem 4.1.1.** (Zorn's lemma) *Let* $R$ *be a partially ordered set. If every totally ordered subset of* $R$ *has an upper bound, then* $R$ *contains a maximal element.*

Zorn's lemma can be stated in terms of minimal elements: if any totally ordered subset of a partially ordering set $\mathcal{R}$ has a lower bound in $\mathcal{R}$, then $\mathcal{R}$ has a minimal element.

## 4.2 Opaque Superlanguages and Sublanguages: Existence and Properties

If a language is not opaque with respect to another language, then we must modify one of the two languages to satisfy opacity. There are several ways to do modifications. We can reduce or shrink a language, that is, consider its sublanguage, or we can enlarge a language, that is, consider its superlanguage. We will consider all possibilities for strong opacity, weak opacity, and no opacity.

### 4.2.1 Strong opacity

In this section we discuss how to modify languages in order to satisfy strong opacity. If a language $L \subseteq \mathcal{L}(G)$ is not strongly opaque with respect to $K \subseteq \mathcal{L}(G)$, that is, $\theta(L) \nsubseteq \theta(K)$, then we may want to modify $L$ or $K$ in order to satisfy strong opacity. There are two ways to do the modification: (1) Reduce $L$ and (2) Enlarge $K$.

To reduce L, we want to find the largest sublanguage of $L$ that is strongly opaque with respect to $K$. Using Theorem 3.3.2, we can show that this sublanguage exists and is unique. To see this, let us define the set of sublanguages of $L$ that are strongly opaque with respect to $K$ as

$$S_K^{sub}(L) = \{L_i \subseteq \mathcal{L}(G) : L_i \subseteq L \wedge \theta(L_i) \subseteq \theta(K)\}$$

By Theorem 3.3.2, if

$$L_i \in S_K^{sub}(L), i = 1, 2, \cdots$$

then

$$\bigcup_i L_i \in S_K^{sub}(L)$$

Therefore, the supremal element of $S_K^{sub}(L)$ exists and is given by

$$supS_K^{sub}(L) = \bigcup_{L_i \in S_K^{sub}(L)} L_i$$

To enlarge K, we want to find the smallest superlanguage of $K$, so that $L$ is strongly

opaque with respect to the superlanguage. Using Remark 3.3.1, we can show that this superlanguage might not be unique. To see this, let us define the set of superlanguages of $K$ such that $L$ is strongly opaque with respect to them as

$$S_L^{super}(K) = \{K_i \subseteq \mathcal{L}(G) : K \subseteq K_i \wedge \theta(L) \subseteq \theta(K_i)\}$$

By Remark 3.3.1.

$$\begin{aligned} & K_1 \in S_L^{super}(K) \wedge K_2 \in S_L^{super}(K) \\ \nRightarrow\ & (K_1 \cap K_2) \in S_L^{super}(K) \end{aligned}$$

Therefore it is not difficult to show that there may not exist a unique smallest language in $S_L^{super}(K)$. In other words, the infimal element $infS_L^{super}(K)$ may not exist. However, we can always find some minimal elements in $S_L^{super}(K)$, which will be denoted by $minS_L^{super}(K)$. To illustrate this we use the following Example

**Example 4.2.1.** *Consider the system in Figure 4.1, where $\Sigma = \{\alpha, \beta, \gamma\}$. Let the observation mapping $\theta$ be defined as illustrated in the figure, where the unobservable transitions are replaced by $\epsilon$. Let*

$$L = \alpha\beta\alpha\gamma$$

$$K = \alpha\gamma$$

*Clearly*

$$\theta(L) = \alpha\beta\gamma$$

$$\theta(K) = \alpha$$

*$L$ is not strongly opaque with respect to $K$, because*

$$\theta(L) \nsubseteq \theta(K)$$

*There are two ways to enlarge $K$:*

$$K_1 = \alpha\gamma + \alpha\beta\alpha\gamma$$

$$K_2 = \alpha\gamma + \alpha\beta\gamma\beta$$

*Clearly*

$$\theta(K_1) = \theta(K_2) = \alpha + \alpha\beta\gamma$$

*$L$ is strongly opaque with respect to $K_1$ and $K_2$, because*

$$\theta(L) \subseteq \theta(K_1) = \theta(K_2)$$

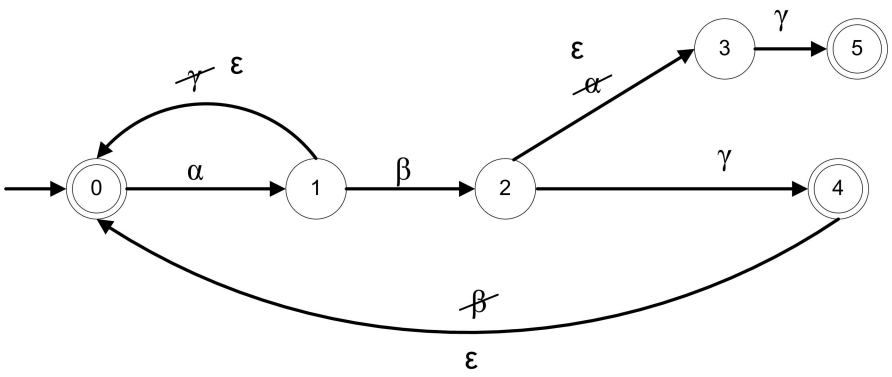



*Figure 4.1: System $G$ and observation mapping $\theta$ used in Example 4.2.1*

*Thus, it is not difficult to see that both $K_1$ and $K_2$ are minimal elements in $S_L^{super}(K)$. That is:*

$$K_1 = minS_L^{super}(K)$$
$$K_2 = minS_L^{super}(K)$$

*It is also not difficult to see that the infimal element $infS_L^{super}(K)$ does not exist. In particular,*

$$\theta(K_1 \cap K_2) = \alpha + \alpha\beta$$
$$\Rightarrow \theta(L) \not\subseteq \theta(K_1 \cap K_2)$$
$$\Rightarrow K_1 \cap K_2 \notin S_L^{super}(K)$$

### 4.2.2 Weak opacity

If $L \subseteq \mathcal{L}(G)$ is not weakly opaque with respect to $K \subseteq \mathcal{L}(G)$, that is, $\theta(L) \cap \theta(K) = \emptyset$, then we may want to enlarge $L$ or $K$ in order to satisfy weak opacity. Since $L$ and $K$ are symmetric, we discuss only how to enlarge $L$. Denote the set of superlanguages of $L$ that are weakly opaque with respect to $K$ as

$$W_K^{super}(L) = \{L_i \subseteq \mathcal{L}(G) : L \subseteq L_i \wedge \theta(L_i) \cap \theta(K) \neq \emptyset\}$$

By Remark 3.3.2.

$$L_1 \in W_K^{super}(L) \wedge L_2 \in W_K^{super}(L)$$
$$\not\Rightarrow (L_1 \cap L_2) \in W_K^{super}(L)$$

Therefore, it is not difficult to show that there may not exist a unique smallest language in $W_K^{super}(L)$. In other words, the infimal element $infW_K^{super}(L)$ may not exist. However, we can always find some minimal elements in $W_K^{super}(L)$, which will be denoted by $minW_K^{super}(L)$.

Again, we illustrate this using the following Example

**Example 4.2.2.** *Consider the system in Figure 4.2, where* $\Sigma = \{\alpha, \beta, \gamma, \sigma\}$. *Let the observation mapping* $\theta$ *be defined as illustrated in the figure, where the unobservable transitions are replaced by* $\epsilon$. *Let*

$$K = \beta\alpha\beta\gamma$$
$$L = \alpha\sigma\beta\sigma$$

*Clearly*

$$\theta(K) = \alpha\beta\gamma$$
$$\theta(L) = \alpha\beta\sigma$$

$K$ *is not weakly opaque with respect to* $L$ *because*

$$\theta(K) \cap \theta(L) = \emptyset$$

*There are two ways to enlarge* $L$*:*

$$L_1 = \alpha\sigma\beta\sigma + \alpha\sigma\beta\alpha\gamma$$
$$L_2 = \alpha\sigma\beta\sigma + \alpha\sigma\beta\gamma$$

*Clearly*

$$\theta(L_1) = \theta(L_2) = \alpha\beta\sigma + \alpha\beta\gamma$$

$K$ *is weakly opaque with respect to* $L_1$ *and* $L_2$*, because*

$$\theta(L_1) \cap \theta(K) = \theta(L_2) \cap \theta(K) \neq \emptyset$$

*It is not difficult to see that both* $L_1$ *and* $L_2$ *are minimal elements in* $W_K^{super}(L)$*:*

$$L_1 = minW_K^{super}(L)$$
$$L_2 = minW_K^{super}(L)$$

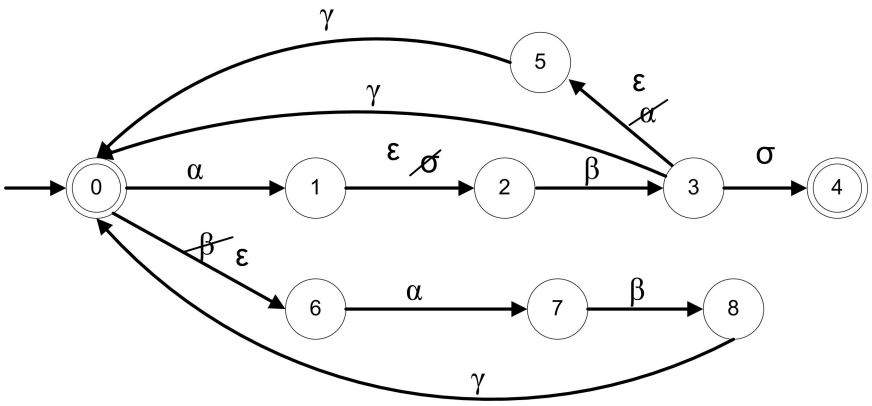



*Figure 4.2: System $G$ and observation mapping $\theta$ used in Example 4.2.2*

*It is also not difficult to see that the infimal element* $infW_K^{super}(L)$ *does not exist. In particular,*

$$\begin{aligned} &\theta(L_1 \cap L_2) = \alpha\beta\sigma \\ \Rightarrow\ &\theta(L_1 \cap L_2) \cap \theta(K) = \emptyset \\ \Rightarrow\ &L_1 \cap L_2 \notin W_K^{super}(L) \end{aligned}$$

### 4.2.3 No opacity

For $L, K \subseteq \mathcal{L}(G)$, $L$ is not opaque with respect to $\theta$ and $K$ if $\theta(L) \cap \theta(K) = \emptyset$. If this is not true, that is $\theta(L) \cap \theta(K) \neq \emptyset$, then we may want to shrink $L$ or $K$ in order to satisfy no opacity. Since $L$ and $K$ are symmetric, we discuss only how to shrink $L$.

To shrink $L$, we want to find the largest sublanguage of $L$ that is not opaque with respect to $K$. By Theorem 3.3.5, we can show that this sublanguage exists and is unique. To see this, let us define the set of sublanguage of $L$ that is not opaque with respect to $K$ as

$$N_K^{sub}(L) = \{L_i \subseteq \mathcal{L}(G) : L_i \subseteq L \wedge \theta(L_i) \cap \theta(K) = \emptyset\}$$

By Theorem 3.3.5, if

$$L_i \in N_K^{sub}(L), i = 1, 2, \cdots$$

then

$$\bigcup_i L_i \in N_K^{sub}(L)$$

Therefore, the supremal element of $N_K^{sub}(L)$ exists and is given by

$$supN_K^{sub}(L) = \bigcup_{L_i \in N_K^{sub}(L)} L_i$$

## 4.3 Formulas for Calculating Opaque Superlanguages and Sublanguages

In this section, we will derive formulas for (1) supremal strong opaque sublanguages, (2) minimal strong opaque superlanguages, (3) minimal weak opaque superlanguages, and (4) supremal not opaque sublanguages. These formulas can be used to calculate various opaque superlanguages and sublanguages so that system behaviors can be properly modified to satisfy opacity properties.

### 4.3.1 Strong Opacity

**Supremal strong opaque sublanguage** $supS_K^{sub}(L)$

As discussed in the previous section, the supremal strong opaque sublanguages $supS_K^{sub}(L)$ is given as the union of all sublanguages $L_i$ that are strong opaque with respect to $K$. To actually calculate $supS_K^{sub}(L)$, we will derive a formula for it. Since a language $L \subseteq \mathcal{L}(G)$ is not strongly opaque with respect to $K \subseteq \mathcal{L}(G)$ , that is, $\theta(L) \not\subseteq \theta(K)$, then $\theta(L) - \theta(K) \neq \emptyset$. Intuitively $\theta(L) - \theta(K)$ are bad strings and $\theta^{-1}(\theta(L) - \theta(K))$ are the strings that need to be removed from $L$ in order to get the largest strong opaque sublanguage of $L$. Thus, the formula for $supS_K^{sub}(L)$ is presented in the following theorem.

**Theorem 4.3.1.**

$$supS_K^{sub}(L) = L - \theta^{-1}(\theta(L) - \theta(K))$$

*Proof.* $(\subseteq)$: First we show that

$$supS_K^{sub}(L) \subseteq L - \theta^{-1}(\theta(L) - \theta(K))$$

Since $supS_K^{sub}(L) \subseteq L$, we only need to show that

$$supS_K^{sub}(L) \cap \theta^{-1}(\theta(L) - \theta(K)) = \emptyset$$

Assume the contrary

$$supS_K^{sub}(L) \cap \theta^{-1}(\theta(L) - \theta(K)) \neq \emptyset$$

Then

$$\begin{aligned}
&(\exists s \in \Sigma^*) s \in \theta^{-1}(\theta(L) - \theta(K)) \wedge s \in supS_K^{sub}(L) \\
\Rightarrow &(\exists s \in \Sigma^*) \theta(s) \in (\theta(L) - \theta(K)) \wedge s \in supS_K^{sub}(L) \\
\Rightarrow &(\exists s \in \Sigma^*) \theta(s) \in \theta(L) \wedge \theta(s) \notin \theta(K) \wedge s \in supS_K^{sub}(L)
\end{aligned}$$

Since $supS_K^{sub}(L)$ is strongly opaque, that is, $\theta(supS_K^{sub}(L)) \subseteq \theta(K)$, we have

$$\begin{aligned}
&\theta(s) \in \theta(supS_K^{sub}(L)) \\
\Rightarrow &\theta(s) \in \theta(K)
\end{aligned}$$

This leads to a contradiction:

$$\theta(s) \in \theta(K) \text{ and } \theta(s) \notin \theta(K).$$

$(\supseteq)$: Next we show that

$$supS_K^{sub}(L) \supseteq L - \theta^{-1}(\theta(L) - \theta(K))$$

By the definition of $supS_K^{sub}(L)$, we need to show the following.

1. $L - \theta^{-1}(\theta(L) - \theta(K)) \subseteq L$
2. $L - \theta^{-1}(\theta(L) - \theta(K))$ is strongly opaque w.r.t $K$.

Condition 1 is obviously true. To show that condition 2 is also true, we need to prove

$$\theta(L - \theta^{-1}(\theta(L) - \theta(K))) \subseteq \theta(K)$$

This can be done as follows.

$$\begin{aligned}
&s \in \theta(L - \theta^{-1}(\theta(L) - \theta(K))) \\
\Rightarrow &(\exists t \in \Sigma^*) \theta(t) = s \wedge t \in (L - \theta^{-1}(\theta(L) - \theta(K)) \\
\Rightarrow &(\exists t \in \Sigma^*) \theta(t) = s \wedge t \in L \wedge t \notin \theta^{-1}(\theta(L) - \theta(K)) \\
\Rightarrow &(\exists t \in \Sigma^*) \theta(t) = s \wedge t \in L \wedge \theta(t) \notin (\theta(L) - \theta(K))
\end{aligned}$$

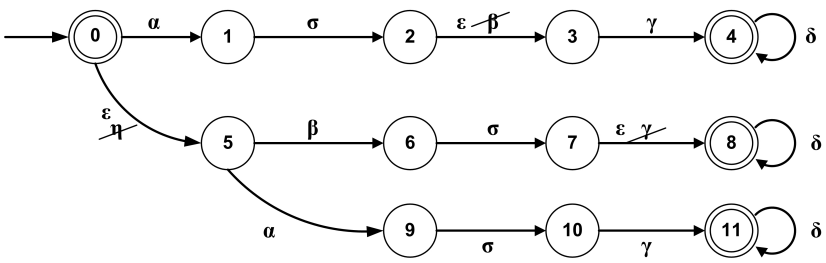



*Figure 4.3: System $G$ and observation mapping $\theta$ used in Example 4.3.1*

$$\Rightarrow (\exists t \in \Sigma^*)\theta(t) = s \wedge t \in L \wedge (\theta(t) \notin \theta(L) \vee \theta(t) \in \theta(K)$$
$$\Rightarrow (\exists t \in \Sigma^*)\theta(t) = s \wedge t \in L \wedge \theta(t) \in \theta(K)$$

Therefore, $s \in \theta(K)$ □

**Example 4.3.1.** *Consider the system in Figure 4.3, where* $\Sigma = \{\alpha, \beta, \gamma, \sigma, \eta, \delta\}$. *Let the observation mapping* $\theta$ *be defined as illustrated in the Figure 4.3, where the unobservable transitions are replaced by* $\epsilon$. *Let*

$$L = \alpha\sigma\beta\gamma\delta^* + \eta\beta\sigma\gamma\delta^* + \eta\alpha\sigma\gamma\delta^*$$
$$K = \alpha\sigma\beta\gamma\delta^*$$

*Clearly*

$$\theta(L) = \alpha\sigma\gamma\delta^* + \beta\sigma\delta^*$$
$$\theta(K) = \alpha\sigma\gamma\delta^*$$

$L$ *is not strongly opaque with respect to* $K$ *because*

$$\theta(L) \nsubseteq \theta(K)$$
$$\theta(L) - \theta(K) = \beta\sigma\delta^*$$
$$\theta^{-1}(\theta(L) - \theta(K)) = \eta\beta\sigma\gamma\delta^*$$

*Therefore*

$$supS_K^{sub}(L) = L - \eta\beta\sigma\gamma\delta^*$$
$$supS_K^{sub}(L) = \alpha\sigma\beta\gamma\delta^* + \eta\alpha\sigma\gamma\delta^*$$

**Minimal strong opaque superlanguage** $minS_L^{super}(K)$

In the previous section we show that the infimal strong opaque superlanguage $infS_L^{super}(K)$ may not exist, but we can always find some minimal strong opaque superlanguage $minS_L^{super}(K)$. To this end, assume that a language $L \subseteq \mathcal{L}(G)$ is not strongly opaque with respect to $K \subseteq \mathcal{L}(G)$ , that is, $\theta(L) \nsubseteq \theta(K)$. We want to enlarge $K$ to satisfy strong opacity. Intuitively, strings that look like $\theta(L) - \theta(K)$ need to be added to $K$ in order to ensure strong opacity. However, adding $\theta^{-1}(\theta(L) - \theta(K))$ to $K$ will enlarge $K$ more than the minimal that is needed. In order to add only the minimal, for any $t \in \theta(L) - \theta(K)$, pick one $s \in \mathcal{L}(G)$ such that $\theta(s) = t$ and denote it by $s = \hat{\theta}^{-1}(t)$. Define

$$\hat{\theta}^{-1}(\theta(L) - \theta(K)) = \bigcup_{t \in (\theta(L) - \theta(K))} \hat{\theta}^{-1}(t)$$

Using this notation, the formula for $minS_L^{super}(K)$ is presented in the following theorem.

**Theorem 4.3.2.**

$$minS_L^{super}(K) = K \cup \hat{\theta}^{-1}(\theta(L) - \theta(K))$$

*Proof.* We need to show the following

1. $K \cup \hat{\theta}^{-1}(\theta(L) - \theta(K))$ is a superlanguage of $K$
2. $L$ is strongly opaque with respect to $K \cup \hat{\theta}^{-1}(\theta(L) - \theta(K))$.
3. Remove any string in $K \cup \hat{\theta}^{-1}(\theta(L) - \theta(K))$ will violate either (1) or (2).

Condition 1 is clearly true. Condition 2 is true because

$$\begin{aligned} & \theta(K \cup \hat{\theta}^{-1}(\theta(L) - \theta(K))) \\ = & \theta(K) \cup (\theta(L) - \theta(K)) \\ = & \theta(K) \cup \theta(L) \\ \supseteq & \theta(L) \end{aligned}$$

Condition 3 is also true because (1) if a string in $K$ is removed, then the resulting language

is not a superlanguage of $K$, and (2) if a string in $\hat{\theta}^{-1}(\theta(L)-\theta(K))$ is removed, then strong opacity is violated. To see (2), suppose $s_1 = \hat{\theta}^{-1}(t_1)$ where $t_1 \in (\theta(L)-\theta(K)$ is removed then

$$\begin{aligned}
&\theta(K \cup \hat{\theta}^{-1}(\theta(L)-\theta(K)) - \{s_1\}) \\
&= \theta(K \cup \hat{\theta}^{-1}(\theta(L)-\theta(K))) - \{t_1\} \\
&= \theta(K) \cup (\theta(L)-\theta(K)) - \{t_1\} \\
&= \theta(K) \cup \theta(L) - \{t_1\}
\end{aligned}$$

Since $t_1 \in (\theta(L)-\theta(K))$

$$\theta(K) \cup \theta(L) - \{t_1\} \not\supseteq \theta(L),$$

that is, strong opacity is violated □

**Example 4.3.2.** *Consider the system in Figure 4.1*

$$L = \alpha\beta\alpha\gamma$$
$$K = \alpha\gamma$$

*Clearly*

$$\theta(L) = \alpha\beta\gamma$$
$$\theta(K) = \alpha$$
$$(\theta(L)-\theta(K)) = \alpha\beta\gamma$$

*Pick* $\hat{\theta}^{-1}(\alpha\beta\gamma) = \alpha\beta\gamma\beta$*, then*

$$\hat{\theta}^{-1}(\theta(L)-\theta(K)) = \alpha\beta\gamma\beta$$

*Therefore,*

$$\begin{aligned}
minS_L^{super}(K) &= K \cup \hat{\theta}^{-1}(\theta(L)-\theta(K)) \\
&= \alpha\gamma + \alpha\beta\gamma\beta.
\end{aligned}$$

*Another choice is* $\hat{\theta}^{-1}(\alpha\beta\gamma) = \alpha\beta\alpha\gamma$*. For this choice,*

$$\begin{aligned}
minS_L^{super}(K) &= K \cup \hat{\theta}^{-1}(\theta(L)-\theta(K)) \\
&= \alpha\gamma + \alpha\beta\alpha\gamma.
\end{aligned}$$

### 4.3.2 Weak Opacity

**Minimal weak opaque superlanguage** $minW_K^{super}(L)$

We know that the infimal weak opaque superlanguage $infW_K^{super}(L)$ may not exist, but we can always find some minimal weak opaque superlanguage $minW_K^{super}(L)$. To this end, if $\theta(L) \cap \theta(K) = \emptyset$, we will enlarge $L$, so that $\theta(L) \cap \theta(K) \neq \emptyset$. This can be done by adding any string in $\theta^{-1}\theta(K)$. Thus, the formula for $minW_K^{super}(L)$ is presented in the following theorem.

**Theorem 4.3.3.** *If* $\theta(L) \cap \theta(K) = \emptyset$, *then*

$$minW_K^{super}(L) = L \cup \{s\},$$

*where* $s$ *is any string such that* $s \in \theta^{-1}\theta(K)$.

*Proof.* To prove the theorem, we need to show the following

1. $L \cup \{s\}$ is a superlanguage of $L$
2. $L \cup \{s\}$ is weakly opaque with respect to $K$.
3. Remove any string in $L \cup \{s\}$ will violate either (1) or (2).

Condition 1 is clearly true. Condition 2 is true because

$$\begin{aligned} & \theta(L \cup \{s\}) \cap \theta(K) \\ = & (\theta(L) \cup \{\theta(s)\}) \cap \theta(K) \\ = & (\theta(L) \cap \theta(K)) \cup (\{\theta(s)\} \cap \theta(K)) \\ = & \{\theta(s)\} \cap \theta(K) \\ = & \{\theta(s)\} \neq \emptyset \end{aligned}$$

Condition 3 is also true because (1) if a string in $L$ is removed, then the resulting language is not a superlanguage of $L$, and (2) if $s$ is removed, then the weak opacity is violated, because $\theta(L) \cap \theta(K) = \emptyset$ □

**Example 4.3.3.** *Consider the system in Figure 4.2. Let*

$$K = \beta\alpha\beta\gamma + \alpha\sigma\beta\gamma$$

$$L = \alpha\sigma\beta\sigma$$

*Clearly*

$$\theta(K) = \alpha\beta\gamma$$

$$\theta(L) = \alpha\beta\sigma$$

$$\theta(L) \cap \theta(K) = \emptyset$$

*Let us enlarge $L$ by picking $s = \beta\alpha\beta\gamma \in \theta^{-1}\theta(K)$. Therefore,*

$$minW_K^{super}(L) = L \cup \{s\} = \alpha\sigma\beta\sigma + \beta\alpha\beta\gamma$$

### 4.3.3 No Opacity

**Supremal not opaque sublanguage** $supN_K^{sub}(L)$

To derive a formula for the supremal not opaque sublanguages $supN_K^{sub}(L)$, we investigate how to shrink $L$. Intuitively, $\theta(L) \cap \theta(K)$ are bad strings and $\theta^{-1}(\theta(L) \cap \theta(K))$ are the strings that wil be observed as $\theta(L) \cap \theta(K)$. Therefore, they have to be removed from $L$. Thus, the formula for $supN_K^{sub}(L)$ is presented in the following theorem.

**Theorem 4.3.4.**

$$supN_K^{sub}(L) = L - \theta^{-1}\left(\theta(L) \cap \theta(K)\right)$$

*Proof.* ($\subseteq$): First we show that

$$supN_K^{sub}(L) \subseteq L - \theta^{-1}\left(\theta(L) \cap \theta(K)\right)$$

Since $supN_K^{sub}(L) \subseteq L$, we only need to show that

$$supN_K^{sub}(L) \cap \theta^{-1}\left(\theta(L) \cap \theta(K)\right) = \emptyset$$

Assume the contrary. Then

$$
\begin{aligned}
&(\exists s \in \Sigma^*) s \in \theta^{-1}(\theta(L) \cap \theta(K)) \wedge s \in supN_K^{sub}(L) \\
&\Rightarrow (\exists s \in \Sigma^*) \theta(s) \in (\theta(L) \cap \theta(K)) \wedge \theta(s) \in \theta(supN_K^{sub}(L)) \\
&\Rightarrow \theta(L) \cap \theta(K) \cap \theta(supN_K^{sub}(L)) \neq \emptyset \\
&\Rightarrow \theta(K) \cap \theta(supN_K^{sub}(L)) \neq \emptyset
\end{aligned}
$$

This contradicts the fact that $supN_K^{sub}(L)$ is not opaque with respect to $K$.

($\supseteq$): Next we show that

$$supN_K^{sub}(L) \supseteq L - \theta^{-1}(\theta(L) \cap \theta(K))$$

This requires us to prove the following

1. $L - \theta^{-1}(\theta(L) \cap \theta(K)) \subseteq L$
2. $L - \theta^{-1}(\theta(L) \cap \theta(K))$ is not opaque w.r.t $K$.

Condition 1 is obviously true. To prove that condition 2 is also true, we need to prove

$$\theta(L - \theta^{-1}(\theta(L) \cap \theta(K))) \cap \theta(K) = \emptyset$$

This can be done as follows. Assume the contrary. Then there exist $s \in \Sigma^*$,

$$
\begin{aligned}
&s \in \theta(L - \theta^{-1}(\theta(L) \cap \theta(K))) \wedge s \in \theta(K) \\
&\Rightarrow (\exists t \in \Sigma^*) \theta(t) = s \wedge t \in (L - \theta^{-1}(\theta(L) \cap \theta(K)) \wedge s \in \theta(K) \\
&\Rightarrow (\exists t \in \Sigma^*) \theta(t) = s \wedge t \in L \wedge t \notin \theta^{-1}(\theta(L) \cap \theta(K)) \wedge s \in \theta(K) \\
&\Rightarrow (\exists t \in \Sigma^*) \theta(t) = s \wedge \theta(t) \in \theta(L) \wedge \theta(t) \notin (\theta(L) \cap \theta(K)) \wedge s \in \theta(K) \\
&\Rightarrow (\exists t \in \Sigma^*) \theta(t) = s \wedge s \in \theta(L) \wedge s \notin (\theta(L) \cap \theta(K)) \wedge s \in \theta(K)
\end{aligned}
$$

This leads to a contradiction:

$$(s \in \theta(L) \wedge s \notin (\theta(L) \cap \theta(K)) \wedge s \in \theta(K))$$

□

**Example 4.3.4.** *Consider the system in Figure 4.3.*

$$
\begin{aligned}
L &= \alpha\sigma\beta\gamma\delta^* + \eta\beta\sigma\gamma\delta^* + \eta\alpha\sigma\gamma\delta^* \\
K &= \alpha\sigma\beta\gamma\delta^*
\end{aligned}
$$

*Clearly*

$$\theta(L) = \alpha\sigma\gamma\delta^* + \beta\sigma\delta^*$$
$$\theta(K) = \alpha\sigma\gamma\delta^*$$
$$(\theta(L) \cap \theta(K)) = \alpha\sigma\gamma\delta^*$$
$$\theta^{-1}(\theta(L) \cap \theta(K)) = \alpha\sigma\beta\gamma\delta^* + \eta\alpha\sigma\gamma\delta^*$$

*Therefore,*

$$\begin{aligned} supN_K^{sub}(L) &= L - \theta^{-1}\left(\theta(L) \cap \theta(K)\right) \\ &= \eta\beta\sigma\gamma\delta^*. \end{aligned}$$

# 5 Opacity of Discrete Event Systems in a Decentralized Framework

Investigating opacity in the decentralized structure lays the foundation to solve important problems in advanced networking technologies. These problems are related to security and privacy of computer systems, communication protocols, and distributed systems that are used in a decentralized manner by several users (observers) located at different nodes. Each has partial information about the system evolution.

This chapter presents an extended version of the formulas presented in the pervious chapter. We present formulas for calculating the sublanguages and superlanguages in a decentralized framework. Our investigation is based on the approach presented in [42]. We consider a system observed by many agents, who observe the system behaviour using their own observation mapping. We consider the system described in Figure 5.1. The agents do not communicate each other and no coordination between them exists.

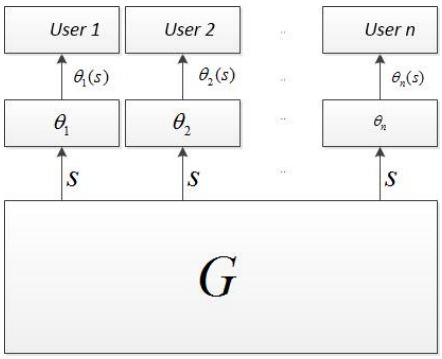



Figure 5.1: Framework of the decentralized opacity

## 5.1 Definitions of Decentralized Opacity

Through the observation mapping set $\Theta = \{\theta_1, \cdots, \theta_n\}$ every agent can only communicate with $G$, so no communication or any kind of coordination exists between agents. With this configuration the definitions of the decentralized opacity are extended from the centralized opacity as follows.

**Definition 5.1.1.** ***(Decentralized strong opacity)***
*Given two languages $L, K \subseteq \mathcal{L}(G)$, $L$ is strongly decentralized opaque with respect to $K$ and $\Theta$ if*

$$(\forall j \in J) L \subseteq \theta_j^{-1}\theta_j(K).$$

**Definition 5.1.2.** ***(Decentralized weak opacity)***
*Given two languages $L, K \subseteq \mathcal{L}(G)$, $L$ is weakly decentralized opaque with respect to $K$ and $\Theta$ if*

$$(\forall j \in J) L \cap \theta_j^{-1}\theta_j(K) \neq \emptyset.$$

**Definition 5.1.3.** ***(Decentralized no opacity)***
*Given two languages $L, K \subseteq \mathcal{L}(G)$, $L$ is not decentralized opaque with respect to $K$ and $\Theta$ if*

$$(\exists j \in J) L \cap \theta_j^{-1}\theta_j(K) = \emptyset.$$

Based on the above definitions, we present the following algorithms to check decentralized strong opacity and decentralized weak opacity.

## 5.2 Checking Decentralized Opacity

Similarly to checking the centralized opacity, decentralized opacity can be also checked using Theorems given in the Section 3.2.1. In order to check decentralized opacity, we need to check opacity for each agent $j \in J$ where $J = \{1, 2, \cdots, n\}$. This can be done as follow: For the observation mapping set $\Theta = \{\theta_1, \cdots, \theta_n\}$ and languages $L, K \subseteq \mathcal{L}(G)$

$$(\forall j \in J) L \subseteq \theta_j^{-1}\theta_j(K).$$

$$\Leftrightarrow (\forall j \in J)\theta_j(L) \subseteq \theta_j(K)$$

With this Theorem, is it possible to check the strong opacity by checking $\forall j \in J$ whether $\theta_j(L) \subseteq \theta_j(K)$. Similarly, for any observation mapping $\Theta = \{\theta_1, \cdots, \theta_n\}$ and languages $L, K \subseteq \mathcal{L}(G)$,

$$(\forall j \in J)L \cap \theta_j^{-1}\theta_j(K) = \emptyset$$

$$\Leftrightarrow (\forall j \in J)\theta_j(L) \cap \theta_j(K) = \emptyset.$$

Therefore, checking weak opacity can be done by checking $\forall j \in J$ whether $\theta_j(L) \cap \theta_j(K) \neq \emptyset$. It follows that checking no opacity can be done by checking $\forall j \in J$ whether $\theta_j(L) \cap \theta_j(K) = \emptyset$.

With these results, we can investigate the properties of opacity in the decentralized framwork and find decentralized opaque superlanguages and sublanguages.

## 5.3 Decentralized Opaque Superlanguages and Sublanguages

The goal of this section is to present the definitions of superlanguages and sublanguages in the decentralized framework. Similar to centralized opacity the modifications of the languages in the decentralized framework will be considered based on the properties of decentralized opacity and the observation mapping set $\Theta$. The properties of decentralized opacity are similar to the properties of centralized opacity. Essentially, if we study the opacity for each agent, then we can derive the following corollary from Theorem 3.3.1:

**Corollary 5.3.1.** *For any observation mapping set $\Theta = \{\theta_1, \cdots, \theta_n\}$ and languages $L_i \subseteq \mathcal{L}(G)$,$i = 1, 2, \cdots$, and $K \subseteq \mathcal{L}(G)$, if $(\forall j \in J)L_i$ is strongly opaque with respect to $K$, then the intersection $\cap_i L_i$ is also strongly opaque with respect to $K$.*

Similarly, from Theorem 3.3.2, Theorem 3.3.3, Theorem 3.3.4, Theorem 3.3.5, and Theorem 3.3.6 we have the following corollaries respectively:

**Corollary 5.3.2.** *For any observation mapping* $\Theta = \{\theta_1, \cdots, \theta_n\}$ *and languages* $L_i \subseteq \mathcal{L}(G)$,$i = 1,2,\cdots$, *and* $K \subseteq \mathcal{L}(G)$, *if* $(\forall j \in J)L_i$ *is strongly opaque with respect to* $K$, *the union* $\cup_i L_i$ *is also strongly opaque with respect to* $K$.

**Corollary 5.3.3.** *For any observation mapping* $\Theta = \{\theta_1, \cdots, \theta_n\}$ *and languages* $L \subseteq \mathcal{L}(G)$, *and* $K_i \subseteq \mathcal{L}(G)$,$i = 1,2,\cdots$, *if* $L$ *is strongly opaque with respect to* $(\forall j \in J)K_i$ *and* $\theta$, *then* $L$ *is also strongly opaque with respect to* $\cup_i K_i$.

**Corollary 5.3.4.** *For any observation mapping* $\Theta = \{\theta_1, \cdots, \theta_n\}$ *and languages* $L_i \subseteq \mathcal{L}(G)$,$i = 1,2,\cdots$, *and* $K \subseteq \mathcal{L}(G)$, *if* $(\forall j \in J)L_i$ *is weakly opaque with respect to* $K$ *and* $\theta$, *then the union* $\cup_i L_i$ *is also weakly opaque with respect to* $K$.

**Corollary 5.3.5.** *For any observation mapping* $\Theta = \{\theta_1, \cdots, \theta_n\}$ *and languages* $L_i \subseteq \mathcal{L}(G), i = 1,2,\cdots$, *and* $K \subseteq \mathcal{L}(G)$, *if* $(\forall j \in J)L_i$ *is not opaque with respect to* $K$ *and* $\Theta = \{\theta_1, \cdots, \theta_n\}$, *then the union* $\cup_i L_i$ *is also not opaque with respect to* $K$.

**Corollary 5.3.6.** *For any observation mapping* $\theta$ *and languages* $L_i \subseteq \mathcal{L}(G), i = 1,2,\cdots$, *and* $K \subseteq \mathcal{L}(G)$, *if* $(\forall j \in J)L_i$ *is not opaque with respect to* $K$ *and* $\Theta = \{\theta_1, \cdots, \theta_n\}$, *then the intersection* $\cap_i L_i$ *is also not opaque with respect to* $K$.

Clearly from the above corollaries we conclude that decentralized opacity is also closed under union, but may not be closed under the intersection.

With these notations, we define the decentralized opaque suprelanguages and sublanguages based on the various sets of superlanguages and sublanguages presented in Chapter 4.

### Strong opacity

For any observation mapping $\Theta = \{\theta_1, \cdots, \theta_n\}$ and languages $L, K \subseteq \mathcal{L}(G)$, if $L \subseteq \mathcal{L}(G)$ is not decentralized strongly opaque with respect to $K \subseteq \mathcal{L}(G)$, that is, $(\forall j \in J)\theta_j(L) \nsubseteq \theta_j(K)$, then we may want to modify $L$ or $K$ in order to satisfy decentralized strong opacity. There are two ways to do the modification: (1) Reduce $L$ and (2) Enlarge $K$.

To reduce L, we want to find the largest sublanguage of $L$ that is decentralized strongly opaque with respect to $K$. Using Corollary 5.3.2, we can show that this sublanguage exists

and is unique. To see this, let us define the set of sublanguages of $L$ that are decentralized strongly opaque with respect to $K$ as

$$DS_K^{sub}(L) = \{L_i \subseteq \mathcal{L}(G) : L_i \subseteq L \wedge (\forall j \in J)\theta_j(L_i) \subseteq \theta_j(K)\}$$

By Corollary 5.3.2, if

$$(\forall j \in J)L_i \in DS_K^{sub}(L), i = 1, 2, \cdots$$

then

$$\bigcup_i L_i \in DS_K^{sub}(L)$$

Therefore, the decentralized supremal element of $DS_K^{sub}(L)$ exists and is given by

$$supDS_K^{sub}(L) = \bigcup_{L_i \in DS_K^{sub}(L)} L_i$$

To enlarge K, we want to find the smallest superlanguage of $K$, so that $L$ is decentralized strongly opaque with respect to the superlanguage. let us define the set of superlanguages of $K$ such that $L$ is decentralized strongly opaque with respect to them as

$$DS_L^{super}(K) = \{K_i \subseteq \mathcal{L}(G) : K \subseteq K_i \wedge (\forall j \in J)\theta_j(L) \subseteq \theta_j(K_i)\}$$

By the results presented in Chapter 4a unique smallest language in $DS_L^{super}(K)$. However, we can always find some minimal elements in $DS_L^{super}(K)$, which will be denoted by $minDS_L^{super}(K)$.

**Weak opacity**

If $L \subseteq \mathcal{L}(G)$ is not decentralized weakly opaque with respect to $K \subseteq \mathcal{L}(G)$, that is, $(\forall j \in J)\theta_j(L) \cap \theta_j(K) = \emptyset$, then we may want to enlarge $L$ or $K$ in order to satisfy decentralized weak opacity. Since $L$ and $K$ are symmetric, we discuss only how to enlarge $L$. Denote the set of superlanguages of $L$ that are weakly opaque with respect to $K$ as

$$DW_K^{super}(L) = \{L_i \subseteq \mathcal{L}(G) : L \subseteq L_i \wedge (\forall j \in J)\theta_j(L_i) \cap \theta_j(K) \neq \emptyset\}$$

Similar to $DS_L^{super}(K)$ the unique smallest language in $DW_K^{super}(L)$ may not exist. In other words, the infimal element $infDW_K^{super}(L)$ may not exist. However, we can always find some minimal elements in $DW_K^{super}(L)$, which will be denoted by $minDW_K^{super}(L)$.

**No opacity**

For $L, K \subseteq \mathcal{L}(G)$, $L$ is not decentralized opaque with respect to $\Theta = \{\theta_1, \cdots, \theta_n\}$ and $K$ if $(\forall j \in J)\theta_j(L) \cap \theta_j(K) = \emptyset$. If this is not true, that is $(\exists j \in J)\theta_j(L) \cap \theta_j(K) \neq \emptyset$, then we may want to shrink $L$ or $K$ in order to satisfy decentralized no opacity. Since $L$ and $K$ are symmetric, we discuss only how to shrink $L$.

To shrink $L$, we want to find the largest sublanguage of $L$ that is not decentralized opaque with respect to $K$ and $\Theta$. By Corollary 5.3.5, we can show that this sublanguage exists and is unique. To see this, let us define the set of sublanguage of $L$ that is not decentralized opaque with respect to $K$ as

$$DN_K^{sub}(L) = \{L_i \subseteq \mathcal{L}(G) : L_i \subseteq L \wedge (\exists j \in J)\theta_j(L_i) \cap \theta_j(K) = \emptyset\}$$

By Corollary 5.3.5, if

$$L_i \in DN_K^{sub}(L), i = 1, 2, \cdots$$

then

$$\bigcup_i L_i \in DN_K^{sub}(L)$$

Therefore, the supremal element of $DN_K^{sub}(L)$ exists and is given by

$$supDN_K^{sub}(L) = \bigcup_{L_i \in DN_K^{sub}(L)} L_i$$

With these definitions, we are now ready to compute the formulas for decentralized opaque superlanguages and sublanguages.

## 5.4 Formulas and Algorithms for Decentralized Opaque Superlanguages and Sublanguages

In this section we present formulas for decentralized opaque superlanguages and sublanguages. Using the following formulas is useful to calculate various opaque superlanguages and sublanguages in the decentralized framework. We first consider $DS_K^{sub}(L)$.

**Theorem 5.4.1.** *For any observation set $\Theta$, languages $L_i \subseteq \mathcal{L}(G), i = 1, 2, 3, \cdots$ and $K \subseteq \mathcal{L}(G)$, if $L_i$ is strongly decentralized opaque with respect to $K$, then the union $\bigcup_i L_i$ is also*

*strongly decentralized opaque with respect to $K$. Therefore, the supremal element of $DS_K^{sub}(L)$ exist. It is given by*

$$supDS_K^{sub}(L) = L \cap (\bigcap_{j\in J}(\theta_j^{-1}\theta_j(K)).$$

*Proof.* To prove $supDS_K^{sub}(L) \subseteq L \cap (\bigcap_{j\in J}(\theta_j^{-1}\theta_j(K))$, we need to show

(1) $supDS_K^{sub}(L) \subseteq L$.

(2) $(\forall j \in J) supDS_K^{sub}(L) \subseteq \theta_j^{-1}\theta_j(K)$.

Both are obvious.

To prove $supDS_K^{sub}(L) \supseteq L \cap (\bigcap_{j\in J}(\theta_j^{-1}\theta_j(K))$, we need to show

1. $L \cap (\bigcap_{j\in J}(\theta_j^{-1}\theta_j(K)) \subseteq L$.

2. $L \cap (\bigcap_{j\in J}(\theta_j^{-1}\theta_j(K))$ is strongly decentralized opaque with respect to $K$ and $\Theta$.

Condition 1 is obviously true. To prove Condition 2 is true, we need to prove

$$(\forall i \in J)\theta_i(L \cap \bigcap_{j\in J}(\theta_j^{-1}\theta_j(K))) \subseteq \theta_i(K)$$

This can be done as follows. For all $i \in J$

$$\begin{aligned}
& s \in \theta_i(L \cap (\bigcap_{j\in J}(\theta_j^{-1}\theta_j(K)))) \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta_i(t) = s \wedge t \in (L \cap (\bigcap_{j\in J}(\theta_j^{-1}\theta_j(K)))) \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta_i(t) = s \wedge t \in L \wedge t \in \bigcap_{j\in J}\theta_j^{-1}\theta_j(K) \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta_i(t) = s \wedge t \in L \wedge (\forall j \in J)\theta_j(t) \in \theta_j(K) \\
\Rightarrow\ & s = \theta_i(t) \in \theta_i(K).
\end{aligned}$$

□

We next consider $DS_L^{super}(K)$. Since centralized opacity is a special case of decentralized opacity, it is easy to conclude that the infimal element of $DS_L^{super}(K)$ may not exist. But we can always find a minimal element of $DS_L^{super}(K)$, which is denoted by $minDS_L^{super}(K)$. The procedure, however, is more complex for the decentralized case. We define for $j \in J$, $\hat{\theta}_j^{-1}(t)$ and $\hat{\theta}_j^{-1}(\theta_j(L) - \theta_j(K))$ similarity to $\hat{\theta}^{-1}(t)$ and $\hat{\theta}^{-1}(\theta(L) - \theta(K))$. Let

$$\begin{aligned}
B_j &= \theta_j(L) - \theta_j(K) \\
A_j &= \hat{\theta}_j^{-1}(B_j).
\end{aligned}$$

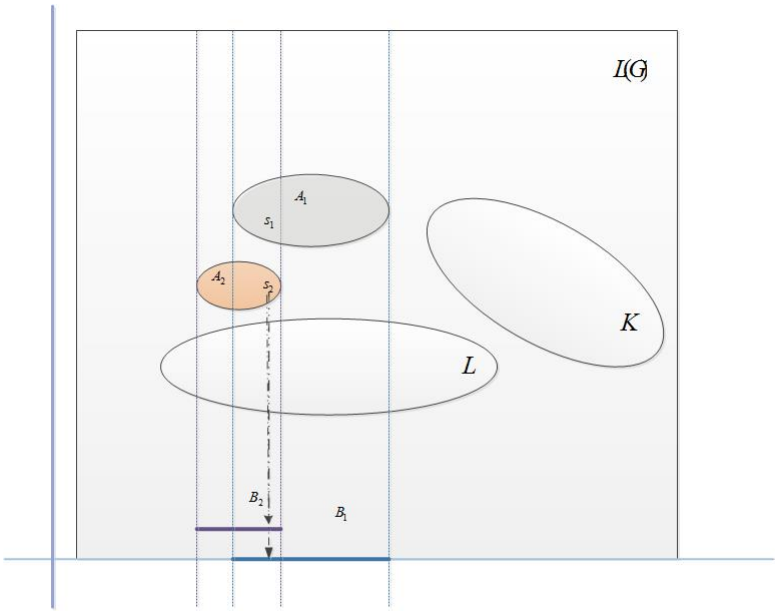



Figure 5.2: Illustration of satisfying $minDS_L^{super}(K)$

If we add $A_1 \cup \ldots \cup A_n$ to $K$, then we can ensure that decentralized strong opacity is satisfied. However, adding all $A_1 \cup \ldots \cup A_n$ to $K$ may be too much. To show that we consider the system in Figure 5.2. Let strings $s_1 \in A_1$ and $s_2 \in A_2$ such that $\theta_1(s_1) = t_1 \in B_1$ and $\theta_2(s_2) = t_2 \in B_2$. Clearly, to ensure decentralized strong opacity with respect to $\theta_1$ and $\theta_2$ we add $A_1$ and $A_2$ to $K$. However, we may have the case that $\theta_1(s_2) = t_1 \in B_1$. That means, adding only $s_2$ can also satisfy decentralized strong opacity with respect to $\theta_1$ and $\theta_2$. In view of the case mentioned, if for $j \neq i$, $\theta_j(A_i) \cap B_j \neq \emptyset$, then some strings in $A_j$ can be removed without violating decentralized strong opacity. The following algorithm removes these strings by updating $A_j$ as follows.

$$A_j = A_j - \theta^{-1}(\theta_j(A_i) \cap B_j).$$

We need to do this for all $i \neq J$. The formal algorithm to calculate $minDS_L^{super}(K)$ is given below.

**Algorithm 5.4.1.** (To calculate $minDS_L^{super}(K)$)
Input: $\mathcal{L}(G), K, L$, and $\Theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$.

Output: $minDS_L^{super}(K)$.

Step 1 For all $i = 1, 2, \ldots, n$, do

$B_i = \theta_i(L) - \theta_i(K)$;

$A_i = \hat{\theta}_i^{-1}(B_i)$;

Step 2 For all $i = 1, 2, \ldots, n$ do

For all $j \neq i$, do

$A_j = A_j - \theta_j^{-1}(\theta_j(A_i) \cap B_j)$;

Step 3 $A = \bigcup_{i=1}^{n} A_i$;

Step 4 $minDS_L^{super}(K) = K \cup A$.

**Theorem 5.4.2.** *For any observation set $\Theta$, languages $K, L \subseteq \mathcal{L}(G)$, the computational complexity of Algorithm 5.4.1 is $minDS_L^{super}(K)$.*

*Proof.* We need to show the following

1. $K \cup A$ is a superlanguage of $K$, that is, $K \subseteq K \cup A$
2. $L$ is decentralized strongly opaque with respect to $K \cup A$ and $\Theta$, that is,

$$(\forall j \in J)\theta_j(L) \subseteq \theta_j(K \cup A).$$

3. Removing any string in $K \cup A$ violates either Condition 1 or Condition 2.

Condition 1 is obvious true. Condition 2 can be proved in two steps. First, for all $j \in J$, we prove

$$B_j \subseteq \theta_j(A_1 \cup \ldots \cup A_n) = \theta_j(A)$$

by contradiction. Suppose $B_j \nsubseteq \theta_j(A_1 \cup \ldots \cup A_n)$, that is, there exists $t$ such that

$$t \in B_j \wedge t \notin \theta_j(A_1 \cup \ldots \cup A_n).$$

Then there exists $u$ such that $\theta_j(u) = t$ and $u$ is removed from $\hat{\theta}_j^{-1}(B_j)$ in Step 2. By Step 2; $u$ is removed from $\hat{\theta}_j^{-1}(B_j)$ because $\theta_j(u) \in \theta_j(A_i)$ for some $i \neq j$. this contradicts the assumption $t \notin \theta_j(A_1 \cup \ldots \cup A_n)$. Next, we have

$$\theta_j(L) \subseteq \theta_j(K) \cup \theta_j(L) = \theta_j(K) \cup (\theta_j(L) - \theta_j(K)) = \theta_j(K) \cup B_j \subseteq \theta_j(K) \cup \theta_j(A) = \theta_j(K \cup A).$$

To prove Condition 3, we note that removing any string from $K$ will violate Condition 1 and removing any string from $A$ will violate Condition 2 because all strings that can be removed have already been removed in Step 2. □

The following example illustrates Algorithm 1.

**Example 5.4.1.** Let $\Sigma = \{\alpha, \beta\}$, $\mathcal{L}(G) = \Sigma^*$, and $\Theta = \{\theta_1, \theta_2, \theta_3\}$. Suppose that $\theta_1$ can see only $\beta$, $\theta_2$ can see only $\alpha$ and $\theta_3$ can see $\gamma$. Let

$$L = \{\alpha\beta, \alpha\gamma\}$$
$$K = \{\beta\alpha\gamma\gamma\beta\}.$$

Clearly $L$ is not strongly decentralized opaque with respect to $K$ and $\Theta$. Let us calculate $minDS_L^{super}(K)$ using Algorithm 5.4.1.

Step 1

$B_1 = \theta_1(L) - \theta_1(K) = \{\epsilon, \beta\} - \{\beta\beta\} = \{\epsilon, \beta\}$

$B_2 = \theta_2(L) - \theta_2(K) = \{\alpha\} - \{\alpha\} = \emptyset$

$B_3 = \theta_3(L) - \theta_3(K) := \{\epsilon, \gamma\} - \{\gamma\gamma\} = \{\epsilon, \gamma\}$

$A_1 = \hat{\theta}_1^{-1}(B_1) = \{\alpha, \gamma\beta\}$

$A_2 = \hat{\theta}_2^{-1}(B_2) = \{\emptyset\}$

$A_3 = \hat{\theta}_3^{-1}(B_3) = \{\alpha, \beta\gamma\}$

Step 2

For $i = 1, j = 2$

$A_2 = A_2 - \theta_2^{-1}(\theta_2(A_1) \cap B_2) = \emptyset$

For $i = 1, j = 3$

$A_3 = A_3 - \theta_3^{-1}(\theta_3(A_1) \cap B_3) = \{\alpha\}$

For $i = 2, j = 1$

$A_1 = A_1 - \theta_1^{-1}(\theta_1(A_2) \cap B_1) = \{\alpha, \gamma\beta\}$

For $i = 2, j = 3$

$A_3 = A_3 - \theta_3^{-1}(\theta_3(A_2) \cap B_3) = \{\alpha\}$

For $i = 3, j = 1$

$A_1 = A_1 - \theta_1^{-1}(\theta_1(A_3) \cap B_1) = \{\gamma\beta\}$

For $i = 3, j = 2$

$A_2 = A_2 - \theta_2^{-1}(\theta_2(A_3) \cap B_2) = \emptyset$

Steps 3 and 4

$minDS_L^{super}(K) = K \cup A_1 \cup A_2 \cup A_3 = \{\beta\alpha\gamma\gamma\beta, \gamma\beta, \alpha\}$

Similarly to $DS_L^{super}(K)$, the centralized weak opacity $W_K^{super}(L)$ is a special case of decentralized weak opacity $DW_K^{super}(L)$. Therefore, the infimal element of $DW_K^{super}(L)$ may also not exist. However, a minimal element of $DW_K^{super}(L)$ always exists and is denoted by $minDW_K^{super}(L)$. To calculate $minDW_K^{super}(L)$, we proceed as follows. For all $j \in J$, if $\theta_j(L) \cap \theta_j(K) \neq \emptyset$, then let $C_j = \emptyset$, else pick one string $t_j \in \theta_j(K)$ and let

$$C_j = \hat{\theta}_j^{-1}(\{t_j\}).$$

If we add $C_1 \cup \ldots \cup C_n$ to $L$, then we can ensure that decentralized weak opacity is satisfied. However, adding all $C_1 \cup \ldots \cup C_n$ to $L$ may be too much. This can be seen using the system presented in Figure 5.3. Let strings $s_1 \in C_1$ and $s_2 \in C_2$ such that $\theta_1(s_1) = t_1 \in \theta_1(C_1)$ and $\theta_2(s_2) = t_2 \in \theta_2(C_2)$. Clearly, to ensure decentralized weak opacity we add $C_1$ and $C_2$ to $L$. However, we may have the case that $\theta_1(s_2) = t_1 \in \theta_1(C_1)$. That means, adding only $s_2$ can satisfy decentralized weak opacity. It follows that for $j \neq i$, $t_j \in \theta_j(C_i)$, $s_j$ does not need to be added in order to ensure decentralized weak opacity. The formal algorithm to calculate $minDW_K^{super}(L)$ is given below.

**Algorithm 5.4.2.** (To calculate $minDW_K^{super}(L)$)

Input: $\mathcal{L}(G), K, L$, and $\Theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$.

Output: $minDW_K^{super}(L)$.

Step 1 For all $j = 1, 2, \ldots, n$, do

If $\theta_j(L) \cap \theta_j(K) \neq \emptyset$, then

$C_j = \emptyset$;

Else

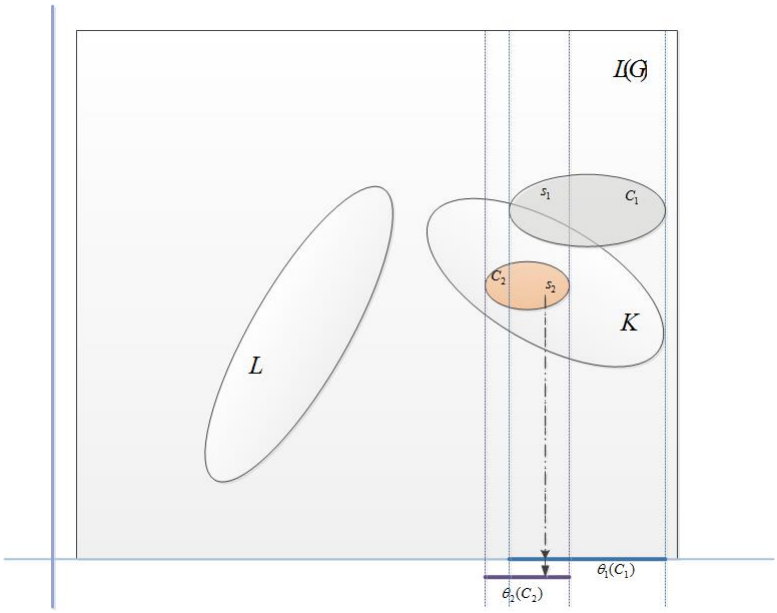



Figure 5.3: Illustration of satisfying $minDW_K^{super}(L)$

pick $t_j \in \theta_j(K)$;

$C_j = \hat{\theta}_j^{-1}(\{t_j\})$;

Step 2 For all $i = 1, 2, \ldots, n$ do

For all $j \neq i$, do

$C_j = C_j - \theta_j^{-1}(\theta_j(C_i) \cap \{t_j\})$;

Step 3 $C = \bigcup_{i=1}^{n} C_i$;

Step 4 $minDW_K^{super}(L) = L \cup C$.

**Theorem 5.4.3.** *For any observation set $\Theta$, languages $K, L \subseteq \mathcal{L}(G)$, the computational complexity of Algorithm 5.4.2 is $minDW_K^{super}(L)$.*

*Proof.* We need to show the following

1. $L \cup C$ is a superlanguage of $L$, that is, $L \subseteq L \cup C$

2. $K$ is decentralized weakly opaque with respect to $L \cup C$ and $\Theta$, that is,

$$(\forall j \in J)\theta_j(K) \cap \theta_j(L \cup C) \neq \emptyset.$$

3. Removing any string in $L \cup C$ violates either Condition 1 or Condition 2.

Condition 1 is obvious true. To prove Condition 2, we consider two possible cases. Case 1: $C_j = \{s_j\} \neq \emptyset$. In this case,

$$\begin{aligned} \theta_j(K) \cap \theta_j(L \cup C) &\supseteq \theta_j(K) \cap \theta_j(C_j) \\ &= \theta_j(K) \cap \theta_j(\{s_j\}) \\ &= \theta_j(K) \cap \{t_j\} \\ &= \{t_j\}. \end{aligned}$$

Hence $\theta_j(K) \cap \theta_j(L \cup C) \neq \emptyset$. Case 2: $C_j = \emptyset$. There are two possibilities. The first possibility: $\theta_j(K) \cap \theta_j(L) \neq \emptyset$. which implies $\theta_j(K) \cap \theta_j(L \cup C) \neq \emptyset$. The second possibility: $s_j$ is remove from $C_j$ is Step 2. By Step 2, $s_j$ is remove from $C_j$ because there exist $i \in J$ such that

$$\begin{aligned} & s_j \in \theta_j^{-1}(\theta_j(C_i) \cap \{t_j\}) \\ \Leftrightarrow \quad & t_j \in \theta_j(C_i) \cap \{t_j\} \\ \Leftrightarrow \quad & t_j \in \theta_j(C_i). \end{aligned}$$

Therefore,

$$\begin{aligned} \theta_j(K) \cap \theta_j(L \cup C) &\supseteq \theta_j(K) \cap \theta_j(C_i) \\ &\supseteq \theta_j(K) \cap \{t_j\} \\ &= \{t_j\}. \end{aligned}$$

Hence $\theta_j(K) \cap \theta_j(L \cup C) \neq \emptyset$. To prove Condition 3, we note that removing any string from $L$ will violate Condition 1 and removing any string from $C$ will violate Condition 2 because all strings that can be removed have already been removed in Step 2. □

The following example illustrates Algorithm 2.

**Example 5.4.2.** Let $\Sigma = \{\alpha, \beta\}$, $\mathcal{L}(G) = \Sigma^*$, and $\Theta = \{\theta_1, \theta_2, \theta_3\}$. Suppose that $\theta_1$ can see only $\beta$, $\theta_2$ can see only $\alpha$ and $\theta_3$ can see $\{\gamma\}$. Let

$$\begin{aligned} L &= \{\alpha\alpha\gamma\beta\beta, \gamma\alpha\alpha\beta\beta, \gamma\alpha\alpha\beta\beta\gamma\} \\ K &= \{\alpha\gamma\beta, \gamma\alpha\beta, \gamma\alpha\beta\gamma, \gamma\alpha\gamma\beta\gamma\}. \end{aligned}$$

$L$ is not decentralized weakly opaque with respect to $K$ and $\Theta$ because $\theta_1(L) \cap \theta_1(K) = \emptyset$ and $\theta_2(L) \cap \theta_2(K) = \emptyset$. Let us calculate $minDW_K^{super}(L)$ using Algorithm 2.

Step 1

pick $t_1 = \beta$, $C_1 = \{\alpha\gamma\beta\}$

pick $t_2 = \alpha$, $C_2 = \{\alpha\beta\gamma\}$

$C_3 = \emptyset$

Step 2

For $i = 1, j = 2$

$C_2 = C_2 - \theta_2^{-1}(\theta_2(C_1) \cap \{t_2\}) = \emptyset$

The rest of iterations will not change $C_1, C_2, C_3$

Steps 3 and 4

$minDW_K^{super}(L) = L \cup C_1 \cup C_2 \cup C_3 = \{\alpha\alpha\gamma\beta\beta, \gamma\alpha\alpha\beta\beta, \gamma\alpha\alpha\beta\beta\gamma, \alpha\gamma\beta\}$

Finally, let us consider $DN_K^{sub}(L)$.

**Theorem 5.4.4.** *For any observation set $\Theta$, languages $L_i \subseteq \mathcal{L}(G), i = 1, 2, 3, \cdots$ and $K \subseteq \mathcal{L}(G)$, if $L_i$ is not decentralized opaque with respect to $K$, then the union $\bigcup_i L_i$ is also not decentralized opaque with respect to $K$. Therefore, the supremal element of $DN_K^{sub}(L)$ exist. It is given by*

$$supDN_K^{sub}(L) = L - \bigcup_{j \in J} \theta_j^{-1}\theta_j(K).$$

*Proof.* First we show that $supDN_K^{sub}(L) \subseteq L - \bigcup_{j \in J} \theta_j^{-1}\theta_j(K)$. Since $supDN_K^{sub}(L) \subseteq L$, we only need to show that $supDN_K^{sub}(L) \cap (\bigcup_{j \in J} \theta_j^{-1}\theta_j(K)) = \emptyset$. Assume the contrary, then

$$\begin{aligned}
&(\exists s \in \Sigma^*) s \in supDN_K^{sub}(L) \wedge s \in \bigcup_{j \in J} \theta_j^{-1}\theta_j(K) \\
\Rightarrow\ &(\exists s \in \Sigma^*) s \in supDN_K^{sub}(L) (\exists j \in J) s \in \theta_j^{-1}\theta_j(K) \\
\Rightarrow\ &(\exists j \in J)(\exists s \in \Sigma^*) s \in supDN_K^{sub}(L) \wedge s \in \theta_j^{-1}\theta_j(K) \\
\Rightarrow\ &(\exists j \in J)(\exists s \in \Sigma^*) \theta_j(s) \in \theta_j(supDN_K^{sub}(L)) \wedge \theta_j(s) \in \theta_j(K) \\
\Rightarrow\ &(\exists j \in J) \theta_j(K) \cap \theta_j(supDN_K^{sub}(L)) \neq \emptyset
\end{aligned}$$

This contradicts the fact that $supDN_K^{sub}(L)$ is not decentralized opaque.

To prove $supDN_K^{sub}(L) \supseteq L - \bigcup_{j \in J} \theta_j^{-1}\theta_j(K)$, we need to show

1. $L - \bigcup_{j \in J} \theta_j^{-1}\theta_j(K) \subseteq L$.

2. $L - \bigcup_{j \in J} \theta_j^{-1}\theta_j(K)$ is not decentralized opaque with respect to $K$ and $\Theta$.

Condition 1 is obviously true. To prove that Condition 2 is also true we assume the contrary, then there exist $i \in J$

$$
\begin{aligned}
& \theta_i(L - \bigcup_{j \in J} \theta_j^{-1}\theta_j(K)) \cap \theta_i(K) \neq \emptyset \\
\Rightarrow\ & (\exists s \in \Sigma^*) s \in \theta_i(L - \bigcup_{j \in J} \theta_j^{-1}\theta_j(K)) \wedge s \in \theta_i(K) \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta_i(t) = s \wedge t \in (L - \bigcup_{j \in J} \theta_j^{-1}\theta_j(K)) \wedge s \in \theta_i(K) \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta_i(t) = s \wedge t \in L \wedge t \notin \bigcup_{j \in J} \theta_j^{-1}\theta_j(K) \wedge s \in \theta_i(K) \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta_i(t) = s \wedge t \in L \wedge (\forall j \in J)\theta_j(t) \notin \theta_j(K) \wedge s \in \theta_i(K) \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta_i(t) = s \wedge t \in L \wedge (\forall j \in J)\theta_j(t) \notin \theta_j(K) \wedge \theta_i(t) \in \theta_i(K) \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta_i(t) = s \wedge t \in L \wedge \theta_i(t) \notin \theta_i(K)) \wedge \theta_i(t) \in \theta_i(K)
\end{aligned}
$$

This leads to a contradiction: $(\theta_i(t) \in \theta_i(K) \wedge \theta_i(t) \notin \theta_i(K))$.

□

# 6 Dining Cryptographers

## 6.1 Dining Cryptographers Problem

In this Chapter we illustrate the introduced concept of opacity in the framework of discrete event system to the property of anonymity. Particularly we apply the proposed opacity to verify and to synthesize the dining cryptographers protocol. Generally speaking, anonymity is described as keeping the identity of agents participating in a certain action secret. Anonymity is one of the properties that is widely needed, for instance, in electronic voting and web browsing.

As one of the wide used security properties, anonymity differs from other properties like non-interference, confidentiality and privacy where anonymity property means the identity of the user of a certain action be kept secret for an observer who does not have a full access to the system. To describe anonymity more accurately we use the voting system as an example. In this case anonymity means that the identity of the voter associated with each vote must be hidden, and not the vote itself or the candidate voted for [2]. The difference between anonymity and other information-hiding properties were discussed in [14], [15]. The characterisation of anonymity is usually relative to the capabilities of the user to observe the activities associated.

The capabilities of the observer is usually considered one of the most important specifications that characterize the anonymity. For example, the case of an anonymous bulletin board, a posting by one member of the group be kept anonymous to the other members; however, it may be possible that the administrator of the board has access to some privileged information that may allow him to infer the identity of the member who posted it [2, 27, 28]. In such cases, a protocol is necessary to specify the set(s) of members or agents that have to be kept anonymous. In other words, the secret set is defined as a group of members in which any user can test their membership in the group but can determine neither the other group members nor the size of the group.

For analyzing anonymity many definitions and frameworks have been developed in the literature. The major concept of analyzing anonymity is based on the principle of confusion. In [14, 47, 53, 57], many approaches have been presented. In general, all approaches indicate to the principle of confusion. Another concept to analyze anonymity is presented in [55]. Using a discrete-time Markove chain a degree of anonymity is described and proved. This approach presents a group members and the adversary participating in Web browsing as a discrete time Markove chain model. In this case, the security properties are easily identified using probabilistic logic formulas.

In [54] the degree of anonymity is presented using a discrete event system. It is shown that a system is anonymous for a set of events $R \subset \Sigma$ if it is possible to permute them and if it is undetectable for an observer. The strong anonymous is defined as follows; if it is impossible for an observer to make a difference between any two occurred strings. Otherwise, a system is weak anonymous if it is impossible to identify a string but possible to detect the occurrence of all events in the same string. In other words, with weak anonymity the sets of messages have the same user can be determined, but the identity of the user is still kept secret.

In [33] anonymity was defined in the framework of opacity. Based on the properties of strong and weak anonymous, anonymity can be studied using opacity definitions. By properly defining two languages corresponding to each action, a set of actions is strongly anonymous if and only if for each action, the two corresponding languages are strongly opaque. Similarly, a set of actions is weakly anonymous if and only if the languages are weakly opaque.

## 6.2 System Description

We now introduce the dining cryptographic protocol described by Chaum in [7]. "Three cryptographers are having dinner. Their waiter informs them that arrangements have been made with the maitre d'hotel for the bill to be paid anonymously. One of the cryptographers might be paying for a dinner, or it might have been NSA (U.S. National Security Agency).

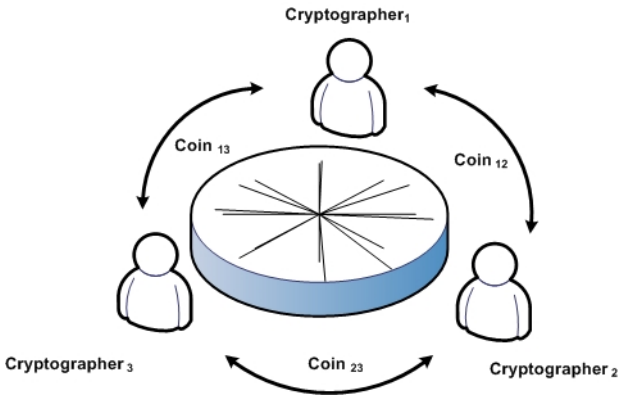



Figure 6.1: Dining Cryptographers Topology

The cryptographers respect each other's right to make an anonymous payment, but want to find out whether the NSA paid. So they decide to execute the following protocol".

A possible solution to this problem, described in [7], is that each cryptographer flips an unbiased coin behind his menu, between him and the cryptographer on his left, so that only the two of them can see the outcome as shown in Figure 6.1. The $\text{cryptographer}_1$ can see only the $\text{coin}_{12}$ and $\text{coin}_{13}$, the $\text{cryptographer}_2$ sees $\text{coin}_{12}$ and $\text{coin}_{23}$, and finally the $\text{cryptographer}_3$ is able to see only the $\text{coin}_{23}$ and $\text{coin}_{13}$. Each cryptographer then states aloud *agree* and *disagree* based on the sides of the coins. If a cryptographer is not paying, he will state *disagree* if the tow sides are not the same and *agree* if-they are. If a cryptographer is paying, he will state the opposite. In [7] it is proved that if the number of disagrees is even then NSA is paying; otherwise, one of the cryptographers is paying.

The intuition behind the protocol is best introduced with the following description. Suppose that three participants in a network possess k-bit messages. Each participant shares a secret key (1-bit random message) with others separately. Clearly, for $i = 1, 2, 3$ participant 1 has two secrets $k_{31}$ and $k_{12}$ in common with the two other participant 3 and 2 respectively. In addition, each participant has a secret $1 - bit$ message that is equal to 1 in the case that the participant is paid, and 0 otherwise [16]. All participants broadcast the value $b_i = k_{i;i-1} \bigoplus k_{i;i+1} \bigoplus m_i$, where $\bigoplus$ denotes the exclusive OR operation (bitwise addition modulo 2). Every observer of all broadcast can reconstruct the message by summing up the keys

and add secret messages to the sum $b_1 \bigoplus b_2 \bigoplus b_3$. Only one of the participants should send a message at the same time. Since there exist only one payer and if broadcast $b_i$ contained for $i = 1, 2, 3$ and $j = 1, 2, 3$ $k_{i;j}$ then by the protocol $b_j$ must contain $k_{j;i}$. As $k_{i;j} = k_{j;i}$, it must be the case that $k_{i;i+1} \bigoplus k_{j;j-1} = 0$. Therefore all $k_{i;j}$ cancel out:

$$\begin{aligned} b_1 \bigoplus b_2 \bigoplus b_3 &= (k_{1;2} \bigoplus k_{1;3} \bigoplus m_1) \bigoplus (k_{2;1} \bigoplus k_{2;3} \bigoplus m_2) \bigoplus (k_{3;1} \bigoplus k_{3;2} \bigoplus m_3) \\ &= m_1 \bigoplus m_2 \bigoplus m_3 \end{aligned}$$

Assuming that the $participant_2$ is paying, then $m_2 = 1$ and $m_1 = m_3 = 0$. Therefore $b_1 \bigoplus b_2 \bigoplus b_3 = m_2$. This refers to no-opacity property where if one of the participants sends a message should not be opaque. In the same time if $participant_2$ payes than no one from other participants know who is the payer. This refers to strong opacity where if $participant_1$ knows all three values $b_i$ for $i = 1, 2, 3$ as well the keys $k_{1;2}$ and $k_{1;3}$, then he cannot know who of both sent the message. Similarly for $participant_3$ he cannot know who of both sent the message.

In this section, we use the dining cryptographers problem to illustrate strong opacity and no opacity. We first verify the dining cryptographers protocol and then see if we can synthesize the protocol.

## 6.3 Verification

To formulate the dining cryptographers problem in the discrete event system framework, let us define the following events.

- $P_i$ : the event that the cryptographer$_i$ is paying, for $i = 1, 2, 3$.

- $N_i$ : the event that the cryptographer$_i$ is not paying, for $i = 1, 2, 3$.

- $h_{ij}$ : the event that the outcome of the coin$_{ij}$ was a head.

- $t_{ij}$ : the event that the outcome of the coin$_{ij}$ was a tail.

- $A_i$ : the event that the cryptographer$_i$ states *agree*, for $i = 1, 2, 3$.

- $D_i$ : the event that the cryptographer$_i$ states *disagree*, for $i = 1, 2, 3$.

The complete behavior (all possible strings) of the system without protocol is given by the automaton $G$ in Figure 6.2. The set of events that observable to cryptographer$_1$ is given by $\Sigma_{1,o} = \{P_1, N_1, h_{13}, t_{13}, h_{12}, t_{12}, A_i, D_i, i = 1, 2, 3\}$. The correspondence observation mapping is denoted by $\theta_1$. $\Sigma_{2,o}, \theta_2$, $\Sigma_{3,o}, \theta_3$ are defined similarly.

Under the protocol, the behavior of cryptographer$_1$ when he is not the payer is described by language $K_1$. The automaton for $K_1$ is shown in Figure 6.3. Similarly to $K_1$ the language $K_2$ is shown in Figure 6.4,and the language $K_3$ is presented in Figure 6.5. The behavior of cryptographer$_i$ when he is the payer under the protocol is described by language $L_i$ ,$i = 1, 2, 3$. The automaton for $L_1$ is shown in Figure 6.6. The languages $L_2, L_3$ and their automata can be defined similarly. Figures 6.7 and 6.8 are showing the automaton $L_2$ and $L_3$ respectively.

Let us verify the protocol using strong opacity and no opacity. First, we want to show that if no cryptographer is paying, then all three cryptographers know. In other words, the language $K_1 \cap K_2 \cap K_3$ (no cryptographer is paying) is not opaque with respect to $L_1 \cup L_2 \cup L_3$ ( one of cryptographer is paying) and $\theta_i, i = 1, 2, 3$. That is

$\theta_i(L_1 \cup L_2 \cup L_3) \cap \theta_i(K_1 \cap K_2 \cap K_3) = \emptyset$, $i = 1, 2, 3$.

To show this, the automaton for $K_1 \cap K_2 \cap K_3$ is computed and shown in Figure 6.9. The automaton for $L_1 \cup L_2 \cup L_3$ is computed and shown in Figure 6.10. The observed behavior

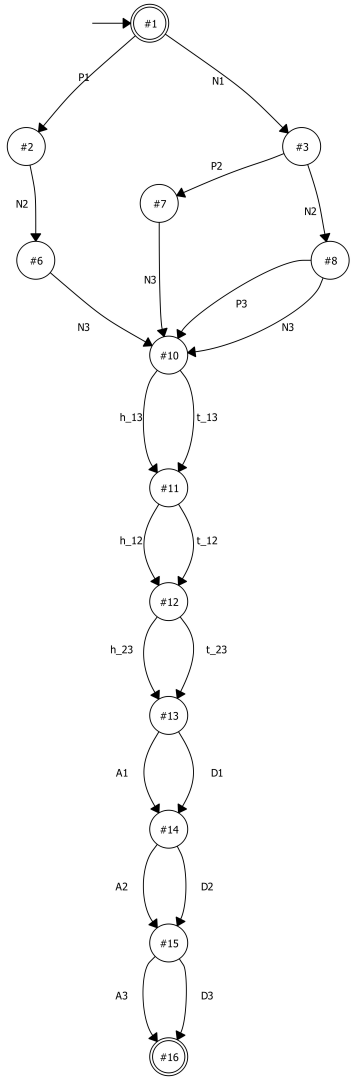



Figure 6.2: Dining Cryptographers Automaton $G$

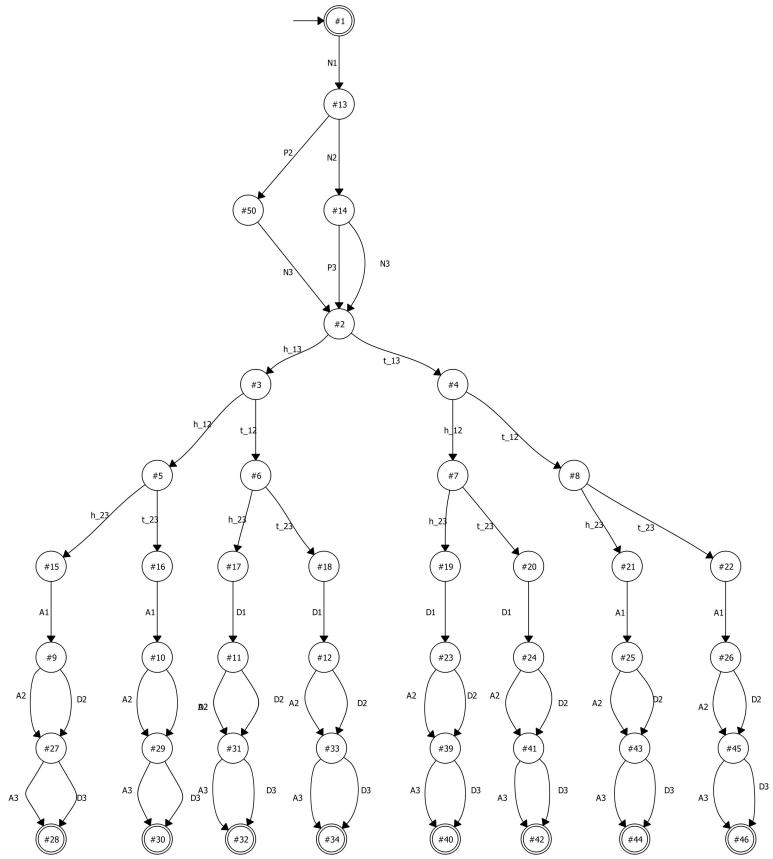



Figure 6.3: $K_1$ : Cryptographer$_1$ is not paying

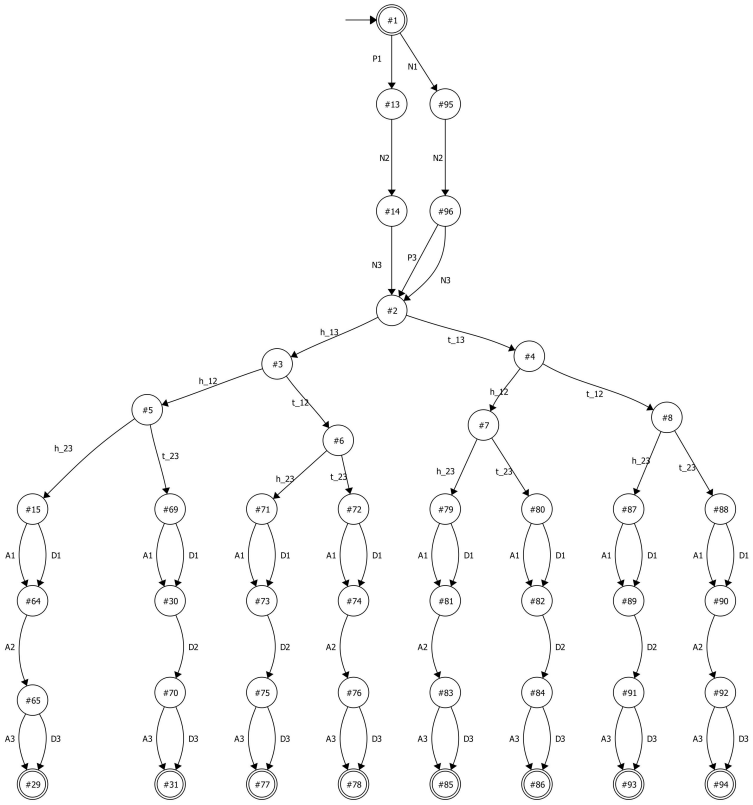



Figure 6.4: $K_2$ : Cryptographer$_2$ is not paying

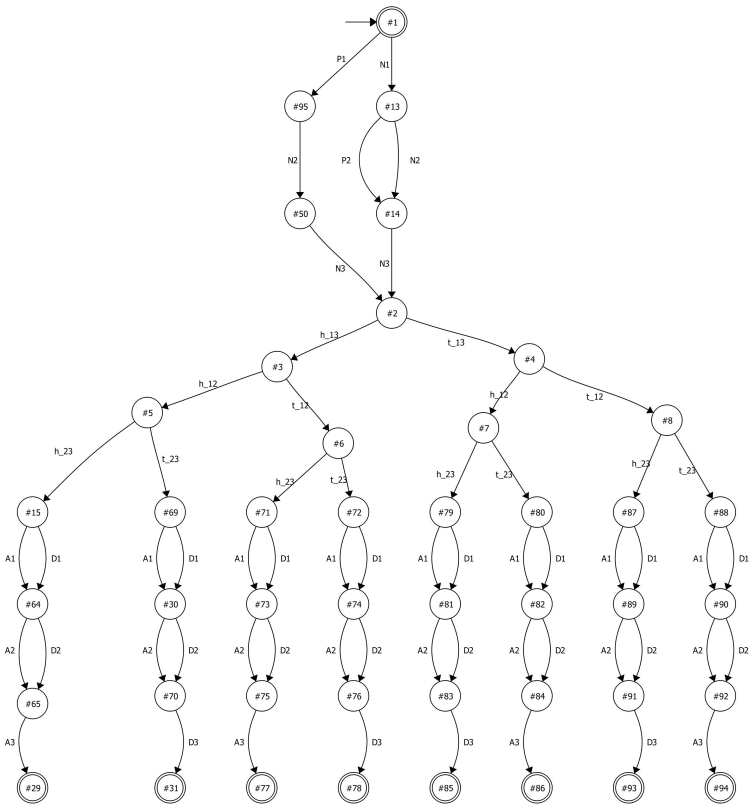



Figure 6.5: $K_3$ : Cryptographer$_3$ is not paying

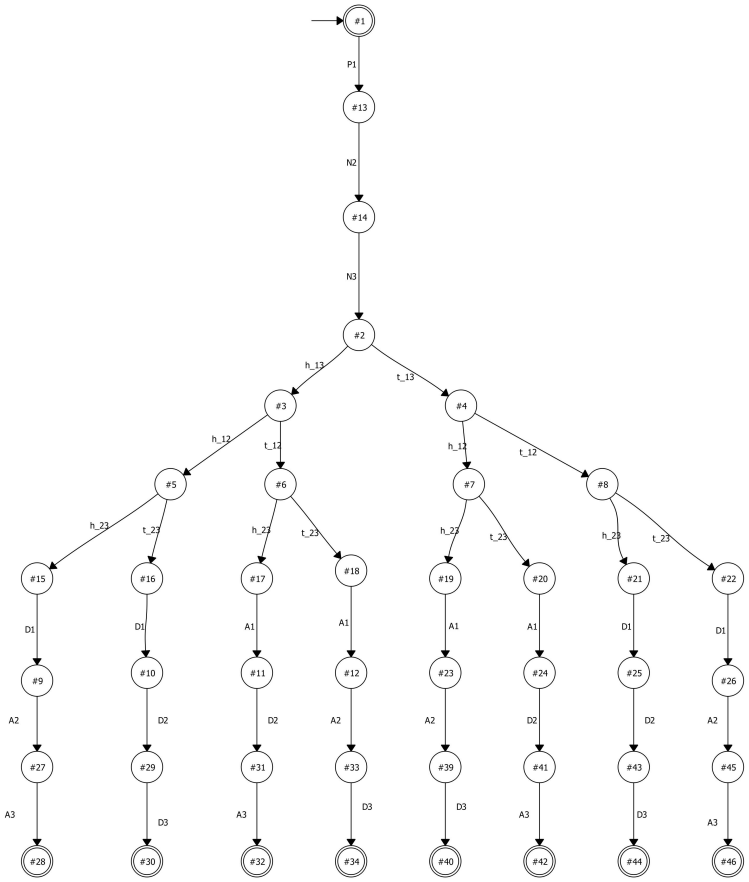



Figure 6.6: $L_1$ : Cryptographer$_1$ is paying

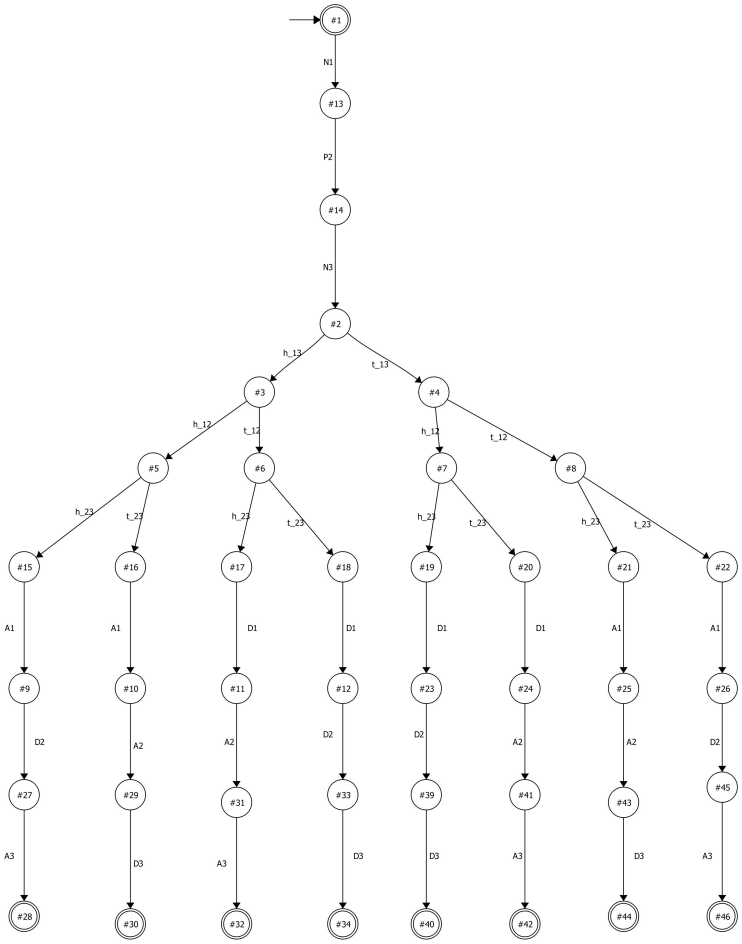



Figure 6.7: $L_2$ : Cryptographer$_2$ is paying

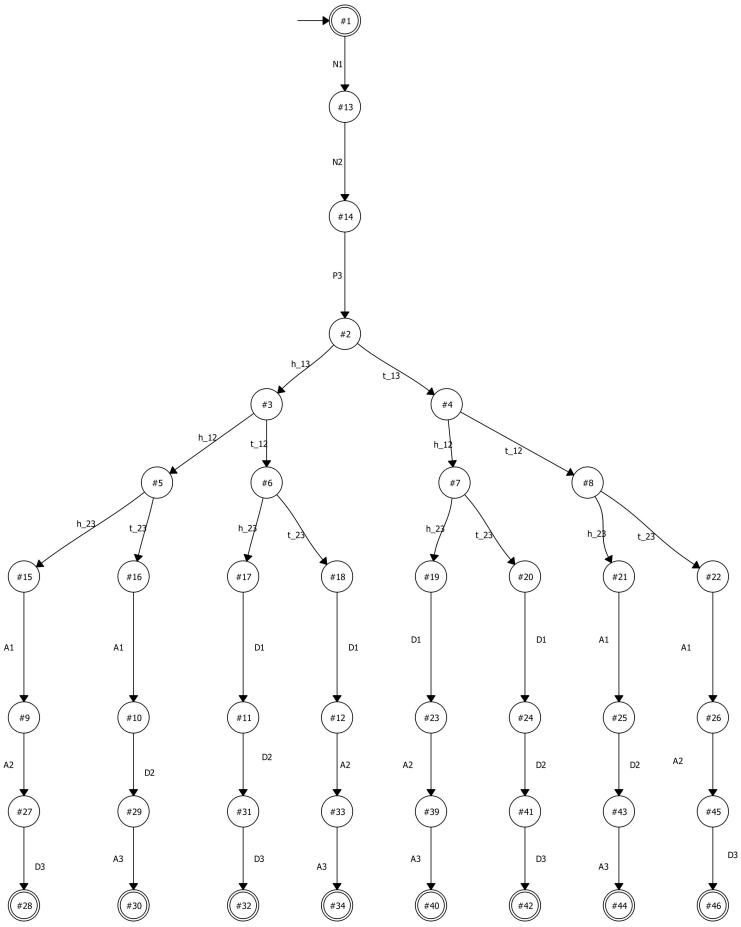



Figure 6.8: $L_3$ : Cryptographer$_3$ is paying

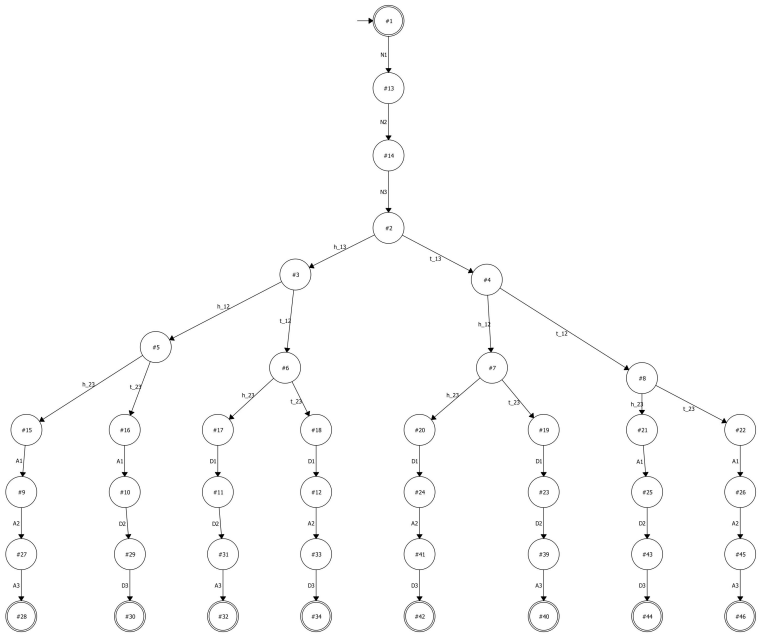


Figure 6.9: Automaton for $K_1 \cap K_2 \cap K_3$

$\theta_1(L_1 \cup L_2 \cup L_3)$ and $\theta_1(K_1 \cap K_2 \cap K_3)$ for cryptographer$_1$ are shown in Figure 6.11. From Figures 6.9, 6.10 and 6.11 it is not difficult to see $\theta_1(L_1 \cup L_2 \cup L_3) \cap \theta_1(K_1 \cap K_2 \cap K_3) = \emptyset$. Similarly, we can show $\theta_2(L_1 \cup L_2 \cup L_3) \cap \theta_2(K_1 \cap K_2 \cap K_3) = \emptyset$ and $\theta_3(L_1 \cup L_2 \cup L_3) \cap \theta_3(K_1 \cap K_2 \cap K_3) = \emptyset$

Next, let us show that if cryptographer$_1$ is paying, then neither cryptographer$_2$ or cryptographer$_3$ knows who is the payer. In other words, for cryptographer$_2$ (with respect to $\theta_2$), $L_1$ is strongly opaque with respect to $L_3$ and $L_3$ is strongly opaque with respect to $L_1$, that is

$\theta_2(L_1) \subseteq \theta_2(L_3) \wedge \theta_2(L_3) \subseteq \theta_2(L_1)$

$\Leftrightarrow \theta_2(L_1) = \theta_2(L_3)$

To see this, Figure 6.8 shows $L_3$ and Figure 6.12 shows the calculated $\theta_2(L_1) = \theta_2(L_3)$. For cryptographer$_3$ (with respect to $\theta_3$), $L_1$ is strongly opaque with respect to $L_2$ and $L_2$ is

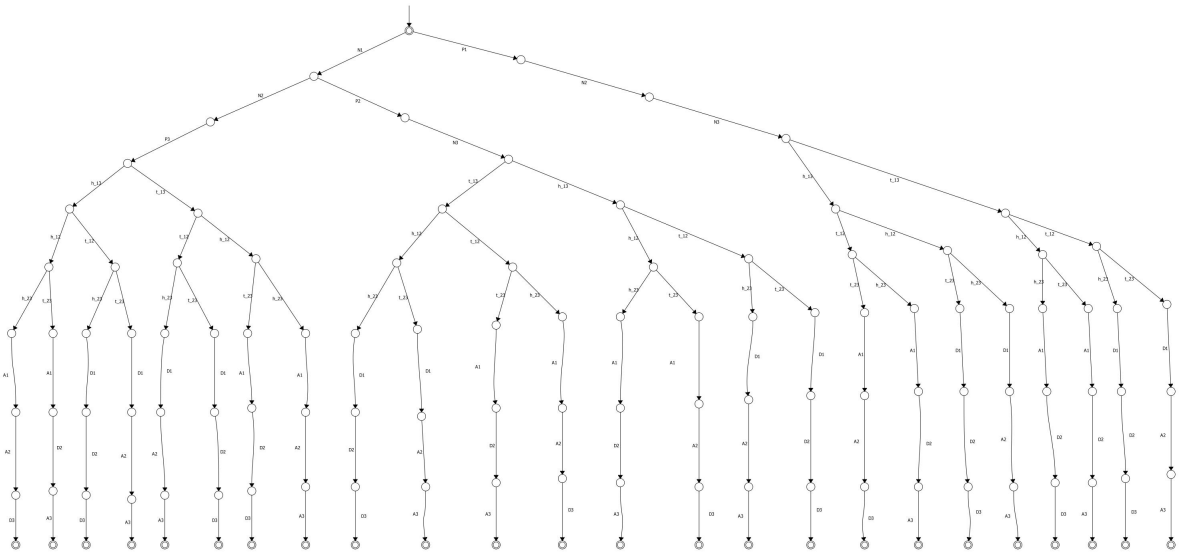


Figure 6.10: Automaton for $L_1 \cup L_2 \cup L_3$

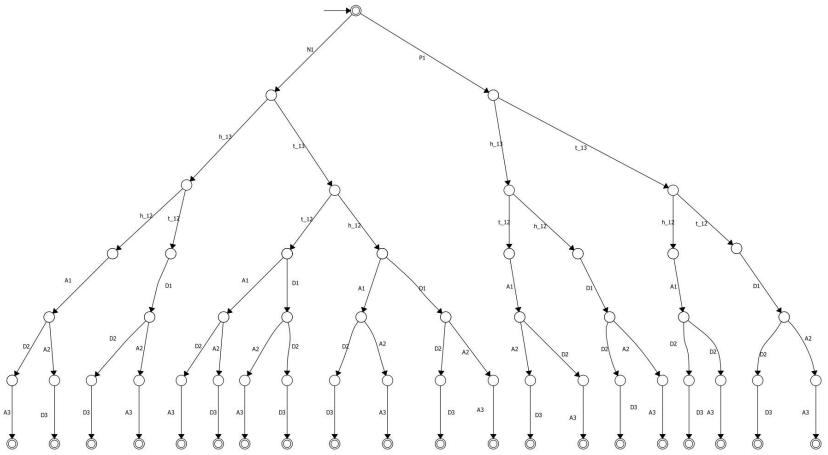
strongly opaque with respect to $L_1$, that is

$$\theta_3(L_1) \subseteq \theta_3(L_2) \wedge \theta_3(L_2) \subseteq \theta_3(L_1)$$
$$\Leftrightarrow \theta_3(L_1) = \theta_3(L_2)$$

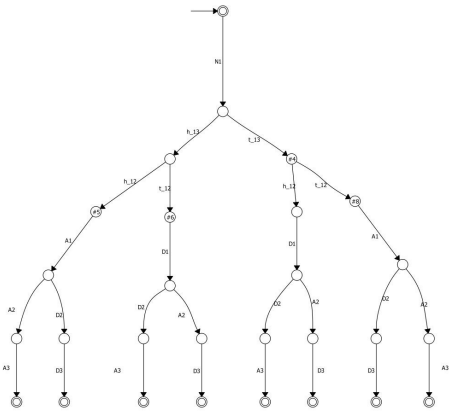
To see this, Figure 6.7 shows $L_2$ and Figure 6.13 shows the calculated $\theta_3(L_1) = \theta_3(L_2)$. Similarly to the previous cases, cryptographer$_1$ (with respect to $\theta_1$), $L_3$ is strongly opaque with respect to $L_2$ and $L_2$ is strongly opaque with respect to $L_3$, that is

$$\theta_1(L_3) \subseteq \theta_1(L_2) \wedge \theta_1(L_2) \subseteq \theta_1(L_3)$$
$$\Leftrightarrow \theta_1(L_3) = \theta_1(L_2)$$

To see this, Figure 6.6 shows $L_1$ and Figure 6.14 shows the calculated $\theta_1(L_3) = \theta_1(L_2)$.



(a)



(b)

Figure 6.11: (a) Automaton for $\theta_1(L)$, (b)Automaton for $\theta_1(K)$ clearly $\theta_1(L) \cap \theta_1(K) = \emptyset$

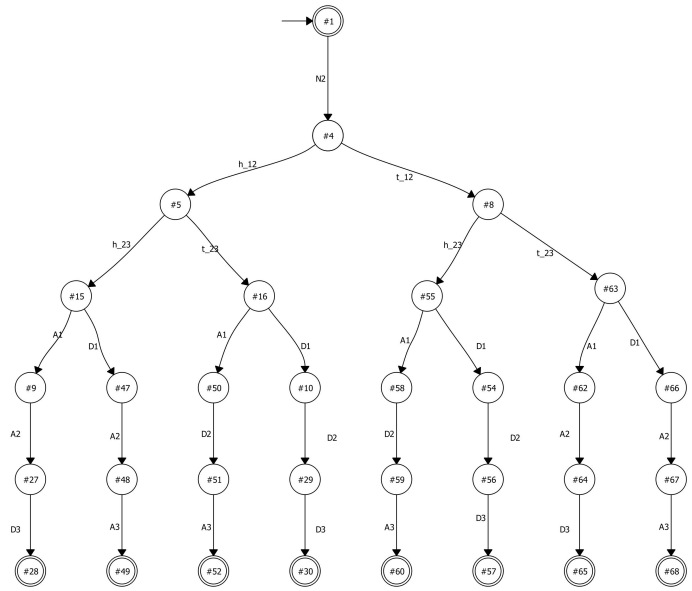



Figure 6.12: Automaton for both $\theta_2(L_1)$ and $\theta_2(L_3)$

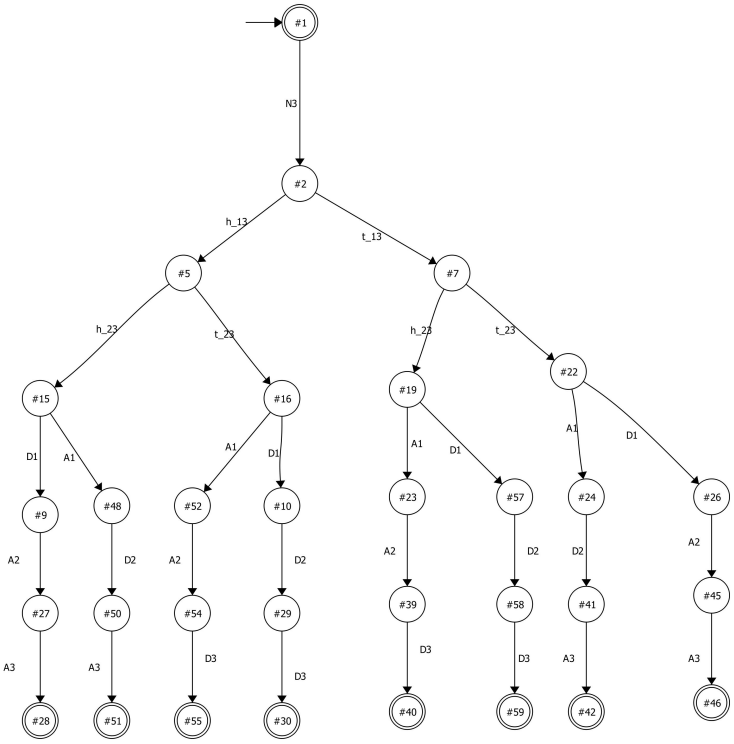



Figure 6.13: Automaton for both $\theta_3(L_1)$ and $\theta_3(L_2)$

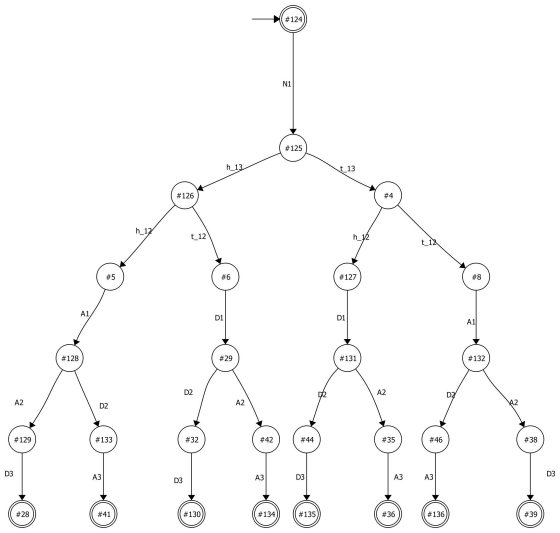



Figure 6.14: Automaton for both $\theta_1(L_3)$ and $\theta_1(L_2)$

## 6.4 Synthesis

The above verifies the dinning cryptographers protocol. Let us now try to see if we can synthesize the protocol. Synthesize is much more difficult than verification. We only manage to solve half of the problem. That is given $K_i$, $i = 1, 2, 3$, we can synthesize $L_i, i = 1, 2, 3$ using formulas for decentralized strong opacity and decentralized no opacity discussed in Chapter 5.

Since it is known that if one of the cryptographers is paying then the others are not paying. This leads us to drive the initial automata of $L_1, L_2$ and $L_3$. Let the initial language for $L_1$ be $\hat{L_1}$ which is given in Figure 6.15. The initial languages $\hat{L_2}$ and $\hat{L_3}$ are given in Figure 6.16 and in Figure 6.17 respectively. $L = (L_1 \cup L_2 \cup L_3)$ must be not opaque with respect to $K = K_1 \cap K_2 \cap K_3$.

Using the formula given in the previous section, we have

$$\begin{aligned} supDN_K^{sub}(\hat{L_i}) &= \hat{L_i} - (\theta_1^{-1}\theta_1(K) \cup \theta_2^{-1}\theta_2(K) \cup \theta_3^{-1}\theta_3(K)) \\ &= \hat{L_i} - \bigcup_{i=1}^{3} (\theta_i^{-1}\theta_i(K) \cap \mathcal{L}(G)) \end{aligned}$$

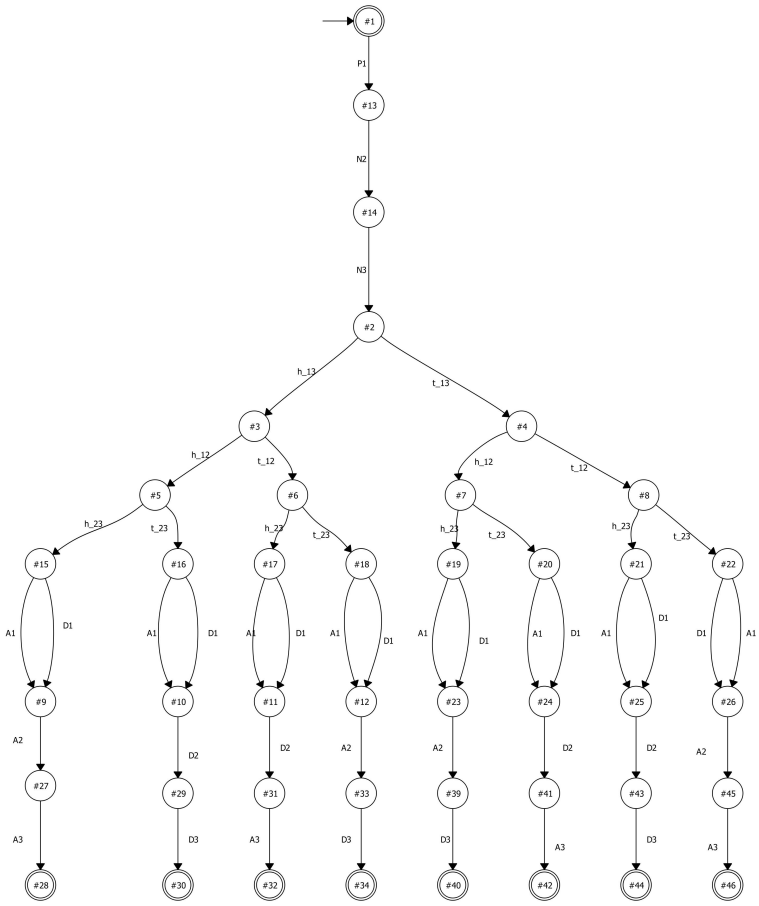



Figure 6.15: The Automaton for the initial $L_1$

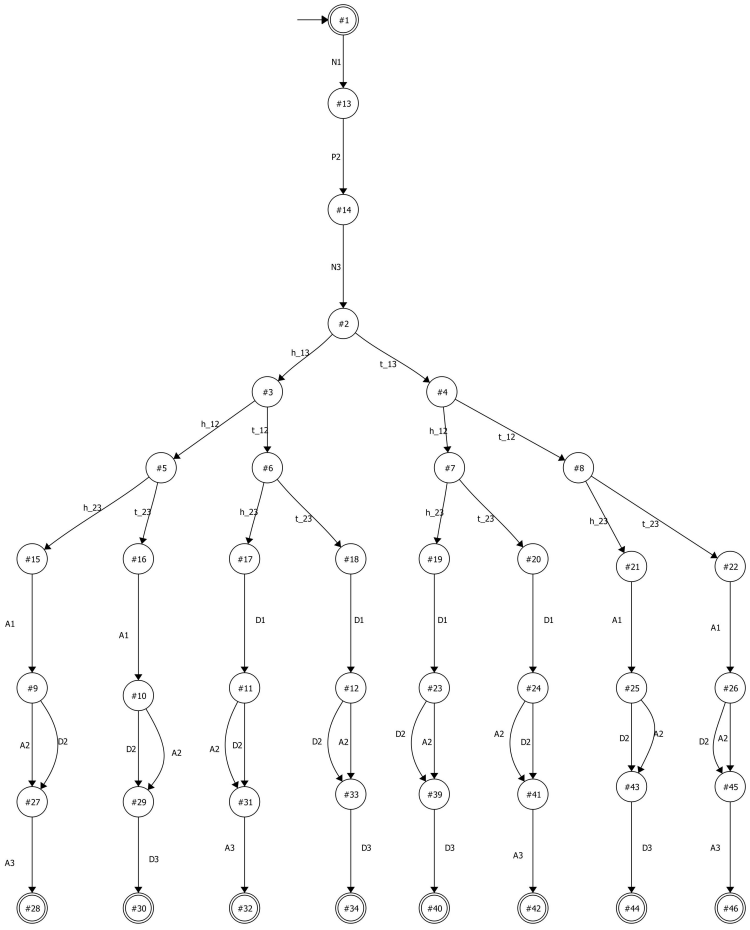



Figure 6.16: The Automaton for the initial $L_2$

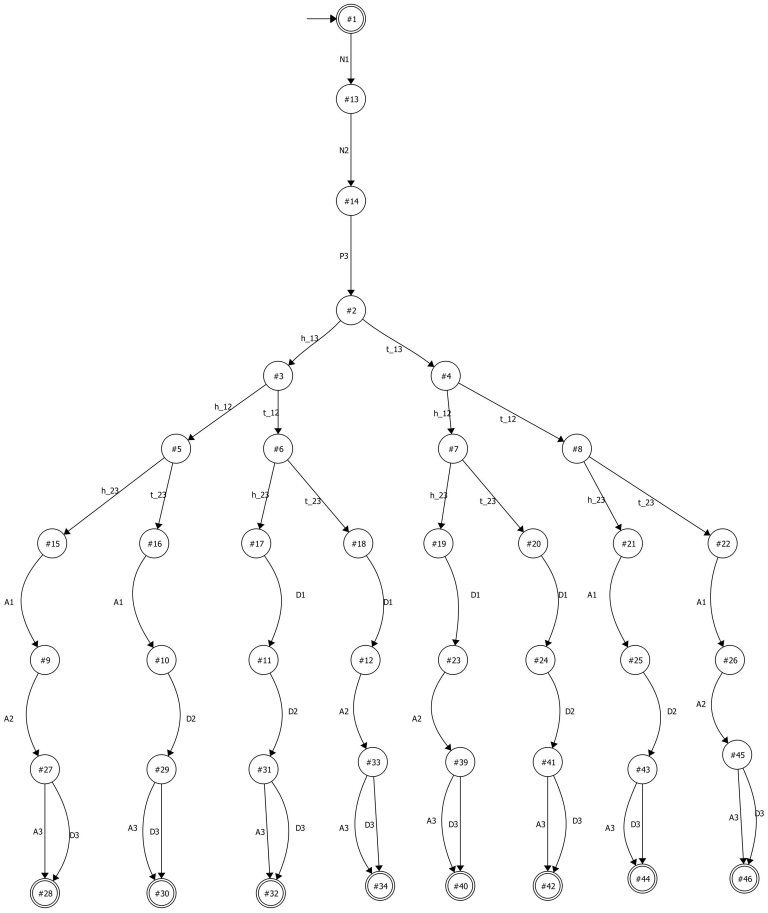



Figure 6.17: The Automaton for the initial $L_3$

$\theta_1(K)$ is shown in Figure 6.11(b) and $\theta_1^{-1}\theta_1(K) \cap \mathcal{L}(G)$ is shown in Figure 6.18. $\theta_2^{-1}\theta_2(K) \cap \mathcal{L}(G)$ and $\theta_3^{-1}\theta_3(K) \cap \mathcal{L}(G)$ can be computed similarly.

Using formula

$$supDN_K^{sub}(\hat{L_1}) = \hat{L_1} - \bigcup_{i=1}^{3}(\theta_i^{-1}\theta_i(K) \cap \mathcal{L}(G))$$

We have $supDN_K^{sub}(\hat{L_1}) = \tilde{L_1}$, which is shown in Figure 6.19. Similarly, we can show $supDN_K^{sub}(\hat{L_2}) = \tilde{L_2}, supDN_K^{sub}(\hat{L_3}) = \tilde{L_3}$. similarly, to $supDN_K^{sub}(\hat{L_1})$, $supDN_K^{sub}(\hat{L_2})$ and $supDN_K^{sub}(\hat{L_3})$ are given by:

$$\begin{aligned} supDN_K^{sub}(\hat{L_2}) &= \hat{L_2} - (\theta_1^{-1}\theta_1(K) \cup \theta_2^{-1}\theta_2(K) \cup \theta_3^{-1}\theta_3(K)) \\ &= \hat{L_2} - \bigcup_{i=1}^{3}(\theta_i^{-1}\theta_i(K) \cap \mathcal{L}(G)) \\ supDN_K^{sub}(\hat{L_3}) &= \hat{L_3} - (\theta_1^{-1}\theta_1(K) \cup \theta_2^{-1}\theta_2(K) \cup \theta_3^{-1}\theta_3(K)) \\ &= \hat{L_3} - \bigcup_{i=1}^{3}(\theta_i^{-1}\theta_i(K) \cap \mathcal{L}(G)) \end{aligned}$$

Clearly from the formalization of the protocol, $L_i$ shall satisfy no-opacity and strong opacity properties. Now we know that no-opacity is satisfied for all $L_i$ by synthesization of $\tilde{L_i}$. The second step is the synthesization of $L_i$ from $\tilde{L_i}$ such that strong opacity is satisfied. That is $\theta_2(L_1) = \theta_2(L_3)$, $\theta_1(L_2) = \theta_1(L_3)$ and $\theta_3(L_1) = \theta_3(L_2)$.

To achieve the equality $\theta_2(L_1) = \theta_2(L_3)$ we reduce $\tilde{L_1}$ to $L_1$ and $\tilde{L_3}$ to $L_3$ using formula

$$\begin{aligned} sup_{\tilde{L_1}}^{sub}(\tilde{L_1}) &= \tilde{L_3} \cap \theta_2^{-1}\theta_2(\tilde{L_1}) \\ sup_{\tilde{L_3}}^{sub}(\tilde{L_3}) &= \tilde{L_1} \cap \theta_2^{-1}\theta_2(\tilde{L_3}) \end{aligned}$$

$sup_{\tilde{L_1}}^{sub}(\tilde{L_1}) = L_1$ is shown in Figure 6.19. Similarly, we have $sup_{\tilde{L_3}}^{sub}(\tilde{L_3}) = L_3$. We could now check that strong opacity is satisfied by showing $\theta_2(L_1) = \theta_2(L_3)$. Figure 6.20 shows that $\theta_2(L_1) = \theta_2(L_3)$.

For the equality $\theta_1(L_2) = \theta_1(L_3)$ we reduce $\tilde{L_2}$ to $L_2$ and $\tilde{L_3}$ to $L_3$ using formula

$$\begin{aligned} sup_{\tilde{L_2}}^{sub}(\tilde{L_2}) &= \tilde{L_3} \cap \theta_1^{-1}\theta_1(\tilde{L_2}) \\ sup_{\tilde{L_3}}^{sub}(\tilde{L_3}) &= \tilde{L_2} \cap \theta_1^{-1}\theta_1(\tilde{L_3}) \end{aligned}$$

$sup^{sub}_{\tilde{L_2}}(\tilde{L_2}) = L_2$ is shown in Figure 6.21. Similarly, we have $sup^{sub}_{\tilde{L_3}}(\tilde{L_3}) = L_3$. We could now check that strong opacity is satisfied by showing $\theta_1(L_2) = \theta_1(L_3)$. Figure 6.22 shows that $\theta_1(L_2) = \theta_1(L_3)$.

For the equality $\theta_3(L_1) = \theta_3(L_2)$ we reduce $\tilde{L_1}$ to $L_1$ and $\tilde{L_2}$ to $L_2$ using formula

$$\begin{aligned} sup^{sub}_{\tilde{L_1}}(\tilde{L_1}) &= \tilde{L_1} \cap \theta_3^{-1}\theta_3(\tilde{L_2}) \\ sup^{sub}_{\tilde{L_2}}(\tilde{L_2}) &= \tilde{L_2} \cap \theta_3^{-1}\theta_3(\tilde{L_1}) \end{aligned}$$

$sup^{sub}_{\tilde{L_1}}(\tilde{L_1}) = L_1$ is shown in Figure 6.19. Similarly, we have $sup^{sub}_{\tilde{L_2}}(\tilde{L_2}) = L_2$. We could now check that strong opacity is satisfied by showing $\theta_3(L_1) = \theta_3(L_2)$. Figure 6.23 shows that $\theta_3(L_1) = \theta_3(L_2)$. This proves the anonymity of the payer in case of one of the cryptographers is the payer, and also proves that if the NSA is the payer then the cryptographers will find out.

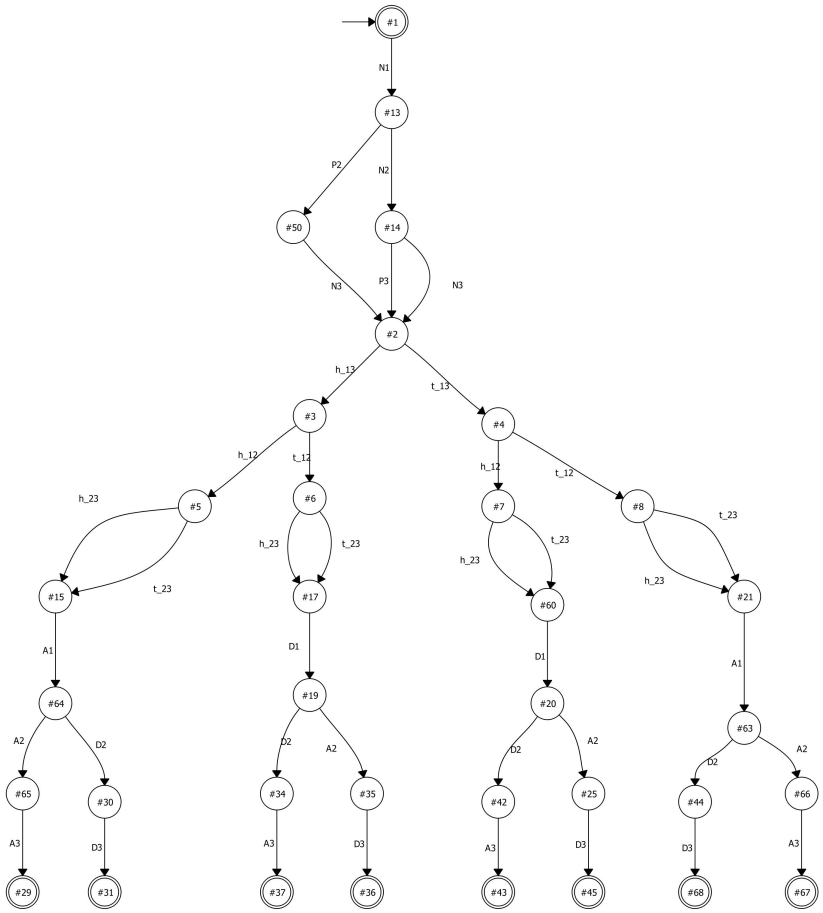



Figure 6.18: Automaton for $\theta_1^{-1}\theta_1(K) \cap \mathcal{L}(G)$

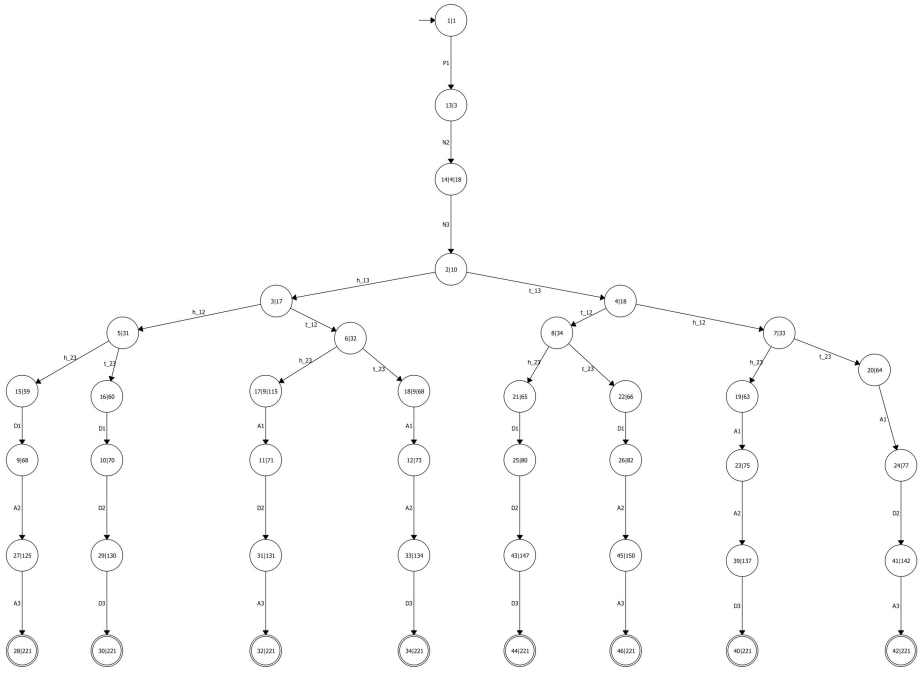



Figure 6.19: Automaton for $\tilde{L_1} = L_1$

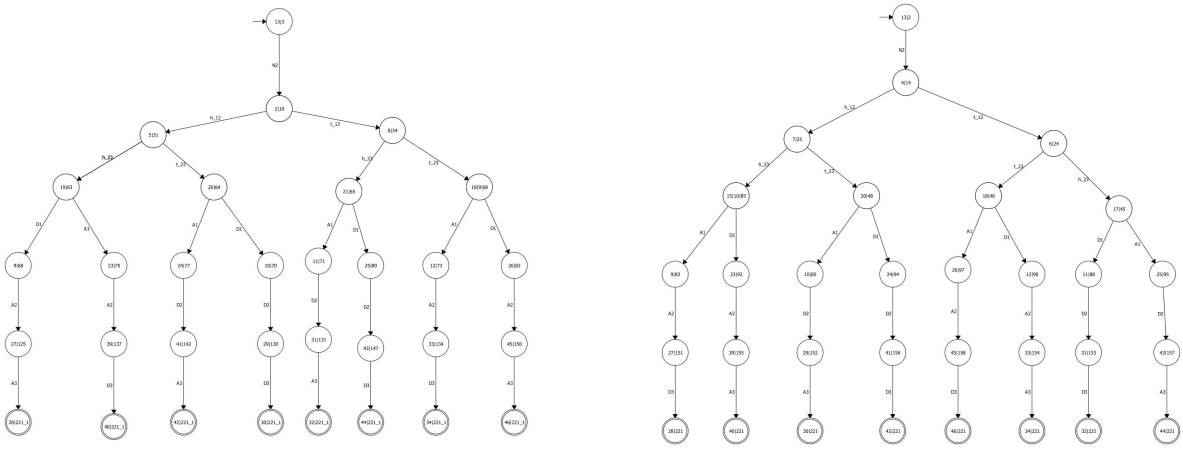



Figure 6.20: Automata for $\theta_2(L_1)$ and $\theta_2(L_3)$

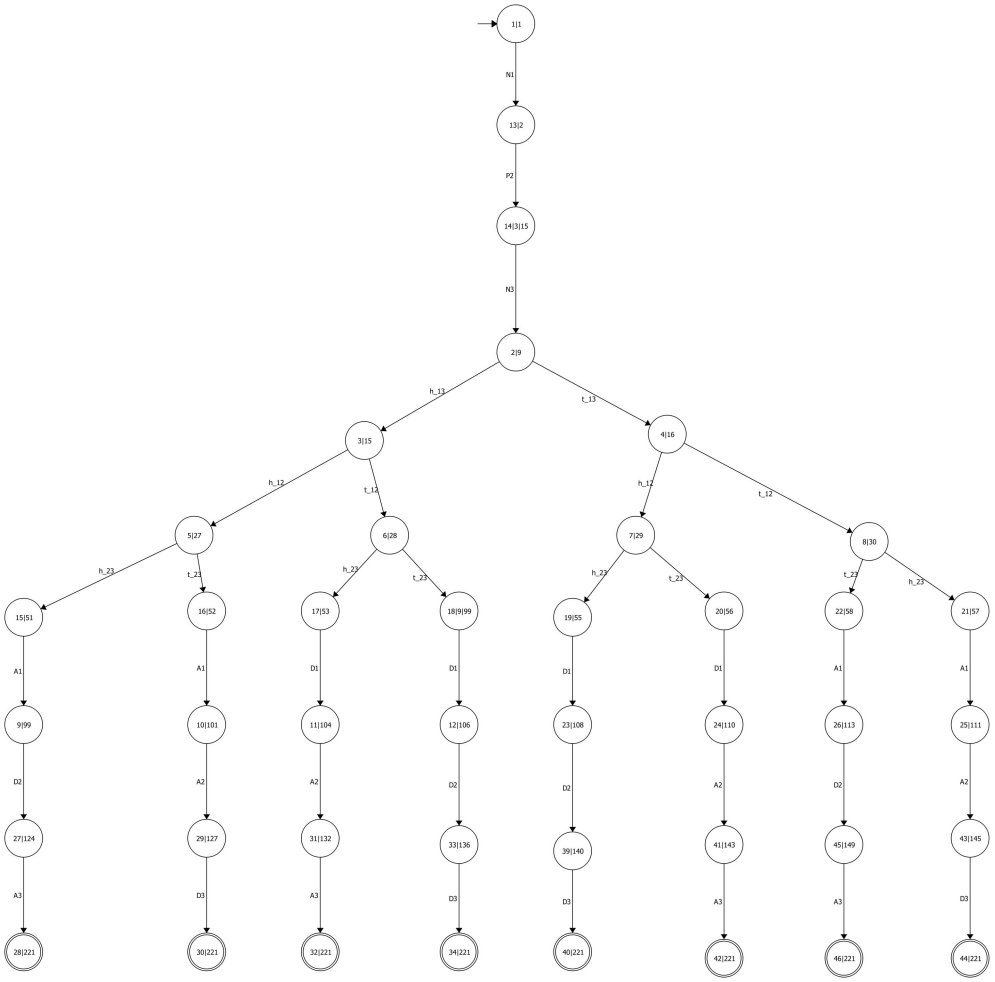



Figure 6.21: Automaton for $\tilde{L_2} = L_2$

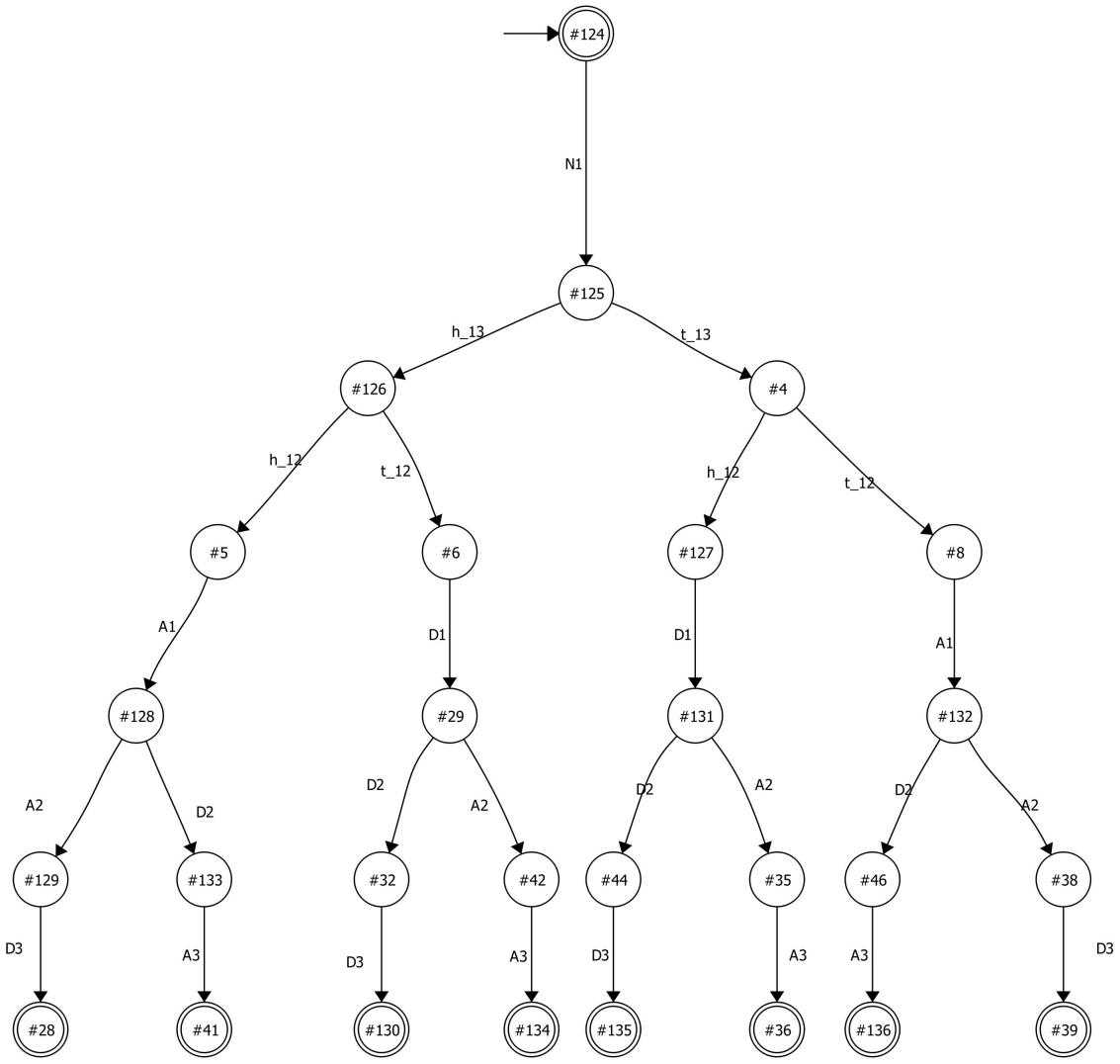



Figure 6.22: Automata for $\theta_1(L_2)$ and $\theta_1(L_3)$

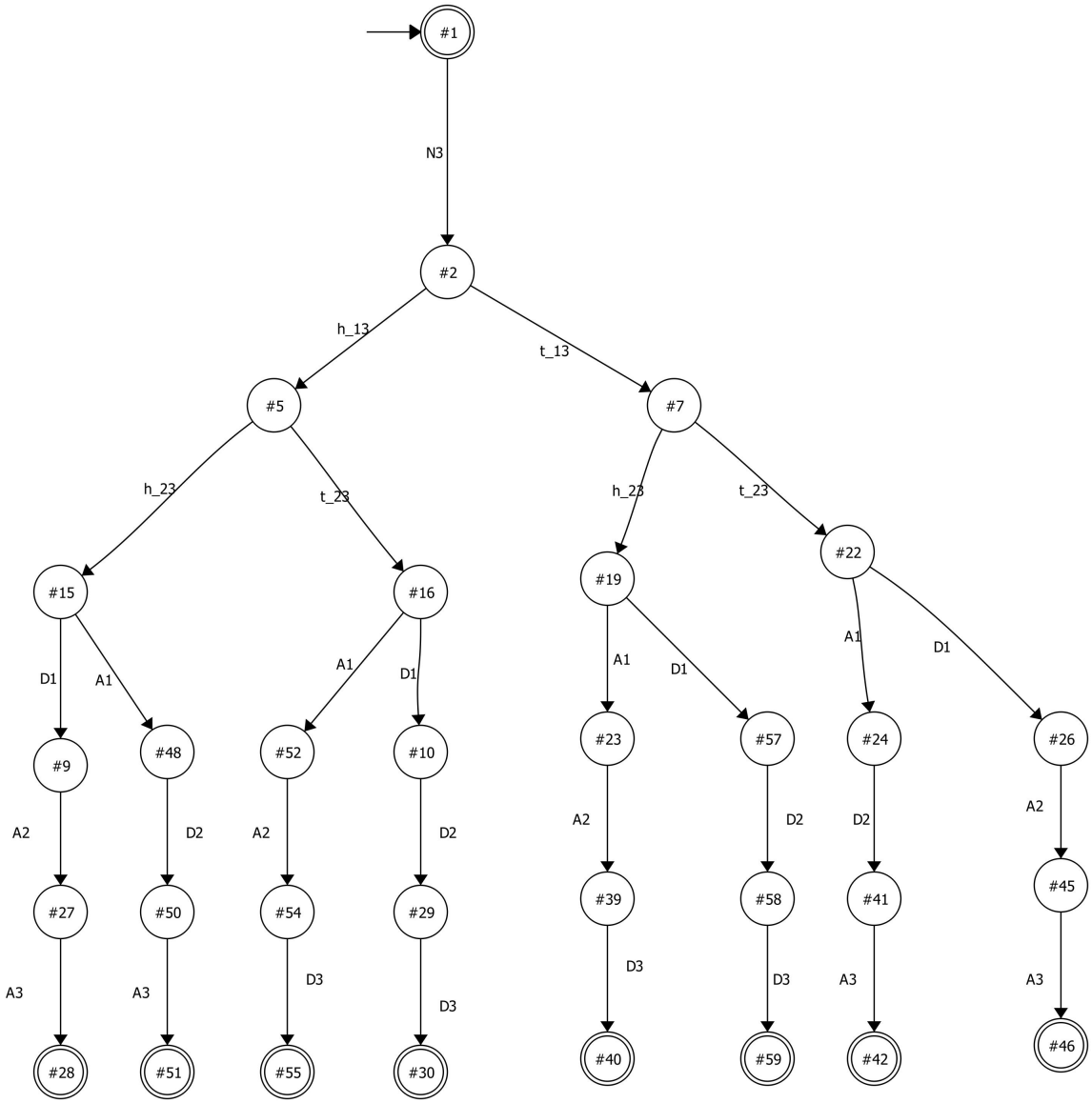



Figure 6.23: Automata for $\theta_3(L_1)$ and $\theta_3(L_2)$

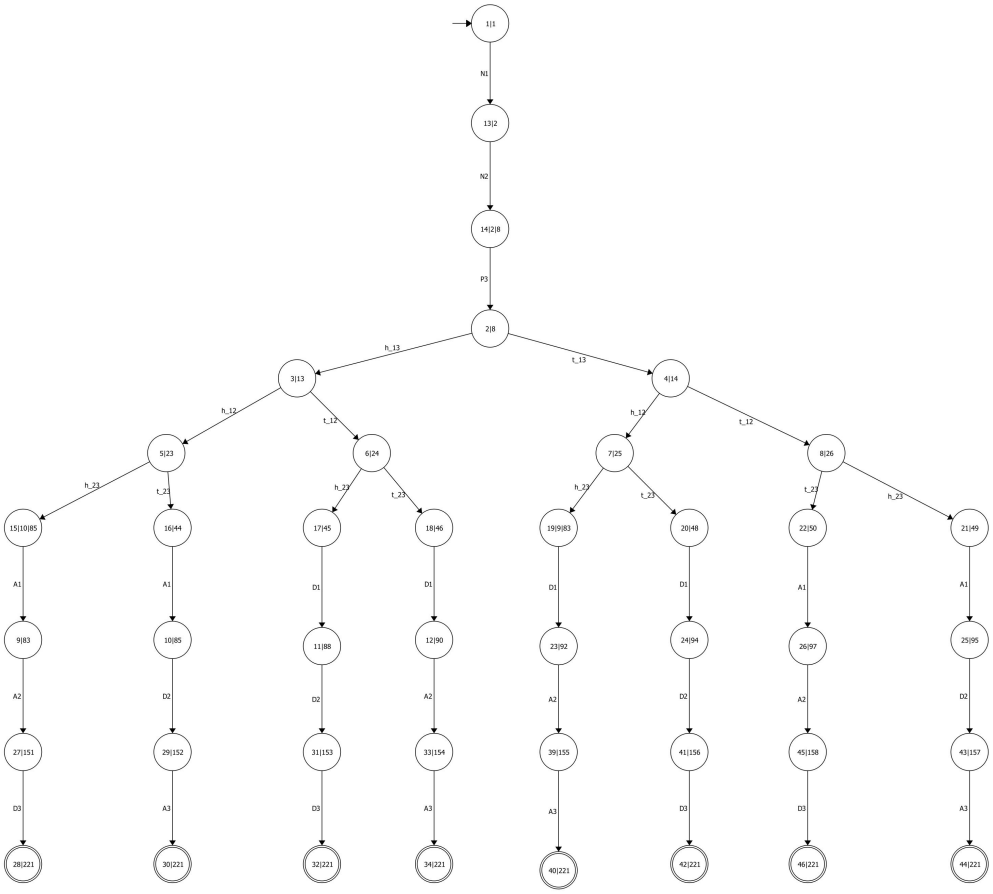



Figure 6.24: Automaton for $\tilde{L_3} = L_3$

# 7 Supervisory Control for Opacity

The supervisory control problem for opacity consists of designing a supervisor that restrict the activities of the system such that opacity properties are satisfied. Therefore the goal in this chapter is to investigate properties of opacity under controllability restriction. Based on theories presented in [45] and [30], the objective of supervisory control is to enforce a specific property on a discrete event system to achieve a desired behaviour. In this framework some of the events called controllable events, that can be disabled, and the control problem is to suitably interact with the process, by disabling of controllable events, so the desired behaviour is within the specification.

In this chapter, we use supervisory control to achieve opacity. The idea is that if a system does not satisfy opacity, then we investigate if a supervisor can be used to restrict the system's behavior so that the supervised system satisfies opacity. We assume that the supervisor is internal to the system and hence can observe events that may not be observable to external observers. The supervisor can disable some controllable events to restrict the behavior of the system. From [31], we know that such a supervisor exists if and only if the desired closed-loop language is controllable and observable. From [32], we know that if all controllable events are observable to the supervisor, then observability can be replaced by normality.

In this context we characterize the solution for the supervisory control problem for strong opacity in terms of the largest sublanguage or smallest superlanguage that satisfies controllability, observability and strong opacity. Similar characterization for supervisory control problem for weak opacity, and no opacity is investigated.

## 7.1 System Description

The formulation of the supervisory control problem considered in this section is as follows. Given a finite deterministic system $G$, two regular languages $L, K \subseteq \mathcal{L}(G)$, and

an observation mapping $\theta$ based on the set of events that can be observed by an external observer, $\Sigma_{ex,o} \subseteq \Sigma$. If $L$ and $K$ do not satisfy opacity properties, we would like to design a controller or supervisor, if possible, to restrict the behavior of $G$ so that opacity properties are satisfied. The supervisor can disable a subset of controllable events $\Sigma_c \subseteq \Sigma$. We assume that the supervisor is internal to the system, so that it can observe more events than the external observer. The set of event that can be observed by the supervisor is denoted by $\Sigma_{in,o} \subseteq \Sigma$. We assume $\Sigma_{ex,o} \subseteq \Sigma_{in,o}$ and the equality may not hold in general.

Our goal is to enforce opacity properties of $L$ and $K$ using a supervisor $S$. The language generated by the supervised system is denoted by $\mathcal{L}(S/G)$. Under the control of $S$, $L$ and $K$ are restricted to $L \cap \mathcal{L}(S/G)$ and $K \cap \mathcal{L}(S/G)$ respectively. We want $S$ to be as less restrictive as possible. Therefore, the goal is equivalent to find a largest language $M = \mathcal{L}(S/G)$ such that $L \cap M$ and $K \cap M$ satisfy opacity properties.

It is shown by Lin and Wonham [31] that a supervisor exists such that $M = \mathcal{L}(S/G)$ if and only if $M$ is controllable and observable. Therefore, we want to find the largest sublanguage $M$ of $\mathcal{L}(G)$ such that $M$ is controllable w.r.t. $\mathcal{L}(G)$ and $\Sigma_c$, and observable w.r.t. $\mathcal{L}(G)$ and $\Sigma_{in,o}$, and furthermore, $L \cap M$ and $K \cap M$ satisfy opacity properties.

Based on the properties of opacity studied in Chapter 3, let us consider the strong opacity control problem (SOCP), the weak opacity control problem (WOCP), and no opacity control problem (NOCP), respectively.

## 7.2 Strong Opacity Control Problem (SOCP)

If $L \subseteq \mathcal{L}(G)$ is not strongly opaque with respect to $K \subseteq \mathcal{L}(G)$, that is $\theta(L) \not\subseteq \theta(K)$, then we want to design a controller $S$ to restrict the language of the system from $\mathcal{L}(G)$ to $\mathcal{L}(S/G) = M$ such that in the closed-loop or the controlled system, strong opacity holds, that is, $\theta(M \cap L) \subseteq \theta(M \cap K)$. Therefore, our goal is to find a language $M \subseteq \mathcal{L}(G)$ such that

1. $M$ is controllable w.r.t. $\Sigma_c$ and $\mathcal{L}(G)$.

2. $M$ is observable w.r.t. $\Sigma_{in,o}$ and $\mathcal{L}(G)$.

3. $M \cap L$ is strongly opaque w.r.t. $M \cap K$, that is, $\theta(M \cap L) \subseteq \theta(M \cap K)$.

4. $M$ is as large as possible, that is, for any other language $\acute{M} \subseteq \mathcal{L}(G)$ satisfying the above three conditions, $M \not\subset \acute{M}$.

Let us focus on strong opacity first. Our first objective is to find $M$ satisfying Conditions 3) and 4). Our solution is inspired by Chapter 4, where the goal is to find the largest sublanguage of $L$ to ensure opacity. Obviously, our goal here is different: Here we want to find the largest sublanguage of $\mathcal{L}(G)$ to ensure opacity. Intuitively to find such a sublanguage, we need to remove all bad strings $\theta^{-1}(\theta(L) - \theta(K))$ from $\mathcal{L}(G)$ that are in $L$. Therefore, we have the following theorem.

**Theorem 7.2.1.** *Given two languages $L, K \subseteq \mathcal{L}(G)$ . Let*

$$M = \mathcal{L}(G) - (\theta^{-1}(\theta(L) - \theta(K)) \cap L).$$

*Then $M$ satisfies the following conditions: (1) $M \cap L$ is strongly opaque w.r.t. $M \cap K$ and $\theta$, that is $\theta(M \cap L) \subseteq \theta(M \cap K)$. (2) $M$ is the largest sublanguage of $\mathcal{L}(G)$ satisfying (1), that is, for any other $\acute{M} \subseteq \mathcal{L}(G)$ satisfying (1), $\acute{M} \subseteq M$.*

*Proof.* We first show that $M \cap L$ is strongly opaque w.r.t $M \cap K$, that is, $\theta(M \cap L) \subseteq \theta(M \cap K)$. This can be done as follows.

$$
\begin{aligned}
& s \in \theta(M \cap L) \\
\Rightarrow\ & s \in \theta((\mathcal{L}(G) - (\theta^{-1}(\theta(L) - \theta(K)) \cap L)) \cap L) \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta(t) = s \wedge t \in ((\mathcal{L}(G) - (\theta^{-1}(\theta(L) - \theta(K)) \cap L)) \cap L) \wedge t \in M \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta(t) = s \wedge t \in (\mathcal{L}(G) - (\theta^{-1}(\theta(L) - \theta(K)) \cap L)) \wedge t \in L \wedge t \in M \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta(t) = s \wedge t \in \mathcal{L}(G) \wedge t \notin (\theta^{-1}(\theta(L) - \theta(K)) \cap L) \wedge t \in L \wedge t \in M \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta(t) = s \wedge t \in \mathcal{L}(G) \wedge t \notin \theta^{-1}(\theta(L) - \theta(K)) \wedge t \in L \wedge t \in M \\
\Rightarrow\ & (\exists t \in \Sigma^*)\theta(t) = s \wedge t \in \mathcal{L}(G) \wedge \theta(t) \notin (\theta(L) - \theta(K)) \wedge t \in L \wedge t \in M
\end{aligned}
$$

$\Rightarrow (\exists t \in \Sigma^*)\theta(t) = s \wedge t \in \mathcal{L}(G) \wedge (\theta(t) \notin \theta(L) \vee \theta(t) \in \theta(K)) \wedge t \in L \wedge t \in M$

$\Rightarrow (\exists t \in \Sigma^*)\theta(t) = s \wedge t \in \mathcal{L}(G) \wedge t \in L \wedge t \in M \wedge \theta(t) \in \theta(K)$

$\Rightarrow (\exists t \in \Sigma^*)\theta(t) = s \wedge t \in \mathcal{L}(G) \wedge t \in L \wedge t \in M \wedge (\exists \acute{t} \in \Sigma^*)\theta(\acute{t}) = \theta(t) \wedge \acute{t} \in K \wedge \acute{t} \in \mathcal{L}(G)$

$\Rightarrow (\exists \acute{t} \in \Sigma^*)\theta(\acute{t}) = s \wedge \acute{t} \in \mathcal{L}(G) \wedge \acute{t} \in K$

$\Rightarrow (\exists \acute{t} \in \Sigma^*)\theta(\acute{t}) = s \wedge \acute{t} \in \mathcal{L}(G) \wedge \acute{t} \in K \wedge \theta(\acute{t}) \in \theta(K)$

$\Rightarrow (\exists \acute{t} \in \Sigma^*)\theta(\acute{t}) = s \wedge \acute{t} \in \mathcal{L}(G) \wedge \acute{t} \in K \wedge \theta(\acute{t}) \notin \theta(L) - \theta(K)$

$\Rightarrow (\exists \acute{t} \in \Sigma^*)\theta(\acute{t}) = s \wedge \acute{t} \in \mathcal{L}(G) \wedge \acute{t} \in K \wedge \acute{t} \notin \theta^{-1}(\theta(L) - \theta(K))$

$\Rightarrow (\exists \acute{t} \in \Sigma^*)\theta(\acute{t}) = s \wedge \acute{t} \in \mathcal{L}(G) \wedge \acute{t} \in K \wedge \acute{t} \notin \theta^{-1}(\theta(L) - \theta(K)) \cap L$

$\Rightarrow (\exists \acute{t} \in \Sigma^*)\theta(\acute{t}) = s \wedge \acute{t} \in M \wedge \acute{t} \in K$

$\Rightarrow (\exists \acute{t} \in \Sigma^*)\theta(\acute{t}) = s \wedge \acute{t} \in M \cap K$

$\Rightarrow s \in \theta(M \cap K)$.

Next, we show that $M$ is the largest language such that $\theta(M \cap L) \subseteq \theta(M \cap K)$. Assume the contrary, that is,

$(\exists \acute{M} \in \mathcal{L}(G))\theta(\acute{M} \cap L) \subseteq \theta(\acute{M} \cap K)$, and $M \subset \acute{M}$.

Then

$(\exists t \in \mathcal{L}(G))$ such that $t \in \acute{M} \wedge t \notin M$.

We have,

$t \in \acute{M} \wedge t \notin M$

$\Rightarrow t \in \acute{M} \wedge t \notin \mathcal{L}(G) - \theta^{-1}(\theta(L) - \theta(K)) \cap L)$

$\Rightarrow t \in \acute{M} \wedge t \in \theta^{-1}(\theta(L) - \theta(K)) \cap L)$

$\Rightarrow t \in \acute{M} \wedge t \in \theta^{-1}(\theta(L) - \theta(K)) \wedge t \in L$

$\Rightarrow t \in \acute{M} \wedge \theta(t) \in (\theta(L) - \theta(K))$

$\Rightarrow t \in (\acute{M} \cap L) \wedge \theta(t) \in \theta(L) \wedge \theta(t) \notin \theta(K)$

$\Rightarrow t \in (\acute{M} \cap L) \wedge \theta(t) \in \theta(L) \wedge \theta(t) \notin \theta(\acute{M} \cap K)$

a contradiction! Therefore, no such $\acute{M}$ exists.

□

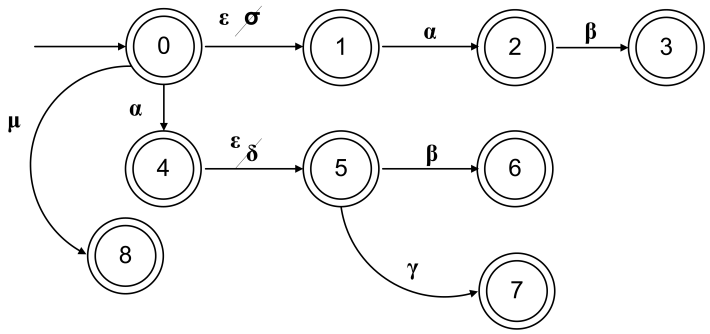



*Figure 7.1: System G and observation mapping θ used in Example 7.2.1*

Let us now consider controllability and observability. If $M$ obtained in Theorem 7.2.1 is not controllable and observable, then we need further reducing $M$ to make it controllable and observable. Unfortunately, reducing $M$ may violate strong opacity. This is because $\theta(M \cap L) \subseteq \theta(M \cap K)$ and $M' \subseteq M$ does not imply $\theta(M' \cap L) \subseteq \theta(M' \cap K)$.

**Example 7.2.1.** *We consider the system G in Figure 7.1. where* $\Sigma = \{\alpha, \beta, \sigma, \delta, \gamma, \mu\}$. *Let the observation mapping* $\theta$ *be defined as illustrated in the figure, where the unobservable transitions are replaced by* $\epsilon$. *Let* $\Sigma_c = \{\alpha, \mu, \sigma, \delta\}$, $\Sigma_{in,o} = \Sigma$, *and* $\Sigma_{ex,o} = \{\alpha, \beta, \gamma, \mu\}$. *Let* $L = \overline{\sigma\alpha\beta} + \overline{\alpha\delta\gamma}$ *and* $K = \overline{\alpha\delta\beta} + \mu$. *The observation mappings of L and K are* $\theta(L) = \overline{\alpha\beta} + \overline{\alpha\gamma}$ *and* $\theta(K) = \overline{\alpha\beta} + \mu$ *respectively. L is not strongly opaque with respect to K and* $\theta$, *because* $\theta(L) \nsubseteq \theta(K)$. *By Theorem 7.2.1, we have*
$M = \overline{\sigma\alpha\beta} + \overline{\alpha\delta\beta} + \overline{\mu}$
*Clearly* $\theta(M \cap L) \subseteq \theta(M \cap K)$. *M is not controllable because* $\gamma$ *is not controllable. Let* $M' = \overline{\sigma\alpha\beta} + \overline{\alpha} + \overline{\mu}$, *then M′ is controllable, M' is also observable because* $\Sigma_{in,o} = \Sigma$.
*It is not difficult to see* $\theta(M' \cap L) \nsubseteq \theta(M' \cap K)$.

From the above example, we know that we may need to repeat the process of reducing a language to satisfy controllability, observability, and strong opacity. For the convenience of iteration, let us define an operator to ensure opacity on languages as follow. For a language

$H \subseteq \mathcal{L}(G)$, define

$$H^{\uparrow(K,L)} = H - (\theta^{-1}(\theta(H \cap L) - \theta(H \cap K)) \cap L).$$

Then $H^{\uparrow(K,L)}$ has the following properties.

**Proposition 7.2.1.**

*(1)* $H^{\uparrow(K,L)}$ *ensures strong opacity, that is,*

$$\theta(H^{\uparrow(K,L)} \cap L) \subseteq \theta(H^{\uparrow(K,L)} \cap K).$$

*(2)* $H^{\uparrow(K,L)}$ *is the largest such sublanguage, that is,*

$$(\forall \acute{H} \subseteq H)\theta(\acute{H} \cap L) \subseteq \theta(\acute{H} \cap K) \Rightarrow \acute{H} \subseteq H^{\uparrow(K,L)}.$$

*(3)*$(.)^{\uparrow(K,L)}$ *is a monotonic operator, that is,*

$$H \subseteq \acute{H} \Rightarrow H^{\uparrow(K,L)} \subseteq \acute{H}^{\uparrow(K,L)}.$$

*Proof.* (1) The proof of this part is similar to that of Theorem 7.2.1.

(2) The proof of this part is similar to that of Theorem 7.2.1.

(3) We need to prove that if $H \subseteq \acute{H}$, then

$$\begin{aligned} & H - \theta^{-1}(\theta(H \cap L) - \theta(H \cap K)) \cap L \\ \subseteq \; & \acute{H} - \theta^{-1}(\theta(\acute{H} \cap L) - \theta(\acute{H} \cap K)) \cap L. \end{aligned}$$

Indeed,

$$\begin{aligned} & t \in H - \theta^{-1}(\theta(H \cap L) - \theta(H \cap K)) \cap L \\ \Rightarrow\; & t \in H \wedge t \notin (\theta^{-1}(\theta(H \cap L) - \theta(H \cap K)) \cap L) \\ \Rightarrow\; & t \in H \wedge (t \notin \theta^{-1}(\theta(H \cap L) - \theta(H \cap K)) \vee t \notin L) \\ \Rightarrow\; & t \in H \wedge (\theta(t) \notin (\theta(H \cap L) - \theta(H \cap K)) \vee t \notin L) \\ \Rightarrow\; & t \in H \wedge (\theta(t) \notin \theta(H \cap L) \vee \theta(t) \in \theta(H \cap K) \vee t \notin L) \\ \Rightarrow\; & t \in \acute{H} \wedge (\theta(t) \notin \theta(\acute{H} \cap L) \vee \theta(t) \in \theta(\acute{H} \cap K) \vee t \notin L) \end{aligned}$$

$\Rightarrow t \in \acute{H} \wedge (\theta(t) \notin (\theta(\acute{H} \cap L) - \theta(\acute{H} \cap K)) \vee t \notin L)$

$\Rightarrow t \in \acute{H} \wedge (t \notin \theta^{-1}(\theta(\acute{H} \cap L) - \theta(\acute{H} \cap K)) \vee t \notin L)$

$\Rightarrow t \in \acute{H} \wedge t \notin (\theta^{-1}(\theta(\acute{H} \cap L) - \theta(\acute{H} \cap K)) \cap L)$

$\Rightarrow t \in \acute{H} - \theta^{-1}(\theta(\acute{H} \cap L) - \theta(\acute{H} \cap K)) \cap L.$

□

With this operator, we will consider controllability and observability. We will consider two cases. The first case is when $\Sigma_c \subseteq \Sigma_{in,o}$. In this case, Lin and Wonham show that controllability and observability are equivalent to controllability and normality [32]. Therefore Condition 2) in the strong opacity control problem becomes

2') $M \subseteq \mathcal{L}(G)$ is normal w.r.t. $\Sigma_{in,o}$ and $\mathcal{L}(G)$.

The advantage of requiring normality is that Lin and Wonham [30] show that the supremal controllable and normal sublanguage of a given language exists and formulas to compute the supremal controllable and normal sublanguage are given in [43]. Therefore, if the language $M$ obtained in Theorem 7.2.1 is not controllable and normal, then we can compute its supremal controllable and normal sublanguage, denoted by $M^{\uparrow(CN)}$. Unfortunately, reducing $M$ to satisfy controllability and observability may violate strong opacity. This means that the process may need to be reiterated. Formally, the following algorithm computes a solution to the strong opacity control problem.

**Algorithm 7.2.1.** *(Solution to SOCP)*

*Step 1. Initially, set*

$i = 0$;

$M_0 = \mathcal{L}(G)$;

*Step 2. Set*

$M_{i+(\frac{1}{2})} = M_i^{\uparrow(K,L)}$;

*Step 3. Set*

$M_{i+1} = M_{i+(\frac{1}{2})}^{\uparrow(CN)}$;

*Step 4. If* $\theta(M_{i+1} \cap L) \subseteq \theta(M_{i+1} \cap K)$, *then stop; else set*

$i = i+1$;

*go to Step 2.*

The convergence of the above algorithm depends on the nature of the observation mapping and the languages involved. If $\mathcal{L}(G)$, $K$, and $L$ are all regular, and $\theta$ preserves regularity (that is, for any regular language $J$, $\theta(J)$ and $\theta^{-1}(J)$ are regular), then we can prove that the above algorithm converges. To this end, let us first prove the following proposition.

**Proposition 7.2.2.** *For the sequence*

$$M_0 = \mathcal{L}(G),$$
$$M_{i+1} = (M_i^{\uparrow(K,L)})^{\uparrow(CN)},$$

*if there exist a finite $k$ such that $M_{k+1} = M_k$, then for any $i > k$ $M_i = M_k$.*

*Proof.* Suppose that there exist a finite $k$ such that $M_{k+1} = M_k$, then for $i = k+2$,

$$M_{i+2} = (M_{k+1}^{\uparrow(K,L)})^{\uparrow(CN)} = (M_k^{\uparrow(K,L)})^{\uparrow(CN)} = M_{k+1} = M_k.$$

By repeating the above reasoning, we conclude that for any $i > k$, $M_i = M_k$.

□

We now prove the convergence of Algorithm 1.

**Theorem 7.2.2.** *If $\mathcal{L}(G)$, $K$, and $L$ are all regular, and $\theta$ preserves regularity, then Algorithm 1 converges.*

*Proof.* We prove the theorem by contradiction. If the sequence

$$M_0 = \mathcal{L}(G),$$
$$M_{i+1} = (M_i^{\uparrow(K,L)})^{\uparrow(CN)},$$

does not converge, then by Proposition 7.2.2, there exist infinite many distinct $M_i$ such that

$M_0 \supset M_1 \supset M_2 \supset ... \supset M_i \supset ....$

On the other hand, since $\mathcal{L}(G)$, $K$, and $L$ are all regular; $\theta$ preserves regularity; and all other operations in computing $^{\uparrow(K,L)}$ and $^{\uparrow(CN)}$ also preserves regularity, all $M_i$ are regular. Let $|M_i|$ denotes the number of states in the minimal automaton generating $M_i$. Then there exists an integer $N$ such that for all $M_i$, $|M_i| < N$. But there are only finite distinct $M_i$ such at $|M_i| < N$. This leads to a contradiction.

□

The following theorem shows the correctness of Algorithm 1.

**Theorem 7.2.3.** *Assume that $\Sigma_c \subseteq \Sigma_{in,o}$. Suppose Algorithm 1 converges after k steps, that is, $M_{k+1} = M_k$. Then the result obtained by Algorithm 1, $M = M_{k+1} = M_k$, is a solution to the SOCP. In other words, $M$ satisfies the following.*

1. *$M$ is controllable w.r.t. $\Sigma_c$ and $\mathcal{L}(G)$.*
2. *$M$ is normal w.r.t. $\Sigma_{in,o}$ and $\mathcal{L}(G)$.*
3. *$M \cap L$ is strongly opaque w.r.t. $M \cap K$, that is, $\theta(M \cap L) \subseteq \theta(M \cap K)$.*
4. *$M$ is largest, that is, for any other language $\acute{M} \subseteq \mathcal{L}(G)$ satisfying the above three conditions, $\acute{M} \subseteq M$.*

*Proof.* Since $M = (M^{\uparrow(K,L)})^{\uparrow(CN)}$, by properties of the supremal controllable and normal sublanguage, $M$ is controllable and normal.

Since $M = (M^{\uparrow(K,L)})^{\uparrow(CN)} \subseteq M^{\uparrow(K,L)} \subseteq M$, $M = M^{\uparrow(K,L)}$. By Proposition 7.2.1, $M \cap L$ is strongly opaque w.r.t. $M \cap K$.

We now prove that $M$ is largest, that is, for any language $\acute{M} \subseteq \mathcal{L}(G)$ satisfying the three conditions, $\acute{M} \subseteq M$.

By properties of $^{\uparrow(K,L)}$ and $^{\uparrow(CN)}$, if $\acute{M}$ satisfying the three conditions, then $\acute{M} = (\acute{M}^{\uparrow(K,L)})^{\uparrow(CN)}$.

$\acute{M} \subseteq \mathcal{L}(G)$ and $M_0 = \mathcal{L}(G)$ implies $\acute{M} \subseteq M_0$. Since both $^{\uparrow(K,L)}$ and $^{\uparrow(CN)}$ are monotonic,

$$\begin{aligned} & \acute{M} \subseteq M_0 \\ \Rightarrow \quad & (\acute{M}^{\uparrow(K,L)})^{\uparrow(CN)} \subseteq (M_0^{\uparrow(K,L)})^{\uparrow(CN)} = M_1 \\ \Rightarrow \quad & \acute{M} \subseteq M_1. \end{aligned}$$

By repeating the above reasoning, we conclude that $\acute{M} \subseteq M_k = M$.

□

**Example 7.2.2.** *We consider the system $G$ in Figure 7.2. Let $\Sigma_{in,o} = \Sigma = \{\alpha, \beta, \gamma, \sigma, \delta\}$, and $\Sigma_{ex,o} = \{\alpha, \beta, \gamma, \delta\}$. Let $\theta$ be defined as illustrated in the figure, where the unobservable transitions are replaced by $\epsilon$. Let $\Sigma_c = \{\beta, \delta\} \subseteq \Sigma_{o,in}$. Let*

$L = \overline{\sigma\alpha\beta\gamma}$; $\quad K = \overline{\alpha\beta}$.

*Then*

$\theta(L) = \overline{\alpha\beta\gamma}$; $\quad \theta(K) = \overline{\alpha\beta}$.

*$L$ is not strongly opaque with respect to $K$ because $\theta(L) \nsubseteq \theta(K)$. Apply Algorithm 1, we have*

$M_0 = \mathcal{L}(G)$;

$M_1 = \overline{\alpha\beta} + \overline{\sigma\alpha\delta\beta}$;

$M_2 = M_1$;

*Therefore,*

$M = \overline{\alpha\beta} + \overline{\sigma\alpha\delta\beta}$.

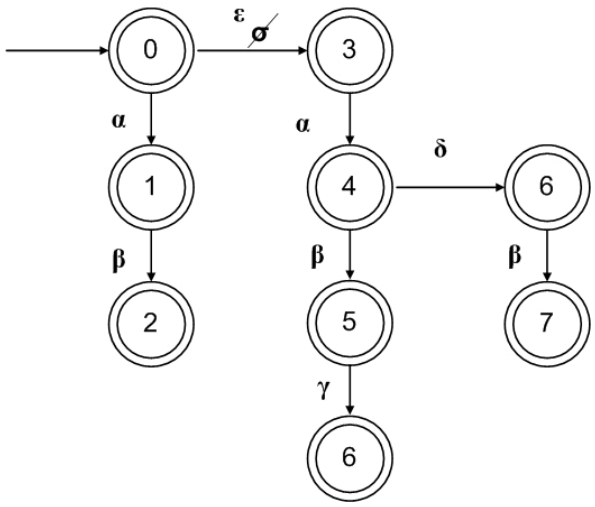



*Figure 7.2: System $G$ and observation mapping $\theta$ for Example 7.2.2*

## 7.3 Weak Opacity Control Problem (WOCP)

If $L \subseteq \mathcal{L}(G)$ is not weakly opaque with respect to $K \subseteq \mathcal{L}(G)$, that is, $\theta(L) \cap \theta(K) = \emptyset$, then no control can help to achieve weak opacity. In other words, there is no controller $S$ generating $\mathcal{L}(S/G) = M$ such that the controlled system satisfies weak opacity, namely, $\theta(M \cap L) \cap \theta(M \cap K) \neq \emptyset$. This is obvious: If $\theta(L) \cap \theta(K) = \emptyset$, then for any $M \subseteq \mathcal{L}(G)$, $\theta(M \cap L) \cap \theta(M \cap K) = \emptyset$. Therefore, there is no solution to WOCP.

## 7.4 No Opacity Control Problem (NOCP)

If $L \subseteq \mathcal{L}(G)$ is opaque with respect to $K \subseteq \mathcal{L}(G)$, that is, $\theta(L) \cap \theta(K) \neq \emptyset$, then we want to design a controller $S$ to restrict the language of the system from $\mathcal{L}(G)$ to $M = \mathcal{L}(S/G)$ such that in the controlled system, no opacity holds, that is, $\theta(M \cap L) \cap \theta(M \cap K) = \emptyset$. Therefore, our goal is to find a language $M \subseteq \mathcal{L}(G)$ such that

1. $M \subseteq \mathcal{L}(G)$ is controllable w.r.t. $\Sigma_c$ and $\mathcal{L}(G)$.

2. $M \subseteq \mathcal{L}(G)$ is observable w.r.t. $\Sigma_{in,o}$ and $\mathcal{L}(G)$.

3. $M \cap L$ is not opaque w.r.t. $M \cap K$, that is, $\theta(M \cap L) \cap \theta(M \cap K) = \emptyset$.

Let us again consider no opacity first, that is, finding a language $M \subseteq \mathcal{L}(G)$ satisfying Conditions (3). Intuitively, to ensure no opacity, we need to remove strings violating no opacity, that is, $\theta^{-1}(\theta(L) \cap \theta(K))$ from $\mathcal{L}(G)$. To remove as less strings as possible, we remove only the strings in $(K \cup \theta^{-1}\theta(K - L)) \cap L$ or $(L \cup \theta^{-1}\theta(L - K)) \cap K$ from $\mathcal{L}(G)$. Therefore, there are two possible solutions.

**Theorem 7.4.1.** *Given two languages $L, K \subseteq \mathcal{L}(G)$. Let*

$$M_1 = \mathcal{L}(G) - (K \cup \theta^{-1}\theta(K - L)) \cap L.$$

$$M_2 = \mathcal{L}(G) - (L \cup \theta^{-1}\theta(L - K)) \cap K.$$

*Then $M_1 \cap L$ is not opaque w.r.t $M_1 \cap K$ that is $\theta(M_1 \cap L) \cap \theta(M_1 \cap K) = \emptyset$ and $M_2 \cap L$ is not opaque w.r.t $M_2 \cap K$ that is $\theta(M_2 \cap L) \cap \theta(M_2 \cap K) = \emptyset$ .*

*Proof.* We first show that $M_1 \cap L$ is not opaque w.r.t $M_1 \cap K$ that is $\theta(M_1 \cap L) \cap \theta(M_1 \cap K) = \emptyset$ than $M_2 \cap L$ is not opaque w.r.t $M_2 \cap K$ that is $\theta(M_2 \cap L) \cap \theta(M_2 \cap K) = \emptyset$. This can be done by contradiction. Suppose the above is not true, then there exists $s \in \Sigma^*$ such that

$$
\begin{aligned}
& s \in \theta(M_1 \cap L) \wedge s \in \theta(M_1 \cap K) \\
\Leftrightarrow \quad & (\exists t \in M_1 \cap L)(\exists t' \in M_1 \cap K)\theta(t) = \theta(t') = s \\
\Leftrightarrow \quad & (\exists t, t' \in \Sigma^*)\theta(t) = \theta(t') = s \\
& \wedge t \in L \wedge t \in \mathcal{L}(G) \wedge t \notin (K \cup \theta^{-1}\theta(K - L)) \cap L \\
& \wedge t' \in K \wedge t' \in \mathcal{L}(G) \wedge t' \notin (K \cup \theta^{-1}\theta(K - L)) \cap L \\
\Leftrightarrow \quad & (\exists t, t' \in \mathcal{L}(G))\theta(t) = \theta(t') = s \\
& \wedge t \in L \wedge (t \notin (K \cup \theta^{-1}\theta(K - L)) \vee t \notin L) \\
& \wedge t' \in K \wedge (t' \notin (K \cup \theta^{-1}\theta(K - L)) \vee t' \notin L) \\
\Leftrightarrow \quad & (\exists t, t' \in \mathcal{L}(G))\theta(t) = \theta(t') = s \\
& \wedge t \in L \wedge t \notin (K \cup \theta^{-1}\theta(K - L)) \\
& \wedge ((t' \in K \wedge t' \notin (K \cup \theta^{-1}\theta(K - L))) \vee (t' \in K \wedge t' \notin L)) \\
Because \quad & (t' \in K \wedge t' \notin (K \cup \theta^{-1}\theta(K - L))) = false \\
\Leftrightarrow \quad & (\exists t, t' \in \mathcal{L}(G))\theta(t) = \theta(t') = s \\
& \wedge t \in L \wedge t \notin K \wedge t \notin \theta^{-1}\theta(K - L) \\
& \wedge t' \in K \wedge t' \notin L \\
\Leftrightarrow \quad & (\exists t, t' \in \mathcal{L}(G))\theta(t) = \theta(t') = s \\
& \wedge t \in L - K \wedge \theta(t) \notin \theta(K - L) \wedge t' \in K - L \\
\Rightarrow \quad & (\exists t, t' \in \mathcal{L}(G))\theta(t) = \theta(t') = s \\
& \wedge \theta(t) \notin \theta(K - L) \wedge t' \in K - L \\
\Rightarrow \quad & (\exists t, t' \in \mathcal{L}(G))\theta(t) = \theta(t') = s \\
& \wedge \theta(t) \notin \theta(K - L) \wedge \theta(t') \in \theta(K - L)
\end{aligned}
$$

a contradiction!$(s \notin \theta(K - L) \wedge s \in \theta(K - L))$

Similar to the previous proof $\theta(M_2 \cap L) \cap \theta(M_2 \cap K) = \emptyset$ can be proved as follow.

$$
\begin{aligned}
& s \in \theta(M_2 \cap L) \wedge s \in \theta(M_2 \cap K) \\
\Leftrightarrow \quad & (\exists t \in M_2 \cap L)(\exists t' \in M_2 \cap K)\theta(t) = \theta(t') = s \\
\Leftrightarrow \quad & (\exists t, t' \in \Sigma^*)\theta(t) = \theta(t') = s \\
& \wedge t \in L \wedge t \in \mathcal{L}(G) \wedge t \notin (L \cup \theta^{-1}\theta(L - K)) \cap K \\
& \wedge t' \in K \wedge t' \in \mathcal{L}(G) \wedge t' \notin (L \cup \theta^{-1}\theta(L - K)) \cap K \\
\Leftrightarrow \quad & (\exists t, t' \in \mathcal{L}(G))\theta(t) = \theta(t') = s \\
& \wedge t \in L \wedge (t \notin (L \cup \theta^{-1}\theta(L - K)) \vee t \notin K) \\
& \wedge t' \in K \wedge (t' \notin (L \cup \theta^{-1}\theta(L - K)) \vee t' \notin K) \\
\Leftrightarrow \quad & (\exists t, t' \in \mathcal{L}(G))\theta(t) = \theta(t') = s \\
& \wedge t \in L \wedge t \notin (L \cup \theta^{-1}\theta(L - K)) \vee t \notin K) \\
& \wedge t' \in K \wedge t' \notin ((L \cup \theta^{-1}\theta(L - K)) \vee (t' \in K \wedge t' \notin K)) \\
\textit{Because} \quad & (t \in L \wedge t \notin (L \cup \theta^{-1}\theta(L - K))) = false \\
\textit{and} \quad & (t' \in K \wedge t' \notin K) = false \\
\Leftrightarrow \quad & (\exists t, t' \in \mathcal{L}(G))\theta(t) = \theta(t') = s \\
& \wedge t \in L \wedge t \notin K) \\
& \wedge t' \in K \wedge t' \notin (L \cup \theta^{-1}\theta(L - K)) \\
\Leftrightarrow \quad & (\exists t, t' \in \mathcal{L}(G))\theta(t) = \theta(t') = s \\
& \wedge t \in (L - K) \\
& \wedge t' \in K \wedge t' \notin L \wedge t' \notin \theta^{-1}\theta(L - K)) \\
\Rightarrow \quad & (\exists t, t' \in \mathcal{L}(G))\theta(t) = \theta(t') = s \\
& \wedge \theta(t) \in \theta(L - K) \wedge \theta(t') \notin \theta(L - K)
\end{aligned}
$$

a contradiction!$(s \notin \theta(L - K) \wedge s \in \theta(L - K))$

□

If the languages $M_1$ or $M_2$ obtained in Theorem 7.4.1 are not controllable and observable, then we need to reduce $M_1$ or $M_2$ further to compute $M_1^{\uparrow(CN)}$ or $M_2^{\uparrow(CN)}$. Unlike the case for strong opacity, the reduced language $M_1^{\uparrow(CN)}$ or $M_2^{\uparrow(CN)}$ are not opaque as shown in the following proposition.

**Proposition 7.4.1.** *Given three languages $L, K, M \subseteq \mathcal{L}(G)$. If $\theta(M \cap L) \cap \theta(M \cap K) = \emptyset$, then for any $\acute{M} \subseteq M$,*

$$\theta(\acute{M} \cap L) \cap \theta(\acute{M} \cap K) = \emptyset.$$

*Proof.* The result follows from the fact

$$\theta(\acute{M} \cap L) \cap \theta(\acute{M} \cap K) \subseteq \theta(M \cap L) \cap \theta(M \cap K).$$

□

From Proposition 7.4.1, we know that no iteration is needed to solve the NOCP.

**Theorem 7.4.2.** *Given two languages $L, K \subseteq \mathcal{L}(G)$. Let*

$$M_1 = (\mathcal{L}(G) - (K \cup \theta^{-1}\theta(K - L)) \cap L)^{\uparrow(CN)},$$

$$M_2 = (\mathcal{L}(G) - (L \cup \theta^{-1}\theta(L - K)) \cap K)^{\uparrow(CN)}$$

*Then $M_1$ and $M_2$ are solutions to the NOCP. That is, $M_1$ and $M_2$ satisfy the following.*

1. *$M_1, M_2 \subseteq \mathcal{L}(G)$ are controllable w.r.t. $\Sigma_c$ and $\mathcal{L}(G)$.*
2. *$M_1, M_2 \subseteq \mathcal{L}(G)$ are observable w.r.t. $\Sigma_{in,o}$ and $\mathcal{L}(G)$.*
3. *$M_1 \cap L$ is not opaque w.r.t. $M_1 \cap K$, that is, $\theta(M_1 \cap L) \cap \theta(M_1 \cap K) = \emptyset$. $M_2 \cap L$ is not opaque w.r.t. $M_2 \cap K$, that is, $\theta(M_2 \cap L) \cap \theta(M_2 \cap K) = \emptyset$.*

*Proof.* Hence $M_1$ and $M_2$ are controllable and observable. Therefore, items 1) and 2) hold.

Since $M_1 \subseteq \mathcal{L}(G) - (K \cup \theta^{-1}\theta(K - L)) \cap L$ and $M_2 \subseteq \mathcal{L}(G) - (L \cup \theta^{-1}\theta(L - K)) \cap K$. By Theorem 7.4.1 and Proposition 7.4.1, $M_1 \cap L$ is not opaque w.r.t. $M_1 \cap K$ and $M_2 \cap L$ is not opaque w.r.t. $M_2 \cap K$. □

We illustrate the result in the following example.

**Example 7.4.1.** *We consider the system $G$ in Figure 7.3. Let $\Sigma = \{\alpha, \beta, \gamma, \sigma, \delta, \pi\}$, $\Sigma_{in,o} = \Sigma$, and $\Sigma_{ex,o} = \{\alpha, \beta, \gamma, \delta, \pi\}$. Let $\Sigma_c = \{\alpha, \beta, \sigma\}$. Let the observation mapping $\theta$ be defined as illustrated in the figure, where the unobservable transitions are replaced by $\epsilon$. Let*

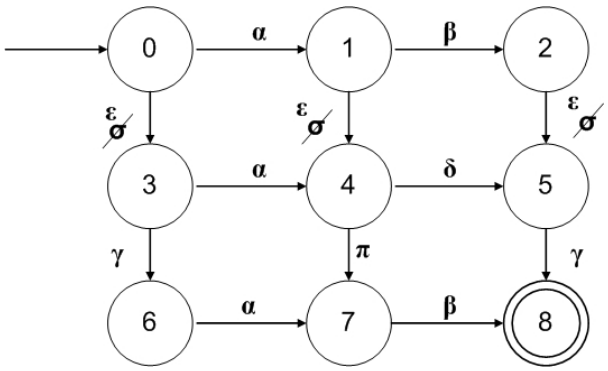



*Figure 7.3: System $G$ and observation mapping $\theta$ used in Example 7.4.1*

$L = \alpha\beta\sigma\gamma + \sigma\alpha\delta\gamma + \alpha\sigma\pi\beta;$

$K = \sigma\gamma\alpha\beta + \alpha\sigma\delta\gamma + \sigma\alpha\pi\beta.$

*Then*

$\theta(L) = \alpha\beta\gamma + \alpha\delta\gamma + \alpha\pi\beta;$

$\theta(K) = \gamma\alpha\beta + \alpha\delta\gamma + \alpha\pi\beta.$

*$L$ is (weakly) opaque with respect to $K$ and $\theta$, because $\theta(L) \cap \theta(K) = \alpha\delta\gamma + \alpha\pi\beta$. By Theorem 7.4.2, we can solve the NOCP as follows.*

$M = (\mathcal{L}(G) - (K \cup \theta^{-1}\theta(K - L)) \cap L)^{\uparrow(CN)}$

$= \sigma\gamma\alpha\beta + \alpha\sigma\delta\gamma + \sigma\alpha\pi\beta + \alpha\beta\sigma\gamma$. *It is not difficult to check that $M$ is controllable, observable, and $\theta(M \cap L) \cap \theta(M \cap K) = \emptyset$.*

# 8 Conclusions

## 8.1 Summary

For discrete event systems under partial observation, opacity and its control is studied in this dissertation. In this dissertation we investigated the properties of the opacity and the behaviour of the opaque languages in the framework of discrete event systems. In the light of these investigations, an intensive study was conducted to the opacity in the framework of discrete event systems proposed in [33]. The goal is to come up with a formal basis and the theoretical foundation to formalize opacity in the modelling and analysis process.

As shown in the dissertation, such strong opacity, weak opacity, and no opacity can be a key for efficient behavioural analysis in complex systems. This analysis improves the understanding of different interests, such as security concerns or providing software engineers with a deeper understanding of any opaque behaviour in order to have a robust system. We presented the properties of opacity that can be used to find a solution to the problems of modification of a language to satisfy strong opacity, weak opacity, and no opacity respectively. We showed that the supremal strongly opaque sublanguage exists and it is unique; the minimal strongly opaque superlanguage exists, but may not be unique. We showed that the minimal weakly opaque superlanguage exists, but may not be unique. Also we showed that the supremal not opaque sublanguage exists and it is unique. Formulas were derived to compute these sublanguages and superlanguages.

We investigated the verification of the dining cryptographers problem in discrete event system framework. We verified the protocol using strong opacity and no opacity properties. We show that we can synthesize the protocol by managing to solve half of the problem using formulas for decentralized strong opacity and decentralized no opacity.

We end the dissertation by investigating the supervisory control problem of opacity of discrete event system. We proposed several approaches for solving these problems. If opacity properties are not satisfied originally, supervisors can be used to restrict the system's

behavior to ensure opacity if possible. Also, conditions have been established for solutions of supervisory control problems of opacity. Algorithms and formulas were derived to compute the largest controllable and observable sublanguage such that the opacity properties hold for an external user of the system.

## 8.2 Future Research

In this thesis, many important aspects on behavioural analysis of opacity have been addressed. However, much research remains to be done to address other important aspects.

### 8.2.1 Stochastic Systems

The presented work is done in the framework of discrete event systems which has been restricted to non-stochastic systems. "theory of stochastic discrete event systems on the other hand has been well developed and understood. Such a stochastic discrete event system can clearly be represented by an automaton with probabilities associated with transitions such that the probabilities of all transitions from any state add up to at most one" [11]. Formally, the probabilistic language is used to describe the behavior of stochastic discrete event systems. Such systems are also represented as nondeterministic automata with probabilities associated with transitions. Intuitively we define probabilistic strong opacity and probabilistic weak opacity as follows. Given a general observation mapping, a language is probabilistic strongly opaque if the probability of all strings in the language are belonging to another language and it is weakly opaque if the probability of some strings in the language are belonging to another language. In the context of opacity we consider a scenario where we are given a stochastic discrete event system that can be modeled as a probabilistic finite automaton with an observation mapping $\theta$, and languages $L, K \subseteq \mathcal{L}(G)$. If $L$ is not probabilistic strongly opaque w.r.t $K$ and $\theta$, then we may enlarge $K$ or shrink $L$. We define the probabilistic strong opacity, where it considers the probability that the enlarged $K$ such that $L$ is probabilistic strongly opaque w.r.t $K$ and $\theta$, if this probability lies below a

predetermined threshold in the system, other wise we consider the probabilistic weak opacity.

### 8.2.2 Decentralized Supervisory Control of Opacity

So far we have restricted our attention to the problem of centralized control of opacity under partial observation. However, in many applications such as manufacturing systems, communication networks, energy management systems, etc., it is desirable to have decentralized controllers, where each controller is able to control a certain set of events and is able to observe a certain set of events in order to achieve opacity properties. The problem of decentralized opacity control may vary on the basis of opacity properties need to be achieved. For instance, strong opacity can be investigated without losing of generality by considering the case of tow controllers. Given a finite deterministic system $G$, two regular languages $L, K \subseteq \mathcal{L}(G)$, and an observation mapping $\theta$ based on the set of events that can be observed by an external observer, $\Sigma_{ex,o} \subseteq \Sigma$. The system $G$ is observed by a set $J = \{1, 2\}$ of agents $A_j$, each one of them observes the system $G$ using its own observation mapping $\theta_j (j \in J)$. If $L$ and $K$ do not satisfy opacity properties, we would like to design a set of controllers or supervisors $S = \{S_1, S_2\}$, if possible, to restrict the behavior of $G$ so that opacity properties are satisfied. Each supervisor can disable a subset of controllable events $\Sigma_{c,j} \subseteq \Sigma_c \subseteq \Sigma$. We assume that the supervisors are internal to the system, so that they together can observe more events than the external observer. The set of events that can be observed by the supervisors are denoted by $\Sigma_{in,o,j} \subseteq \Sigma$ for $(j \in J)$. We assume $\Sigma_{ex,o} \subseteq \bigcup_{j \in J}^{n} \Sigma_{in,o,j}$ and the equality may not hold in general.

### 8.2.3 Software Implementation

A software implementation of opacity algorithms and theories in the framework of discrete event systems would help for verification and validation of models, and designing simulation experiments. Verification and supervisory control of opacity procedures described in this thesis can be directly translated into software functions within widely used tools. Such software tool is essential to bring the result of this dissertation to potential non-specialist

users. Clearly implementation of opacity algorithms and theories require special arrangement in the input interface that might still not be able to use the general observation mapping properties. On the hand interaction of the corresponding operations (composition, decomposition, abstractions, ... etc.) are already known in many discrete event system tools.

# BIBLIOGRAPHY

[1] A.A.Puntambekar. *Formal Languages And Automata Theory.* Technical Publications Pune, 2008.

[2] Romain Beauxis and Catuscia Palamidessi. Probabilistic and nondeterministic aspects of anonymity. *Theoretical Computer Science*, 410(41):4006–4025, 2009.

[3] Majed Ben-Kalefa and Feng Lin. Opaque superlanguages and sublanguages in discrete event systems. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 199–204. IEEE, 2009.

[4] Majed Ben-Kalefa and Feng Lin. Supervisory control for opacity of discrete event systems. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 1113–1119. IEEE, 2011.

[5] Majed Ben-Kalefa and Feng Lin. Opaque superlanguages and sublanguages in discrete event systems. *Submitted to Cybernetics and systems*, 2013.

[6] Lafortune Stephane Cassandras, Christos G. *Introduction to Discrete Event Systems SpringerLink Engineering.* Springer, 2007.

[7] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[8] A. Borzyszkowski B. Caillaud E. Badouel, M. Bednarczyk and P. Darondean. Concurrent secrets. *Discrete Event Dynamic Systems*, 17(4), 2007.

[9] J. Dubreil F. Cassez and H. Marchand. Dynamic observers for the synthesis of opaque systems. Technical Report 1930, IRISA, May 2009. Extended version of the paper of ATVA'09.

[10] M. Fabian and R. Kumar. Mutually non-blocking supervisory control of discrete event systems. In *Decision and Control*, volume 3. IEEE, Proceedings of the 36th IEEE Conference on, 1997.

[11] Kumar R. Garg, V. K. and Marcus S. I. Probabilistic language framework for stochastic discrete event systems. *Technical Report 96-18*, April 1996.

[12] Y. Guan. Implementation of hierarchical observer theory. *M.A.Sc. thesis, Department of electrical and Computer Engineering, University of Toronto*, 1997.

[13] W.M. Wonham H. Zhong. On the consistency of hierarchical supervision in discrete event systems. *IEEE Transaction on Automatic Control*, 35(10):1125–1134, 1990.

[14] Joseph Y. Halpern and Kevin R. OŠNeill. Anonymity and information hiding in multiagent systems. *In Proc. of the 16th IEEE Computer Security Foundations Workshop*, pages 75–88, 2003.

[15] Joseph Y. Halpern and Kevin R. OŠNeill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–512, 2005.

[16] Tomas Harreveld. Dining cryptographer networks.

[17] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[18] D. Hughes and V. Shmatikov. Information hiding, anonymity and privacy: a modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.

[19] P. Darondeau J. Dubreil and H. Marchand. Opacity enforcing control synthesis. *Workshop on Discrete Event Systems, WODES'08*, March 2008.

[20] P. Darondeau J. Dubreil and H. Marchand. Supervisory control for opacity. *IEEE Transactions on Automatic Control*, May 2010.

[21] T. Jéron J. Dubreil and H. Marchand. Automatic testing of access control for security properties. September 2009.

[22] T. Jéron J. Dubreil and H. Marchand. Monitoring confidentiality by diagnosis techniques. *Proceedings of the 10th European Control Conference (ECC'09)*, August 2009.

[23] L. Mazare J. W. Bryans, M. Koutny and P. Y. A Ryan. Opacity generalized to transition systems. *FAST 2005, LNCS 2860*, pages 81–95, 2006.

[24] M. Koutny J. W. Bryans and P. Y. A. Ryan. Modeling opacity using petri nets. *Electronic notes in Theoretical Computer Science*, 121:101–115, February 2005.

[25] M.Koutny J. W. Bryans and P. Y. A. Ryan. Modelling dynamic opacity using petri nets with silent actions. *FAST*, 2004.

[26] W.M. Wonham K.C. Wang. Hierarchical control of discrete event systems. *Theory and Applications*, 6(3):241–273, 1996.

[27] Catuscia Palamidessi Konstantinos Chatzikokolakis and Prakash Panangaden. Anonymity protocols as noisy channels. *Information and Computation*, 2007.

[28] Catuscia Palamidessi Konstantinos Chatzikokolakis and Prakash Panangaden. Probability of error in information-hiding protocols. *In Postproceedings of CSF'07, Lecture Notes in Computer Science*, 2007.

[29] F. Lin. Diagnosability of discrete event systems and its applications. *Discrete Event Dynamic Systems: Theory and Applications*, 4(1):197–212, 1994.

[30] F. Lin and W. M. Wonham. Decentralized supervisory control of discrete event systems. *Information Sciences*, 44:199–224, 1988.

[31] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44:173–198, 1988.

[32] F. Lin and W. M. Wonham. Supervisory control of timed discrete event systems under partial observation. *IEEE Transactions on Automatic Control*, 40(3):558–562, 1995.

[33] Feng Lin. Opacity of discrete event systems and its applications. *Automatica*, 47(3):496–503, 2011.

[34] S. Lafortune M. Sampath and D. Teneketzis. Active diagnosis of discrete event systems. *IEEE Trans. Automatic Control*, 43(7):908–929, 1998.

[35] S. Lafortune K. Sinnamohideen M. Sampath, R. Sengupta and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[36] S. Lafortune K. Sinnamohideen M. Sampath, R. Sengupta and D. Teneketzis. Failure diagnosis using discrete event models. *IEEE Trans. Control Systems Technology*, 4(2):105–124, 1996.

[37] L. Mazare. Decidability of opacity with non-atomic keys. *Verimag Technical Report number*, 23, Nov 2004.

[38] L. Mazare. Using unification for opacity properties. *The Theory of Security*, WITS, 2004.

[39] A. Nerode. Linear automaton transformations. *In In Proc. of the American Mathematical Society*, 9:541Ű544, 1958.

[40] The Encyclopedia of Mathematics wiki. Automata, composition of.

[41] A. D. Fabio P. K. Anumula and J. Zhu. Introduction to theoretical computer science/theory of computation @OTHER, 2011.

[42] A. Paoli and F. Lin. Decentralized opacity of discrete event systems. *Proceedings of 2012 American Control Conference*, 2011.

[43] R. Kumar F. Lin S. I. Marcus R. D. Brandt, V. Garg and W. M. Wonham. Formulas for calculating supremal controllable and normal sublanguages. *Systems and Control Letters*, 15:111–117, 1991.

[44] N Mathewson R Dingledine and P Syverson. Reputation in p2p anonymity systems. *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[45] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete-event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987. Seminare Maurey-Schwartz (1975-1976).

[46] M.K. Reiter and A.D. Rubin. Crowds: anonymity for web transactions. *ACM Transactions on Information and System Security*, 1998.

[47] Peter Y. Ryan and Steve Schneider. *Modelling and Analysis of Security Protocols*. 2001.

[48] F. Lin S. Shu and H. Ying. Detectability of discrete event systems. *IEEE Transactions on Automatic Control*, 52(12):2356–2359, 2007.

[49] H. Ying S. Shu, F. Lin and X. Chen. State estimation and detectability of probabilistic discrete event systems. *Automatica*, 44(12):3054–3060.

[50] A. Saboori and C. N. Hadjicostis. Notion of security and opacity in discrete event systems. *46th IEEE Conference on Desision and Control*, pages 5056–5061, December 2007.

[51] A. Saboori and C. N. Hadjicostis. Verfication of initial-state opacity in security applications of discrete event systems. *9th Workshop on Discrete Event Systems*, May 2008.

[52] Klaus Schmidt. *Hierarchical Control of Decentralized Discrete Event Systems Theory and Application*. Doktor-ingenieur, Der Technischen Fakultät der Universität Erlangen-Nürnberg, 2005.

[53] Steve Schneider and Abraham Sidiropoulos. Csp and anonymity. *In Proc. of the European Symposium on Research in Computer Security (ESORICS)*, 1146 of Lecture Notes in Computer Science:198–218, 1996.

[54] Herve Marchand et Sophie Pinchinat Sebastien Chedor encadre par Christophe Morvan. Infinite discrete event systems and partial information. *ENS, IRISA*, 2009.

[55] V Shmatikov. Probabilistic model checking of an anonymity system. Technical report, 2004.

[56] T. A. Sudkamp. *Languages and machines: an introduction to the theory of computer science*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA ĺ1988 ISBN:0-201-15768-3, 1988.

[57] Paul F. Syverson and Stuart G. Stubblebine. Group principals and the formalization of anonymity. *In World Congress on Formal Methods*, 1:814–833, 1999.

[58] S. Genc T. Jeron, H. Marchand and S. Lafortune. Predictive diagnosis for discrete event systems diagnosis. *Technical Report, IRISA*, 1834, 2007.

[59] S. Pinchinat T. Jeron, H. Marchand and M. O. Cordier. Supervision patterns in discrete event systems diagnosis. *In Workshop on Discrete Event Systems, WODES"06, Ann-Arbor (MI, USA)*, 111, july 2006.

[60] Xingyuan Zhang Wu, Chunhan and Christian Urban. A formalisation of the myhill-nerode theorem based on regular expressions (proof pearl).interactive theorem proving. *Springer Berlin Heidelberg*, pages 341–356, 2011.

# ABSTRACT

## OPACITY OF DISCRETE EVENT SYSTEMS: ANALYSIS AND CONTROL

by

**MAJED BEN KALEFA**

**December 2013**

**Advisor:**Prof. Feng Lin

**Major:** Electrical Engineering

**Degree:** Doctor of Philosophy

The exchange of sensitive information in many systems over a network can be manipulated by unauthorized access. Opacity is a property to investigate security and privacy problems in such systems. Opacity characterizes whether a secret information of a system can be inferred by an unauthorized user. One approach to verify security and privacy properties using opacity problem is to model the system that may leak confidential information as a discrete event system. The problem that has not investigated intensively is the enforcement of opacity properties by supervisory control. In other words, constructing a minimally restrictive supervisor to limit the system's behavior so an unauthorized user cannot discover or infer the secret information.

We describe and analyze the complexity of opacity in systems that are modeled as a discrete event system with partial observation mapping. We define three types of opacity: strong opacity, weak opacity, and no opacity. Strong Opacity describes the inability for the system's observer to know what happened in a system. On the other hand, No-opacity refers to the condition where there is no ambiguity in the system behavior. The definitions introduce properties of opacity and its effects on the system behavior. Strong opacity can be used to study security related problems while no opacity can be used to study fault, detection and diagnosis, among many other applications. In this dissertation, we investigate the largest opaque sublanguages and smallest opaque superlanguages of a language if the language is not opaque. We present formulas for calculating the sublanguages and superlanguages in centralized framework as well as in a decentralized framework. We illustrate the concept of

opacity in the framework of discrete event system to the property of anonymity by applying the proposed opacity to verify and to synthesize the dining cryptographers protocol. We studied how to ensure strong opacity, weak opacity and no opacity by supervisory control. If strong opacity, weak opacity or no opacity is not satisfied, then we can restrict the system's behavior by a supervisor so that strong opacity, weak opacity or no opacity is satisfied. We investigate the strong opacity control problem (SOCP), the weak opacity control problem (WOCP), and no opacity control problem (NOCP).

As illustrated by examples in the dissertation, the above properties of opacity can be used to characterize the security requirements in many applications, as anonymity requirements in protocols for web browsing. Solutions to SOCP in terms of the largest sublanguage that is controllable, observable (or normal), and strongly opaque were characterized. Similar characterization is available for solutions to NOCP.

# AUTOBIOGRAPHICAL STATEMENT

Born in small city Zwara located on the west coast of Libya. I have been raised in a good family environment with well-educated brothers since. This environment was the most effective factor that helped me to develop strengths in myself such as being a good listener and quick learner. Afterwards, it was to me very clear that my family is the starting point of my success in my past educations and my current working life.

For my academic life, I have had good starting school with good teachers who definitely were one the reasons for enhancing my creativity, initiative, and leadership. At the undergraduate level, I had also a good chance to study in a good university which awarded for my graduation with outstanding award. I have received a scholarship for my graduate study in Germany.

Graduate study in Germany was the point of the change in my academic life. I have been accepted at Karlsruhe University that offers excellent quality graduate programs in electrical and information technology. Having success with this opportunity was my goal. I have worked hard toward my Master degree and was successfully graduated in 2003. I have substantially attained my professional and academic goals.

After the graduation went back to Libya and started my work experience in teaching at the university. I obtained a position as a teacher assistant. I was very grateful to had this job for three years. While working for the university I have had the opportunity to work with well known professors. Being able to interact with many different students and classes has helped me decide which area to continuo my research and PhD.

Finally, by looking ahead, I have intended to get myself heavily trained in Wayne state university doctoral programme to acquire excellent research and teaching skills to ensure the first-class contribution in my future academic profession. I am confident that this training, together with my consistently-cultivated strengths, are considered beneficial not only to myself or Zawia University as my prospective sponsor but also to Libyan students and Libyan community as a whole.