Journal of Theoretical and Applied Information Technology
31st May 2024. Vol.102. No. 10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TOWARDS A UNIFIED FORMALIZATION OF OPACITY PROPERTIES IN DISCRETE AND REAL-TIME SYSTEMS

HAFEDH MAHMOUD ZAYANI[1] , IKHLASS AMMAR[2], MOHAMMAD H. ALGARNI[3], AHMAD ALSHAMMARI[4], AMJAD A. ALSUWAYLIMI[5], JIHANE BEN SLIMANE[6], MAROUAN KOUKI[7], AMANI KACHOUKH[8], REFKA GHODHBANI[9], TAOUFIK SAIDANI[10]

[1]Department of Electrical Engineering, College of Engineering, Northern Border University, Arar, Saudi Arabia, (corresponding author)

[2]OASIS Laboratory, National Engineering School of Tunis, University of Tunis El Manar, Faculty of Sciences of Tunis (FST), Tunisia

[3]Department of Computer Science Al-Baha University, Saudi Arabia

[4,6,9,10]Department of Computer Sciences Faculty of Computing and Information Technology, Northern Border University, Rafha 91911, Saudi Arabia

[5]Department of Information Technology, College of Computing and Information Technology, Northern Border University, Saudi Arabia

[7,8]Department of Information Systems, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia

E-mail: [1]hafedh.zayani@nbu.edu.sa, [2]Ikhlass_ammar@yahoo.fr, [3]malgarni@bu.edu.sa, [4]Ahmad.Almkhaidsh@nbu.edu.sa, [5]amjad.alsuwaylimi@nbu.edu.sa, [6]jehan.saleh@nbu.edu.sa, [7]marouan.kouki@nbu.edu.sa, [8]amani.khasookh@nbu.edu.sa, [9]Refka.Ghodhbani@nbu.edu.sa, [10]Taoufik.Saidan@nbu.edu.sa

ABSTRACT

This paper proposes a unified framework for defining opacity properties in both discrete and real-time systems. The framework leverages language inclusion problems to establish a common ground for expressing and comparing various opacity concepts under different observation categories. We build upon existing formalisms for opacity in Labeled Transition Systems (LTS) and Timed Transition Systems (TTS). We explain the connection between these automata models and how they are used to represent system behavior. Our framework allows for the unification of opacity definitions across these models, enabling easier comparison and analysis. Additionally, we present transformations between different opacity concepts and compile decidability results for the unified framework. Finally, we illustrate the relationships between key opacity studies through a dependency diagram.

Keywords: *Discrete Event System, Real Time System, Opacity, Verification, Decidability.*

1. INTRODUCTION

Ensuring confidentiality in complex systems is crucial, especially when dealing with sensitive information. Traditional security models like non-interference might not always suffice. This paper delves into a powerful security property called opacity. This property guarantees a system's ability to hide a specific subset of its behavior, even if the general operation is visible to an external observer (often referred to as an attacker). This means the attacker cannot definitively determine if the system is in a secret state or performing a secret action, even by observing its public behavior.

Research on opacity has been steadily growing, with applications in diverse areas like cryptography and Discrete Event Systems (DES). Different studies utilize various system models (e.g., Petri nets [4], Labeled Transition Systems [6, 35], Automata [7, 22, 23], recursive tile [23] and pushdown systems [8]) and observation scenarios. This can make it challenging to compare and analyze opacity properties across these diverse contexts. In a system's LTS model, predicates act as spotlights, highlighting specific subsets of states or events. LTS, unlike Finite State Automata (FSA), aren't limited to a finite number of states or transitions in [27, 30, 34]. Then, the property of opacity is introduced in a real-time system modeled by Timed Transition System (TTS).

The author in [21], proposes the timed opacity for real-time system modeled by timed automata (TA).

This paper proposes a unified framework for formalizing opacity properties. This framework allows us to analyze and compare opacity across different system models and observation settings. Here's what you can expect:

- We explore various observation categories through clear examples.
- We unify the definitions of opacity properties within our framework.
- We establish mathematical connections between existing opacity formalisms.
- We compile existing results on the decidability of these unified opacity concepts.
- We present a dependency diagram visualizing the relationships among key studies on opacity.

This unified framework paves the way for a more comprehensive understanding of opacity. It allows researchers to compare different opacity properties, fostering advancements in the field. The framework also lays the groundwork for potential future research on verification methods and decidability of opacity properties under various conditions.

By the end of this paper, you will have a deeper understanding of:

- The concept of opacity and its importance in system security.
- How a unified framework simplifies the analysis of opacity across diverse systems.
- The existing body of research on opacity and its connection to our proposed framework.

The paper is structured as follows. Section 2 introduces the background concepts of transition systems and languages, including Labeled Transition Systems (LTS) and Timed Transition Systems (TTS). Section 3 delves into Timed Automata, the standard modeling formalism for real-time systems. Section 4 explores the concept of observation functions, covering static, dynamic, and Orwellian projections. Section 5 presents established opacity properties for discrete systems with static projections. In this section, we propose a unified framework for formalizing opacity properties. Section 6 explores the transformation between different opacity notions. Sections 7 and 8 examine opacity with dynamic and Orwellian projections, respectively. Section 9 extends opacity to timed systems with static projections using the proposed framework. Section 10 discusses verification and decidability of opacity properties. Section 11 provides a comparative overview of existing opacity definitions and our proposed framework. Finally, Section 12 concludes the paper by summarizing the contributions of the unified framework and outlining potential avenues for future research.

2. TRANSITION SYSTEMS AND LANGUAGES

Transition systems can be used to simulate software and hardware systems, with states representing various configurations and actions causing transitions between them, in [35]. One way to represent this is by using a graph, where the states are represented as vertices and the actions are represented as labeled edges. State labeling enhances the available information regarding the values of variables. The paradigm used for discrete systems is referred to as a Labeled Transition System, whereas for real-time systems it is called a Timed Transition System.

2.1. Labeled Transition Systems (LTS) and Discrete Languages

LTSs are essentially infinite, directed graphs with labeled edges, in [12]. Nodes represent the system's states, and edges depict transitions between them triggered by specific actions.

Definition 1: The **Labeled Transition System** is a quadruple $LG = (Q, \mathrm{Q}_0, \Sigma, \rightarrow)$ where: Q is a finite set of states, Σ is a finite set of actions, $Q_0 \subset Q$ is the set of initial state, $\rightarrow \subseteq (Q \times \Sigma^* \times Q)$ is the transition relation.

Note: $\mathbb{N}$ is the set of natural numbers. $\mathbb{Q}, \mathbb{Q}_+, \mathbb{Q}^*$ is respectively the set of rational, nonnegative rational and positive rational. X is the set of clocks i.e., the set of conjunctions of constraints of the form $x \sim c$ and C(X) be the set of convex constraints on X, in the form $\phi ::= x - y \sim c \mid x \sim c \mid \phi \wedge \phi$ with $\sim \in \{<, \leq, =, \geq, >\}$ and x,y $\in \mathbb{Q}_+$. A clock valuation is a mapping $\upsilon: X \longrightarrow \mathbb{Q}_+$. $(\upsilon + d)(x) = \upsilon(x) + d$ where $d \in \mathbb{Q}_+$. $\upsilon[X' \longrightarrow 0] = 0$ if x$\in$ X', otherwise $\upsilon[X' \longrightarrow 0] = \upsilon(x)$, $\forall$ X'$\subset$ X.

A **path** $\Phi = \mathrm{q}_0, \mathrm{q}_1, .. \mathrm{q}_i ..$ is an infinite sequence of states. Φ [i] is the ith element of $\Phi$and $\Phi[..\mathrm{i}] = \mathrm{q}_0, \mathrm{q}_1, \cdots \mathrm{q}_i$, $\Phi[\mathrm{i}..\mathrm{j}] = \mathrm{q}_i, \mathrm{q}_{i+1}, .\mathrm{q}_j$ where $q_0 \in Q_0$ and $\forall \mathrm{i} \geq 0, q_i \in Q$. $Path(LG, \mathrm{q_i})$ is the set of all paths executed by a LTS started by the set of state $\mathrm{q_i}$ and $Path(LG) = Path(LG, \mathrm{q_0})$ when $\mathrm{q_0}$ is the initial states of LG. We note that the set of path is infinite and uncountable set.

An **execution** $w = exec(\Phi) = \mathrm{a}_0, \mathrm{a}_1, .. \mathrm{a}_i ..$ is an infinite sequence of actions. The LTS can accept the empty string, denoted by ε. w_1 is a prefix of w_2

denoted by $w_1 \preccurlyeq w_2$, if $\exists\, w_3$ such that $w_1.w_3 = w_2$ and $w_3 = w_2 - w_1$. $|w|$ returns the length of the discrete word w where $w, w_1, w_2, w_3 \in \Sigma^*$.

A **discrete language** $Lang$ is an infinite set of executions. $Lang(LG, \mathrm{q}_i, \mathrm{q}_j) = \{exec(\Phi[i,j]), \Phi[i,j] \in Path(LG)\}$ is the set of executions stared by q_i and ended by q_j. Extended: $Lang(LG, \mathrm{Q}_i, \mathrm{Q}_j) = \cup_{\mathrm{q}_i \in \mathrm{Q}_i \wedge\, \mathrm{q}_j \in \mathrm{Q}_j} Lang(LG, \mathrm{q}_i, \mathrm{q}_j)$.
$Lang(LG, \mathrm{q}_i) = \{exec(\Phi[i,..]),\ \Phi[i..] \in Pat\ (LG)\}$ is the set of executions started by q_i. Extended: $Lang(LG, \mathrm{Q}_i) = \cup_{\mathrm{q}_i \in \mathrm{Q}_i} Lang(LG, \mathrm{q}_i)$.
$Lang(LG) = Lang(LG, \mathrm{Q}_0)$ is the set of executions started by initial states.

A **bounded discrete language** $Lang_K$, is a finite set of executions, where $K \in \mathbb{N}$ is a constant value. $Lang_K(LG, \mathrm{q}_i) = \{w, \exists\, w' \in Lang\ (LG, q_i)\ such\ that\ w' \preccurlyeq w, |w' - w| \leq K\ \}$ is the set of executions stated by q_i and the length of each execution is less than or equal to K. $Lang_K(LG, \mathrm{Q}_i) = \cup_{\mathrm{q}_i \in \mathrm{Q}_i} Lang_K(LG, \mathrm{q}_i)$.
$Lang_K(LG, \mathrm{q}_i, \mathrm{q}_j) = \{w, \exists\, w' \in Lang\ (LG, q_i, \mathrm{q}_j)\ such\ that\ w_1 \preccurlyeq w, |w_1 - w| \leq K\}$.

Extended:
$Lang_K(LG, \mathrm{Q}_i, \mathrm{Q}_j) = \cup_{\mathrm{q}_i \in \mathrm{Q}_i \wedge\, \mathrm{q}_j \in \mathrm{Q}_j} Lang_K(LG, \mathrm{q}_i, \mathrm{q}_j)$.
$Lang_\infty(LG, \mathrm{q}_i, \mathrm{q}_j) = \{w, \exists w' \in Lang\ (LG, q_i, \mathrm{q}_j)\ such\ that\ w' \preccurlyeq w\}$
Extended:
$Lang_\infty(LG, \mathrm{Q}_i, \mathrm{Q}_j) = \cup_{\mathrm{q}_i \in \mathrm{Q}_i \wedge\, \mathrm{q}_j \in \mathrm{Q}_j} Lang_\infty(LG, \mathrm{q}_i, \mathrm{q}_j)$.

The language can be described by a regular expression. The regular expressions are all strings over the alphabet $\Sigma \cup \{(,), \emptyset, \cup, *, \varepsilon\}$. Formally, $Lang(\emptyset) = \emptyset$;

$Lang(a) = \{a\}$;

$Lang((\mathrm{w}_1, \mathrm{w}_2)) = Lang(\mathrm{w}_1) Lang(\mathrm{w}_2)$;

$\mathrm{Lang}((\mathrm{w}_1 \cup \mathrm{w}_2)) = Lang(\mathrm{w}_1) \cup Lang(\mathrm{w}_2)$;

$Lang(w^*) = Lang(w)^*$

Where a, w_1, w_2 and $w^*$are regular expression.

Time is a critical component in a system. The researchers introduce the concept of time into classical transition systems by assuming that all discrete transitions occur instantly, whereas real-time restrictions limit the possible times at which these transitions might take place. In their work, the authors in [37] present the concept of TTSs and provide the precise definition of a real-time system as a collection of timed execution sequences. The TTS is a Long-Term Support (LTS) system that encompasses two types of labels: discrete and continuous activities of real-time systems.

2.2. Timed Transition Systems (TTS) and timed languages

Timed Transition Systems (TTS) are characterized by a framework that allows for the association of time with a transition relationship [37]. In a TTS, there are two types of transitions: continuous transitions, which depict the passage of time or a gradual change, and discrete transitions, which represent the progression after a specific action or event.

Definition 2: The Timed Transition System is a quadruple $G = (Q, \mathrm{Q}_0, \Sigma, \rightarrow)$ where: Q is a finite set of states, $Q_0 \subset Q$ is the set of initial state, Σ is a finite set of actions, $\rightarrow \subseteq (Q \times (\Sigma \cup \mathbb{Q}_+) \times Q)$ is the transition relation.

The relation $\rightarrow$ is defined by $q \xrightarrow{e} q'$, where $q, q' \in Q$ and e is a transition between them, $(q, e, q') \in \rightarrow$. There are two kinds of transition relation $\rightarrow$: continuous transition relation (or delay transition relation) $\xrightarrow{d \in \mathbb{Q}_+}$ and discrete transition relation $\xrightarrow{a \in \Sigma}$. The properties of TTS are Null delay property or 0-delay if $q \xrightarrow{0} q'$ then $q = q'$; Time additivity property if $q \xrightarrow{\gamma} q'$ and $q' \xrightarrow{\gamma'} q''$ then $q \xrightarrow{\gamma + \gamma'} q''$ with $\gamma, \gamma' \in \mathbb{Q}_+$; Time continuity property if $q \xrightarrow{\gamma} q'$ then $\forall \gamma, '\, \gamma'' \in \mathbb{Q}_+$ such that $\gamma = \gamma' + \gamma''$ and $\exists q'' \in Q$ such that $q' \xrightarrow{\gamma'} q''$ and $q'' \xrightarrow{\gamma''} q'$; Time determinism property if $q \xrightarrow{\gamma} q'$ and $q \xrightarrow{\gamma} q''$ then $q' = q''$.

We extend by $u \preccurlyeq v$ and $|u|$ where $w, v \in (\Sigma \times \mathbb{Q}_+)^*$. $Path$ is the set of transition-run ψ that is a sequence of states.

A **transition-run** $\psi = \mathrm{e}_0, e_1, .. e_i ..$ is an infinite sequence of transitions. For simplicity reason, it is denoted by $\psi = q_0 \xrightarrow{e_0} q_1 \ldots q_i \xrightarrow{e_i} \ldots$ ψ_i is a prefix of ψ, $\psi_i \preccurlyeq \psi$ if $\psi = \psi_i \xrightarrow{e_{i+1}} q_{i+2} \ldots$ and $\psi_i = q_0 \xrightarrow{e_0} q_1 \ldots q_i \xrightarrow{e_i} q_{i+1}$, $\forall i \geq 0$.

$TPath(G)$ is the set of all **path** executed by G. A run ρ_G of a transition-run ψ is a possibly infinite sequence of alternating delay and discrete transition relations $\rho_G(\psi) = q_0 \xrightarrow{d_0} q'_0 \xrightarrow{a_0} q_1 \xrightarrow{d_1} q'_1 \xrightarrow{a_1} q_1 \ldots$ where d_i corresponds to the duration between q_i and q_{i+1}.

An **execution** ρ_G is a possibly infinite execution $\rho_G(\psi) = q_0 \xrightarrow{(a_0, d_0)} q_1 \xrightarrow{(a_1, d_1)} q_2 \ldots \xrightarrow{(a_i, d_i)} q_{i+1} \ldots$

A **trace** $tr(\rho_G(\psi))$ of an execution $\rho_G(\psi)$ is a possibly infinite sequence of alternating time and

discrete transition $tr(\rho_G(\psi)) = q_0 \xrightarrow{(a_0,\gamma_0)} q_1 \xrightarrow{(a_1,\gamma_1)} q_2 \dots \xrightarrow{(a_i,\gamma_i)} q_{i+1} \dots$ where $\gamma_0 = d_0$ and $\gamma_i = \sum_{j=0}^{i} d_j$ that is the executing time at the state q_i that is the sum of all the previous durations in the path.

A **timed word** u of a given trace $tr(\rho_G(\psi))$ is $u = tw(tr(\rho_G(\psi))) = (a_0, \gamma_0)(a_1, \gamma_1) \dots (a_p, \gamma_p) \dots$ The set of generated timed words is represented by the **timed language** denoted by $(G) = \{u = (a_0, \gamma_0) (a_1, \gamma_1) \dots (a_p, \gamma_p) \dots = u = tw(tr(\rho_G(\psi))), \psi \in TPath(G)\}$.

The **finite timed language** is the set of the finite timed words $TL^* = \{u, u = (a_0, \gamma_0)(a_1, \gamma_1) \dots (a_n, \gamma_n)\}$. TL^∞ contains the infinite and finite timed words where $TL^\infty = TL^w \cup TL^*$.

Typically, TTS systems are employed to provide the meaning and description of a model. Timed Automata (TA) are a type of models that are more appropriate for the purposes of modeling, verification, and control.

3. TIMED AUTOMATA AND TIMED LANGUAGE

This section presents the standard modeling formalism for real-time systems, known as Timed Automata, along with its several subclasses. Timed automata, as described in references [5], are automata that have a finite control and a finite set of clocks. They are used to represent real-time systems that operate in continuous time.

Definition 3: [5] A **Timed Automaton** (TA) A is a tuple $A = (L, l_0, X, \Sigma, I, T)$ where L is a finite set of locations; $l_0 \in L$ is the initial location; X is a finite set of clocks such that $n = |X|$; Σ is a finite set of actions; $I \in C(X)^L$ is an application that associates an invariant to each location; T is a finite set of transitions $T \subseteq L \times C(X)^L \times \Sigma \times 2^X \times L$. In an edge $e = (l, g, a, r, l_0) \in T$, g is the guard, a is the action and r is the reset set.

Definition 4: A **Timed Automaton with final states** A_f is a tuple $A_f = (A, F)$ where A is the TA as defined in Definition 3 and F⊆ L is a finite set of final locations.

The semantics of a TA A is determined by a timed transition system that is labeled with transitions. The delay transition signifies the passage of time, while discrete transitions indicate the changeover to the next attainable state in A.

Definition 5: The TTS is a tuple $TG = (Q, \{q_0\}, \Sigma, \rightarrow)$ where Q is the set of states $Q \in (L \times \mathbb{Q}_+^{|X|})$; $\{q_0\}$ is the initial state; $\rightarrow \subseteq (Q \times (\Sigma \cup \mathbb{Q}_+) \times Q)$ is the transition relation.

There are two kinds of transition relation →in TA: Delay transition relation if $(l, v) \xrightarrow{\varepsilon(t)} (l', v')$ then $l = l', v' = v + t$ and $I(l')(v') = True$; Discrete transition relation if $(l, v) \xrightarrow{a} (l', v')$ then $g(v) = True$; $v' = v[r \rightarrow 0]$ and $I(l')(v') = True$ where $e = (l, g, a, r, l_0) \in T$ satisfying the guard g by the clock valuation obtained from adding the delay to the current valuation.

Let A_f be a timed automaton with final locations, a path in A_f is started by the initial location l_0 and ended by a final location $l_p \in F$. This path contains a sequence of transition that is called transition-run $\psi_p = l_0 \xrightarrow{e_0} l_1 \dots \xrightarrow{e_{p-1}} l_p$. $last(\psi_p) = l_p$ returns the last location of ψ_p.

For a finite transition-run ψ_p , an execution automaton $\rho_G(\psi_p) = l_0 \xrightarrow{(a_0,d_0)} l_1 \dots \xrightarrow{(a_{p-1},d_{p-1})} l_p$, a trace $tr(\rho_G(\psi_p)) = l_0 \xrightarrow{(a_0,\gamma_0)} l_1 \dots \xrightarrow{(a_{p-1},\gamma_{p-1})} l_p$, a timed word $u = tw(tr(\rho_G(\psi_p))) = (a_0, \gamma_0)(a_1, \gamma_1) \dots (a_{p-1}, \gamma_{p-1})$, an accepted timed language of A_f is $TL(A_f) = \{u, u = tw(tr(\rho_G(\psi_p))), \psi_p \in TPath(A_f, F)\}$.

$TL(A_f, l) = \{u, u = tw(tr(\rho_G(\psi_p))), \psi_p \in TPath(A_f, F)\ and\ last(\psi_p) = l\}$ is an accepted timed language of A_f where the final location is l. By extension, $TL(A_f, SL) = \cup_{l \in SL} TL(A_f, l)$ is the timed language ended by a subset of locations $SL \subseteq L$.

4. THE OBSERVATION FUNCTIONS

The purpose of opacity is to ascertain whether the concealed actions of a particular system are effectively hidden from external observers. A predicate represents the subset of the system behavior. The outsiders are shown as passive observers of the system's actions and are referred to as intruders. More specifically, the outsider is presumed to possess a comprehensive understanding of the system's architecture and limited observations of the system's functioning. Partial observation typically involves the observation of an execution when an external observer is unable to perceive a subset of events. Hence, the set of events Σ is partitioned into an observable set Σ_o and an unobservable set Σ_u. The visible behavior by an observer is defined by its projection that removes from a sequences w all events that are not in Σ_o.

Opacity qualifies a given predicate $\varphi \in Lang(LG)$ with respect to an observation function Obs modeling user capabilities for observing the system. Formally, $Obs: \Sigma^* \to {\Sigma_o}^*, \forall w \in \Sigma^*, Obs(w) \in {\Sigma_o}^*$ is an observation function. For two executions $w, w' \in \text{Lang(LG)} \subseteq \Sigma^*$, w and w' are observationally equivalent w.r.t. Obs if $Obs(w) = Obs(w')$. Thus, we define some categories of projection in literature in the next section.

4.1. Static projection

Static observation (projection) is the most used observation in the model system, also called simple projection. Static observation is defined when the same occurrence is always interpreted in the same way by an observer. The interface between an observer and a system is identified by a set of events $\Sigma_o \subseteq \Sigma$, with $\Sigma - \Sigma_o = \Sigma_u$ where Σ_u is the set of unobservable events and Σ_o is the set of observable events. Thus, the static projection is defined for the discrete sequence $w = a_1 a_2 \ldots\ldots a_n$, denoted *by* P_{Σ_o}. Formally, $P_{\Sigma_o}: \Sigma^* \to {\Sigma_o}^*$ is defined as follows:

$$\begin{cases} P_{\Sigma_o}(\varepsilon) = \varepsilon; \\ P_{\Sigma_o}(w.a) = P_{\Sigma_o}(w) & if\ a \in \Sigma_u \\ P_{\Sigma_o}(w.a) = P_{\Sigma_o}(w).a & otherwise \end{cases} \quad (1)$$

where $w \in \Sigma^*, a \in \Sigma$ and ε is the empty string.

$[w]_{\Sigma_o}$ represents the set of all execution having the same projection as w. [32] expressed this projection in another form.

Definition 6: [32] According to observation function $Obs: \Sigma^* \to {\Sigma_o}^*$ the static projection is a mapping $Obs': \Sigma \to \Sigma \cup \{\varepsilon\}$ such that $\forall\, w = a_1 a_2 \ldots\ldots a_n \in \Sigma^*, Obs(w) = Obs'(a_1)Obs'(a_2) \ldots Obs'(a_n)$.

Example 1: Let LG be a labeled transition system as shown in Figure 1 with $Q = q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$ set of states, $\Sigma_o = \{a, b\}$ is observable events and $\Sigma_u = \{c\}$ is unobservable events.

The static projection of the word $u = ccabbc$ is defined by $P_{\Sigma_o}(u) = abb$. Using the definition by [32], the static projection of$u = ccabbc$ is $Obs(u) = Obs'(c)Obs'(c)Obs'(a)Obs'(b)Obs'(b)Obs'(c) = \varepsilon\varepsilon abb\varepsilon$ (where $Obs'(c) = \varepsilon$, $Obs'(a) = a$ and $Obs'(b) = b$).

In the same way $P_{\Sigma_o}(v) = abb$ where $v = abcb$. The static projection of the word $v = abcb$ is $Obs(v) = Obs'(a)Obs'(b)Obs'(c)Obs'(b) = ab\varepsilon b$ (where $Obs'(c) = \varepsilon$, $Obs'(a) = a$ and $Obs'(b) = b$).

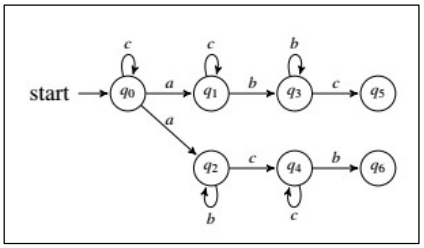

Figure 1: Example of automaton

4.2. Dynamic projection

Dynamic observation, in contrast, involves the study of how a system evolves and changes over time. It considers the interactions, processes, and behaviors that unfold within the system. This approach provides a more holistic understanding by capturing the system's temporal aspects, making it particularly valuable for analyzing systems with fluid and evolving characteristics.

A filter is employed in dynamic projection to impede the transmission of information between the system and the attacker. This approach is introduced in [10, 25] when the projection dynamically modifies the observability of events, and the attackers cannot infer secret information from observations. Dynamic observation is based on the prefix and an observer that can deduce the knowledgement using the previous events to interpret the current i.e. the set of observable events change over time conforming to a dynamic mask. The observer can update after each observation the set of events that he can observe. The interface between an observer and a system is identified by a dynamic observability that is a mapping $T_D: \Sigma^* \to 2^\Sigma$. Formally, the dynamic projection denoted by T_D is the mapping $DP_{T_D}: \Sigma^* \to \Sigma^*$is defined as follows:

$$\begin{cases} DP_{T_D}(\varepsilon) = \varepsilon; \\ DP_{T_D}(w.a) = DP_{T_D}(w).a & if\ a \in T_D(DP_{T_D}(w)) \\ DP_{T_D}(w.a) = DP_{T_D}(w) & otherwise \end{cases} \quad (2)$$

Dynamic functions are akin to an observer with unlimited memory capacity to retain labels. However, they can only rely on knowledge of past labels to understand the present label and cannot subsequently reinterpret it. The dynamic projection can be expressed in another form by [32].

Definition 7: [32] According to observation function : $\Sigma^* \to {\Sigma_o}^*, \Sigma_o \subseteq \Sigma$. The dynamic projection is a mapping $Obs': \Sigma \times \Sigma^* \to \Sigma_o \cup \{\varepsilon\}$ such that $\forall\, w = a_1 a_2 \ldots\ldots a_n \in \Sigma^*$, $Obs(w) = Obs'(a_1, \varepsilon)\, Obs'(a_{2,}, a_1) \ldots Obs'(a_n, a_1 .. a_{n-1})$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Example 2: We recall the LTS system LG of Example 1 in Figure 1. The dynamic observability is defined as follows:

$$\begin{cases} T_D(w) = \{a\} & if\ T_D(w) \in c^*a \\ T_D(w) = \{a,b,c\} & otherwise\ (w \in \Sigma^*) \end{cases} \quad (3)$$

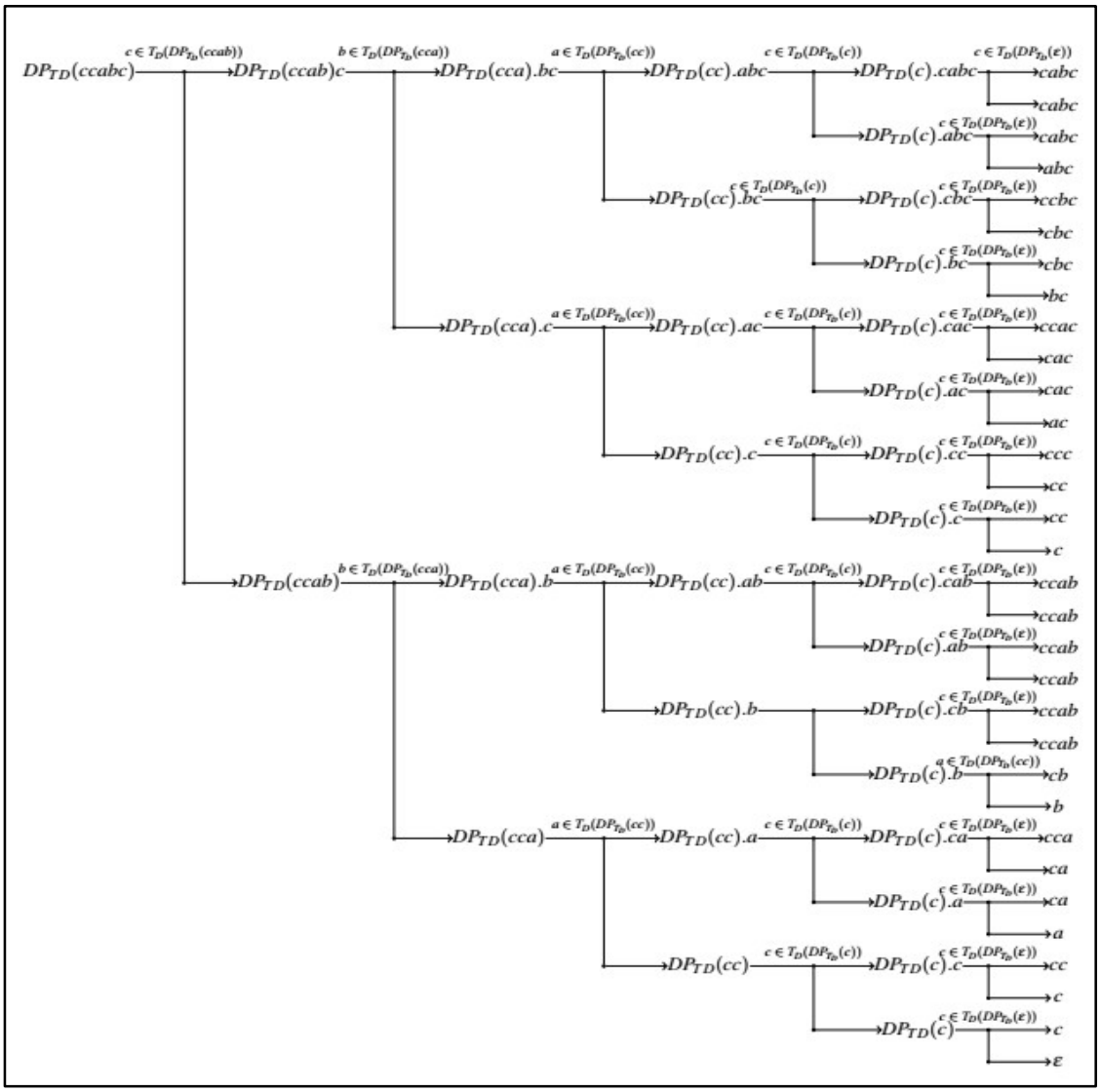

Figure 2: Details Of Dynamic Projection

According to the Figure 2, the dynamic projection of $u = ccabc$ is $DP_{T_D}(u) = DP_{T_D}(ccabc) = cca$. In the same way, $(ccabc) = Obs'(c,\varepsilon)\ Obs'(c,c)\ Obs'(a,cc)\ Obs'(b,cca)\ Obs'(c,ccab) = c.c.a.\varepsilon.\varepsilon = cca$.

Initially, all events are perceivable. However, when event a takes place, it obscures all occurrences of events $b\ or\ c$, allowing just the observation of a. Once a has been spotted, the mask reveals its concealment by allowing a, b, and c to be visible once more.

Consider $\Sigma_o \subseteq \Sigma$, if DP_{T_D} is a dynamic projection where this projection defines a constant mapping making events in Σ_o observable, then we extend the dynamic projection as $DP_{T_D} = DP_{\Sigma_o}$. For this, we present the dynamic mask encoding a dynamic projection using automata.

Definition 8: A mask is a complete and deterministic labeled automaton $LG_M = (Q_M, Q_{M0}, \Sigma, \Gamma, T)$ for a LTS LG where Q_M is the set of states, Q_{M0} is the initial states $q_0 \in Q_{M0}$, Σ is the set of events, Γ: $Q_M \to 2^{\Sigma}$ is a labeling function that specifies the set of events that the mask keeps observable at state q. T: $Q \times \Sigma^* \to Q$ is the transition

function. The transition started by q, ended by state q' and execute the action a, is denoted by $T(q,a) = q'$ correspond to the transition relation $(q,a,q') \in \rightarrow$ in $LG = (Q, Q_0, \Sigma, \rightarrow)$. The transition function is defined as follows:

$$\begin{cases} T(q,\varepsilon) = q \\ T(q,w.a) = T(T(q,w),a) \\ = \begin{cases} T(q',a) \text{ if } a \in \Gamma(T(q,w)) \text{ and } T(q,w)) = q' \\ T(q,w) \text{ if } a \notin \Gamma(T(q,w)) \end{cases} \end{cases} \quad (4)$$

Therefore, each dynamic projection DP_{T_D} can be associated with a dynamic mask DP_{Σ_o}.

Example 3: According to Example 1, we determine the mask as shown in Figure 3. Let w be an execution, the dynamic projection of $w = ccabc$ is presented as follows (where $\forall\, w \in Lang(c^*a)$, $T_D(w) = \{a\}$ and $\forall\, w \in \Sigma^*, T_D(w) = \{a,b,c\}$:

- $DP_{T_D}(w) = DP_{T_D}(ccabc) = DP_{T_D}(ccab).c$ and $w = ccab \notin Lang(c^*a)$ then $T_D(w) = \{a,b,c\}$ and c is an observable action.
- $DP_{T_D}(w) = DP_{T_D}(ccab).c = DP_{T_D}(cca)$ and $w = cca \in Lang(c^*a)$ then $T_D(w) = \{a\}$ and b, c are unobservable events. $DP_{T_D}(w) = DP_{T_D}(cca) = cca$.
- In the same way, $DP_{\Sigma_o}(w) = DP_{\Sigma_o}(ccabc) = DP_{\Sigma_o}(ccab)$ and $c \notin \Gamma(T(q,ccab))$.
- $DP_{\Sigma_o}(w) = DP_{\Sigma_o}(ccab) = DP_{\Sigma_o}(cca)$ and b $\notin \Gamma(T(q,cca))$.
- $DP_{\Sigma_o}(w) = DP_{\Sigma_o}(cca) = DP_{\Sigma_o}(cc).a = DP_{\Sigma_o}(c).ca = cca$, $a \in \Gamma(T(q,cc))$ and $c \in \Gamma(T(q,c))$.

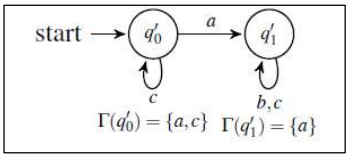

Figure 3: A dynamic mask according to Figure 2

To summarize this part, we define the dynamic projection for a LTS $LG = (Q, Q_0, \Sigma, \rightarrow)$ and a correspond mask $LG_M = (Q_M, Q_{M0}, \Sigma, \Gamma, T)$ as follows where $Q_M = \{q_0\}$:

$$\begin{cases} DP_{\Sigma_o}(\varepsilon) = \varepsilon; \\ DP_{\Sigma_o}(w.a) = DP_{\Sigma_o}(w).a & \text{if } a \in \Gamma(T(q_0,w)) \\ DP_{\Sigma_o}(w.a) = DP_{\Sigma_o}(w) & \text{otherwise} \end{cases} \quad (5)$$

4.3. Orwellian projection

Orwellian observation is based on the prefix and suffix of the trace and an observer that can deduce the knowledge to reinterpret events. This projection is studied in [5]. The interface between a system and an observer is specified by the set of observable events $\Sigma_o \subseteq \Sigma$ and the subset of downgrading events $\Sigma_d \subseteq \Sigma$. Thus, The Orwellian projection is defined for the discrete sequence $w = a_1 a_2 \ldots . a_n$, denoted by P_{Σ_o,Σ_d}. Formally, $P_{\Sigma_o,\Sigma_d}: \Sigma^* \rightarrow {\Sigma_o}^*$

$$\begin{cases} P_{\Sigma_o,\Sigma_d}(\varepsilon) = \varepsilon; \\ P_{\Sigma_o,\Sigma_d}(w.a) = w.a & \text{if } a \in \Sigma_d \\ P_{\Sigma_o,\Sigma_d}(w.a) = P_{\Sigma_o,\Sigma_d}(w).a & \text{if } a \in \Sigma_o \\ P_{\Sigma_o,\Sigma_d}(w.a) = P_{\Sigma_o,\Sigma_d}(w) & \text{otherwise} \end{cases} \quad (6)$$

Orwellian functions pertain to an observer with the capacity for unlimited memory to retain labels and the ability to employ knowledge of other labels, whether acquired before or after, to reinterpret a label. The Orwellian projection can be expressed in another form by [32].

Definition 9: [32] According to observation function $Obs: \Sigma^* \rightarrow {\Sigma_o}^*$, $\Sigma_o \subseteq \Sigma$ and $\Sigma_d \subseteq \Sigma$. The Orwellian projection is a mapping $Obs': \Sigma \times \Sigma^* \rightarrow \Sigma_o \cup \{\varepsilon\}$ such that $\forall\, w = a_1 a_2 \ldots . a_n \in \Sigma^*, Obs(w) = Obs'(a_1,w)Obs'(a_2,w) \ldots Obs'(a_n,w)$

Example 4: Consider the automaton A shown in Figure 2 with $\Sigma_o = \{a\}$, $\Sigma_u = \{c\}$ and $\Sigma_d = \{b\}$. Table 1 represents the Orwellian projection of executions.

Table 1: Executions with Orwellian projection

Events (read part)	Orwellian observation	Events (read part)	Orwellian observation
c	***ε***	***c***	***ε***
cc	***ε***	***ca***	***a***
cca	***a***	***cab***	***cab***
ccac	***a***	***cabc***	***cab***
ccacb	***ccacb***	***cabcc***	***cab***
ccacbc	***ccacb***	***cabccb***	***cabccb***

The Orwellian concept is extended to the m-Orwellian category further by incorporating modern technologies and methods of mass surveillance. M-Orwellian observation involves the use of advanced monitoring tools, data analytics, and interconnected systems to exert pervasive control. It often raises concerns about privacy, data ethics, and the potential misuse of technology for surveillance purposes.

Orwellian observation is defined for a fixed number of observation events that are called m-Orwellian observation. The number of observable events before a downgrading action is less than or equal to m. The m-Orwellian projection can be defined as follows by [32].

Definition 10: [32] According to observation function $Obs: \Sigma^* \rightarrow {\Sigma_o}^*$, $\Sigma_o \subseteq \Sigma$, $\Sigma_d \subseteq \Sigma$ and $m \in \mathbb{N}^*$.

The m-Orwellian projection is a mapping $Obs' : \Sigma \times \Sigma^* \to \Sigma_o \cup \{\varepsilon\}$ such that $\forall\, w = a_1\, a_2 \ldots\ldots a_n \in \Sigma^*$, $Obs(w) = Obs'(a_1, j_1)\, Obs'(a_2, j_2) \ldots Obs'(a_n, j_n)$ where $\forall\, w \in \| 1, n \|$, $j_p = a_{\max(1,p-m+1)}\, a_{\max(1,p-m+1)+1} \ldots a_{\min(1,p+m-1)}$

4.4. Timed static projection

Timed static projection can be reflected in narratives that focus on pivotal moments in a society's history, capturing the static essence of each era while acknowledging the temporal transitions between them. This approach allows authors to explore the nuanced interplay between stability and change within complex systems, offering readers a richer understanding of the narrative's temporal landscape.

The static projection is expanded into a timed sequence on a real-time system. Formally, $TP_{\Sigma_o} : (\Sigma \times \mathbb{Q}_+)^* \to (\Sigma_o \times \mathbb{Q}_+)^*$ is defined by:

$$\begin{cases} TP_{\Sigma_o}((\varepsilon, \gamma)) = (\varepsilon, \gamma); \\ TP_{\Sigma_o}(u.(a, \gamma)) = TP_{\Sigma_o}(u) & if\ a \in \Sigma_u \\ TP_{\Sigma_o}(u.(a, \gamma)) = TP_{\Sigma_o}(u).(a, \gamma) & otherwise \end{cases} \quad (7)$$

where $u \in (\Sigma \times \mathbb{Q}_+)^*, a \in \Sigma$, $\gamma \in \mathbb{Q}_+$ and ε is the empty string. The notion $[u]_{\Sigma_o}$ is extended that represents the set of all timed executions having the same projection as u.

5. DISCRETE OPACITY WITH STATIC PROJECTION

The opacity properties are introduced for the first time for the analysis of cryptographic protocols in [33, 36]. Next, the opacity is defined in the communication network. In [4, 24], opacity has been introduced in DES when the system can be modeled by Petri nets. In [5], previous work has been deepened by studying opacity in more general systems and which are labelled LTS.

The opacity parameters are determined by the following conditions: (1) S contains a collection of confidential information; (2) the intruder is an observer of S who possesses complete understanding of the architecture of A. An opaque system is characterized by the presence of a non-secret behavior that is indistinguishable from a secret behavior, hence making it impossible for an outsider to discern the secret behavior. Consequently, the invader remains uncertain about the occurrence of the secret. Building on existing research, [4] delves into opacity for DES by FSA with partial transition observability. Previous literature, however, categorizes formal LTS opacity definitions into two main families.

5.1. Language Based Opacity

The concept of LBO was initially introduced in [9]. The secret behavior is defined by a language called $LangS$, which is a subset of Σ^*. Additionally, it is known as trace-based opacity. The system is opaque w.r.t. $LangS$ and the projection map P_{Σ_o} if the intruder should be unable to ascertain if the word is in the secret language or not. Yet, in [11], The LBO is specified across two sub languages of the system, $(Lang1, Lang2) \subseteq (Lang(LG, Q_0 0))^2$. The term "opaque" is used between $Lang1$ and $Lang2$ under the projection map P_{Σ_o}, if the intruder has an ambiguity between every string in $Lang1$ with some strings in $Lang2$ under the projection map. Consider that $Lang = LangS \cup LangNS$ is a language where $LangS$ and $LangNS$ are secret and non-secret languages.

Definition 11: The secret language $LangS$ is said **language-based opaque** under P_{Σ_o} if: $\forall\, w \in LangS, \exists\, w' \in LangNS$ such that $P_{\Sigma_o}(w) = P_{\Sigma_o}(w')$

A secret language is considered opaque if every string w in the secret language, $LangS$, has a corresponding string w', having the same projection, in $LangNS$. In other words, we present the following Lemma.

Lemma 1: The secret language $LangS$ is said **language-based opaque** under P_{Σ_o} iff: $P_{\Sigma_o}(\text{LangS}) \subseteq P_{\Sigma_o}(LangNS)$

Definition 12: The secret language $LangS$ is said **weakly opaque** under P_{Σ_o} if: for some $w \in LangS, \exists\, w' \in LangNS$ such that $P_{\Sigma_o}(w) = P_{\Sigma_o}(w')$

The secret language is considered weakly opaque if there is a string w in $LangS$ such that there is another string w' in $LangNS$ that has the same projection. We give a more formal notation in Lemma 2.

Lemma 2: The secret languages $LangS$ is said **weakly opaque** under P_{Σ_o} iff: $P_{\Sigma_o}(\text{LangS}) \cap P_{\Sigma_o}(LangNS) \neq \emptyset$

Definition 13: The secret language $LangS$ is said **no-opaque** under P_{Σ_o} if $LangS$ is not weakly opaque under P_{Σ_o}.

The secret language is no-opaque if for each string w in $LangS$, there not exists a string w' in $LangNS$ with the same projection. In other words, we present the following Lemma.

***Lemma 3**:* The secret languages $LangS$ is said **no-opaque** under P_{Σ_o} iff: $P_{\Sigma_o}(\text{LangS}) \cap P_{\Sigma_o}(LangNS) = \emptyset$

Example 5: We consider the secret language $LangS = Lang(\text{aba(cba)}^*) \cup Lang(\text{ca(bac)}^*)$ and the non-secret language $LangNS = Lang(\text{a(bac)}^*) \cup Lang(\text{a(bac)}^* b) \cup Lang(\text{a(bac)}^* bab) \cup Lang(\text{ca(bac)}^* b) \cup Lang(\text{ca(bac)}^* ba) \cup Lang(\text{ca(bac)}^* baa)$ where $\Sigma_o = \{a, b\}$ and $\Sigma_u = \{c\}$ are the set observable and unobservable actions. The intruder is not sure of each word that is observationally equivalent to a word in secret language or equivalent to a word in non-secret language. Therefore, the secret language $LangS$ under P_{Σ_o}. is language-based opaque.

5.2. State Based Opacity

The state-based approach is associated with the covert actions of a single state or a group of states. Multiple opacity properties have been established based on the type of secret being considered. Let LG be a LTS, with $\Sigma_o \subseteq \Sigma$ and $S \subseteq F$ as secret states where $F \subseteq Q$ is final states.

1) Current-State Opacity or CSO:

CSO is initially presented in [4] for the application of Petri Nets. The state property pertains to the inclusion of the system's final state inside a specific set of undisclosed states. This property was adapted to LTS in [2,4, 28, 35].

Definition 14: The secret S is said **current-state opaque** under P_{Σ_o} if: $\forall w \in Lang(LG, Q_0, S), \exists w' \in Lang(LG, Q_0, Q - S); P_{\Sigma_o}(w) = P_{\Sigma_o}(w')$

The system is deemed opaque in its current state if the intruder, although possessing comprehensive knowledge of the system's architecture and making partial observations of its behavior, is unaware of the true essence of the outcome. Definition 5 presents a direct consequence of Lemma 4:

***Lemma 4**:* The secret S is said **current-state opaque** under P_{Σ_o} iff:

$$P_{\Sigma_o}(\text{Lang}(LG, Q_0, S)) \subseteq P_{\Sigma_o}(Lang(LG, Q_0, Q - S))$$

Example 6: According to Example 1, we built a LTS system LG corresponding to the secret and non-secret languages as shown in Figure 4, with, $Q=\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\}$ is the set of states, $\Sigma_o = \{a, b\}$ and $\Sigma_u = \{c\}$. If we consider that $S = \{q_3\}$, then S is a CSO because the intruder confuses between the word aba and $caba$. Thus, the outsider is not certain if the system is in $q_3 \in S$ or in $q_8 \in Q - S$. But, if $S = \{q_3, q_9\}$, then S is not a CSO. The outsider is certain whether the system is in $q_9$when $cabaa$ is executed.

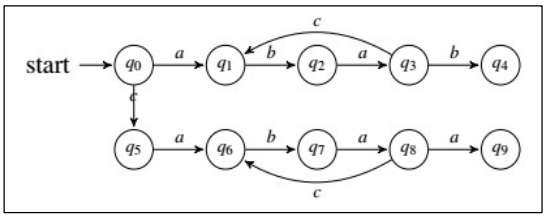

Figure 4: Opacity example

2) The Initial-State Opacity or ISO

ISO is defined within Petri Nets models in [4]. Thus, this property is an extension of LTS in [7, 22]. ISO refers to a state property that pertains to the inclusion of the system's starting state in a collection of confidential states. If the intruder is unable to conclude if the initial state of the system is a secret or not, then the system is opaque in its initial state.

***Definition 15**:* The secret S is said **initial-state opaque** under P_{Σ_o} if: $\forall w \in Lang(LG, S), \exists w' \in Lang(LG, Q_0 - S)$ such that $P_{\Sigma_o}(w) = P_{\Sigma_o}(w')$

The system is completely opaque in its initial state. For each individual word w that comes from a confidential state $q_0 \in S \subseteq Q_0$, there is another word w' from a non-confidential initial state $q_0 \in Q_0 - S$, such that w and w' are observationally similar. Thus, the intruder is unable to ascertain if the system originated from a confidential state q_0 or from a non-confidential state q'. Formally, ISO can be defined in the following Lemma.

***Lemma 5**:* The secret S is said **initial-state opaque** under P_{Σ_o} iff:

$$P_{\Sigma_o}(\text{Lang}(LG, S)) \subseteq P_{\Sigma_o}(Lang(LG, Q_0 - S))$$

Example 7: We consider the LTS system LG as shown in Figure 4 and $Q_o = \{q_0, q_5\}$. If $S = \{q_0\}$, then S is initial-state opaque. The set of word starting from q_0 is $Lang(LG, S) = Lang(\text{a(bac)}^*) \cup Lang(\text{a(bac)}^* b) \cup Lang(\text{a(bac)}^* bab)$. The set of the words starting from q_5 is $Lang(LG, S) = Lang(\text{ca(bac)}^* b) \cup Lang(\text{ca(bac)}^* ba) \cup Lang(\text{ca(bac)}^* baa)$. If $S = \{q_5\}$, then S isn't initial-state opaque. The outsider is convinced that the system is initiated by q_5 and ending by $q_9$when the discrete word $abaa$ is executed.

The efficient resolution of both CSO and ISO can be achieved in bounded Petri nets by utilizing a concise depiction of the reachability graph [15].

3) Initial-and-Final-State Opacity or IFO

IFO is a state property that is related to both system's initial and final states [13]. This property defines secret states as a pair of states.

Definition 16: The secret S is said **initial-and-final state opaque** under P_{Σ_o} if: $\forall(q_i, q_j) \in S$, $and\ \forall\ w \in Lang(LG, q_i, q_j), \exists(q'_i, q'_j) \in (Q_0 \times Q) - S,\ and\ \exists\ w' \in Lang(LG, q'_i, q'_j);\ P_{\Sigma_o}(w) = P_{\Sigma_o}(w')$

The system is initial-and-final-state opaque if for every word w starting from q_i and ending at q_j, there exists another word w' beginning from q'_i and terminated at q'_j such that w and w' are the same observationally. Thus, the outsider is unable to ascertain the secrecy of the state couple. We propose another definition for this property in Lemma 6.

Lemma 6: The secret S is said **initial-and-final opaque** under P_{Σ_o} iff:

$$P_{\Sigma_o}\left(\cup_{(q_0,q_f)\in S} Lang(LG, q_0, q_f)\right) \subseteq P_{\Sigma_o}(\cup_{(q'_0,q'_f)\in(Q_0\times Q)-S} Lang(LG, q'_0, q'_f))$$

According to the previous Lemma, IFO is similar to strong language-based opacity where LangS = $\cup_{(q_0,q_f)\in S} Lang(LG, q_0, q_f)$ is the secret language and LangNS = $\cup_{(q'_0,q'_f)\in(Q_0\times Q)-S} Lang(LG, q'_0, q'_f)$ is the non-secret language.

Example 8: We recall LG as shown in Figure 4 and $Q_0 = \{q_0, q_5\}$. If $S = \{(q_0, q_5)\}$, then S is initial-and-final-state opaque. The outsider is never certain whether the word aba corresponding to the secret state pair (q_0, q_3).

4) K-step opacity:

It was initially presented in [4], and later in [7]. It allows for the verification of whether a system has a concealed state K that could be observed through past events. Two forms of this property are presented in [3] strong and weak.

a) *K-step weakly opacity*

Definition 17: The secret S is said **K-weakly opaque** under P_{Σ_o} if: $\forall\ w \in Lang(LG, Q_0), \forall\ w_1 \preccurlyeq w$ and $|w_1 - w| \leq K$ such that $w_1 \in Lang(LG, Q_0, S)\ \exists\ v \in Lang(LG, Q_0), \forall\ v_1 \preccurlyeq v$ and $|v_1{\_}v| \leq K$; $P_{\Sigma_o}(v) = P_{\Sigma_o}(w)$, $P_{\Sigma_o}(v_1) = P_{\Sigma_o}(w_1)$ and $v_1 \in Lang(LG, Q_0, Q - S)$

The system is K-weakly opaque if for every discrete word w where K longest of its prefixes lead to a secret state, there is another compatible discrete word where the K longest of its prefixes do not lead to a secret state.

This definition is reformulated in [1], for every execution w and where w_s is the prefix of w and the difference between the observable executions is less or equal to K, there is w' and w'_s executions have the same projection as w and w_s where w'_s is not a secret execution. In other words, we present the following Lemma.

Lemma 7: The secret S is said **K-weakly opaque** under P_{Σ_o} iff: $P_{\Sigma_o}(\text{Lang}(LG, Q_0, S)) \subseteq P_{\Sigma_o}(Lang(LG, Q_0, Q - S))$ and $P_{\Sigma_o}(\text{Lang}_k(LG, S)) \subseteq P_{\Sigma_o}(\text{Lang}_k(LG, Q - S))$

The K-weakly opacity is similar to language-base opaque where $LangS = \{w, w \in Lang(LG, Q_0), \forall w_1 \in Lang(LG, Q_0, S);\ w_1 \preccurlyeq w\ and\ |w - w_1| \leq K\}$ is the secret language and $LangNS = \{w, w \in Lang(LG, Q_0), \forall w_1 \in Lang(LG, Q_0, \text{Q} - \text{S});\ w_1 \preccurlyeq w\ and\ |w - w_1| \leq K\}$ is the non-secret language.

Example 9: Let LG as shown in Figure 4. If we consider that $S = \{q_3, q_6\}$ and K=2, then S is K-weakly opaque. However, if K=3, then S is not K-weakly opaque because there is not observationally equivalent to the word $cabaa$. The outsider concludes where the system passes via the secret state q_6.

b) *K-step strong opacity*

It acts as a detective, scrutinizing the system's recent history (the last K observable actions) to uncover any hidden visits to secret states. It ensures that even a cunning observer, armed with partial knowledge, can't definitively tell if the system dipped into the shadows of secrecy within this timeframe.

Definition 18: The secret S is said **K-strongly opaque** under P_{Σ_o} if: $\forall\ w \in Lang(LG, Q_0, 0),\ \exists\ v \in Lang(LG, Q_0, 0);\ P_{\Sigma_o}(v) = P_{\Sigma_o}(w)\ \forall\ v_1 \preccurlyeq v$ with $|v_1{\_}v| \leq K$ and $v_1 \in Lang(LG, Q_0, Q - S)$

A system boasts K-strong opacity if, for every possible behavior sequence, there's another identical-looking one (same "projection") that avoids secret states within the last K observed actions. This ensures even a watchful observer can't definitively tell if the system dipped into the shadows of secrecy recently. Definition 9 is formulated in Lemma 8.

Lemma 8: The secret S is said **K-strongly opaque** under P_{Σ_o} iff: $P_{\Sigma_o}(\text{Lang}(LG, Q_0, S)) \subseteq P_{\Sigma_o}(Lang(LG, Q_0, Q - S))$ and $P_{\Sigma_o}(\text{Lang}_k(LG, Q_0, S)) \subseteq P_{\Sigma_o}(\text{Lang}(LG, Q_0, S))$

Example 10: Let LG be a LTS as shown in Figure 4. If $S = \{q_3, q_6\}$ and K=1, then S is K-strongly

opaque. However, if K=2, then S is not K-strongly opaque because there is no word that does not pass through any secret.

The property of K-step opacity is translated on **trace-based K-step opacity** (or trajectory) when the system has recently been in a specific state, in [14, 20]. This property is defined as follows: for any given word w, there exists at least one discrete word that is observationally similar to w. Additionally, the states visited while generating the last K actions are exclusively non-secret states in w. The distinction between K-step opacity and trace-based K-step opacity is in the timing of when the system's state is revealed. Hence, if the system exhibits trace-based K-step opacity, it likewise demonstrates K-step weak opacity. The concept of K-step opacity has been expanded to include infinite-step opacity in the works cited [18, 20].

Definition 19: The secret S is said **weakly infinite-step opaque** under P_{Σ_o} if: $\forall w \in Lang(LG, Q_0)$, ; $\forall w' \in Lang(LG, Q_0, S)$, $such\ that\ w' \preccurlyeq w$, $\exists v \in Lang(LG, Q_0)$ and $\exists v' \in Lang(LG, Q_0, Q - S)$; $P_{\Sigma_o}(v) = P_{\Sigma_o}(w)$ and $P_{\Sigma_o}(v') = P_{\Sigma_o}(w')$.

The system is infinite-step opaque if for every w, the outsider is unable to deduce that the system was previously in a concealed state.

Lemma 9: The secret S is **infinite-step opaque** under P_{Σ_o} iff:
$P_{\Sigma_o}(Lang(LG, Q_0, S)) \subseteq P_{\Sigma_o}(Lang(LG, Q_0, Q - S))$ and $P_{\Sigma_o}(Lang^{\infty}(LG, Q_0, S)) \subseteq P_{\Sigma_o}(Lang^{\infty}(LG, Q_0, Q - S))$

The weak infinite-step opacity is similar to language based opaque where $LangS = \{w, w \in Lang(LG, Q_0),\ \forall w' \in Lang(LG, Q_0, S)\ such\ that\ w' \preccurlyeq w\}$ is the secret language and $LangNS = \{w, w \in Lang(LG, Q_0), \forall w' \in Lang(LG, Q_0, Q - S)\ such\ that\ w' \preccurlyeq w\}$ is the non-secret language.

Definition 20: The secret S is **strongly infinite-step opaque** under the projection map P_{Σ_o} if: $\forall w \in Lang(LG, Q_0), \exists v \in Lang(LG, Q_0)\ such\ that P_{\Sigma_o}(v) = P_{\Sigma_o}(w)\ \forall v' \preccurlyeq v$ and $v' \in Lang(LG, Q_0, Q - S)$

Lemma 10: The secret S is **strongly infinite-step opaque** under P_{Σ_o} iff: $P_{\Sigma_o}(Lang(LG, Q_0, S)) \subseteq P_{\Sigma_o}(Lang(LG, Q_0, Q - S))$ and $P_{\Sigma_o}(Lang^{\infty}(LG, Q_0, S)) \subseteq P_{\Sigma_o}(\text{Lang}(LG, Q_0, S))$

Example 11: Let LG be an LTS system as shown in Figure 4. If S = $\{q_3, q_6\}$ and K=3 then S is not K-weakly opaque, then S is not infinite-step opaque.

Those notions have strong connections between each other and the transformations relationships between them.

6. TRANSFORMATION BETWEEN DIFFERENT NOTIONS OF OPACITY

The opacity property can be reduced to varying degrees of transparency with a polynomial time complexity that is defined in [13]. The relationships are presented in Figure 5.

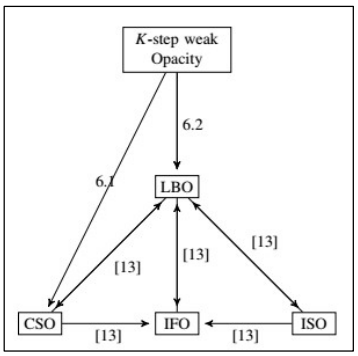

Figure 5: Transformation between notions of opacity

6.1. Transformation between K-step weak opacity and CSO

CSO is equal to K-step opacity where K = 0. Let LG be an LTS, $S_K \subseteq Q$ is the secret states and $K \in \mathbb{N}$ is a constant value. We consider that S_K is K-step weak opaque. From the K-step weak opacity, we determine S where S is the current secret states. Formally, $S = \{q_s, \forall w \in Lang(LG, Q_0)\ \exists w_1 \in Lang(LG, Q_0, S_K)$; $w_1 \preccurlyeq w, |P_{\Sigma_o}(w_1) - P_{\Sigma_o}(w)| \leq K\ and\ last(w_1) = q_s\}$ is the set of current secret states. Then, we determine the non-secret state $NS = Q - S$. To verify if S_K is K-step weak opaque, we check if every string that pass through by a secret state $q \in S_K$ has the same projection as a string that pass through by a non-secret state $q' \in NS_K$. If every string ending by $q_s \in S$ there is a string ending by $q_{ns} \in NS$ having the same observability. This approach is identical to determining whether S is current-state opaque.

6.2. Transformation from K-step weak opacity to LBO

We consider that S_K is K-step weak opaque. From the K-step weak opacity, we determine the secret language $L_S = \{w \in Lang(LG, Q_0)\ \exists w_1 \in Lang(LG, Q_0, S_K);\ w_1 \preccurlyeq w, |P_{\Sigma_o}(w_1) - P_{\Sigma_o}(w)| \leq K\}$. Similarly, we determine the non-secret language $L_{NS} = \{w \in Lang(LG, Q_0)\ \exists w_1 \in Lang(LG, Q_0)$; $w_1 \preccurlyeq w, |P_{\Sigma_o}(w_1) - P_{\Sigma_o}(w)| \leq K\ and\ w_1 \notin Lang(LG, Q_0, S)\}$. To verify if S_K is K-step weak opaque, we check if every string that pass through by

a secret state $q \in S_K$ has the same projection as a string passing through by $q' \in NS_K$, that is, if every word $w \in L_S$ has the same projection as a string $w' \in L_{NS}$. This approach is identical to determining whether L_S is language-based opaque.

7. DISCRETE OPACITY WITH DYNAMIC PROJECTION

In this section, we generalize the opacity approach by considering the notion of dynamic projections encoded by dynamic masks. Based on these assumptions, we define opacity under the dynamic projection as follows [2].

Definition 21: Let LG be a LTS and $S \times Q$ is secret states. The system S is opaque under DP_{T_D} if : $\forall\, u \in Lang(LG),\ last([u]_{T_D}) \notin S$
Where $[u]_{T_D}$ represents the set of words observationally equivalent to u under the dynamic projection map DP_{T_D}.

Example 12: Let LG be a LTS as shown in Figure 4. If $S = \{q_3\}$ is secret and the dynamic projection map as follows: $T_D(\varepsilon) = \{a\}$, $T_D(ab) = \{a\}$, $T_D(aba) = \{a, b\}$ and $T_D(u) = \{a, b, c\}$ otherwise, then S is opaque under DP_{T_D}. However, if the dynamic projection map as follows: $T_D(ab) = \{a, b\}$ and $T_D(u) = \{a, b, c\}$ otherwise, then S isn't opaque. There is not observationally equivalence to the sequence aba. Therefore, the intruder can conclude that the system is in the secret state q_3.

The issue of verification opacity is exacerbated by using dynamic projection compared to static projection. Specifically, the verification opacity problem becomes PSPACE-complete.

Dynamic projection is frequently employed to ensure opacity. The opacity is specified within the Orwellian projection map illustrated in the following section.

8. DISCRETE OPACITY WITH ORWELLIAN PROJECTION

This section examines the opacity property in relation to the Orwellian projection in [39]. The observability of w under the Orwellian projection is determined by the observability of all actions that occur prior to each downgrading action. With more simplicity, each discrete word is partitioned in two parts $D(w)$ and $C(w, Lang)$ where $D(w)$ represents the first part of w ending by the last downgrading action and $C(w, Lang)$ is the continuation of $D(w)$ and does not contain the downgrading actions. Formally, $\forall\, w \in Lang, w = D(w).C(w, Lang)$ where $D(w) \subseteq \{\varepsilon\} \cup (\overline{Lang} \cap \Sigma^*\Sigma_d)$ where $\overline{Lang}$ is the prefix of Lang ending in downgrading action and $C(w, Lang) = (\Sigma_o \cup \Sigma_u)^* \cap \overline{\overline{Lang}}$ where $\overline{\overline{Lang}}$ is a continuation of $\overline{Lang}$. We extend $D(w)$ to $D(Lang) = \{\varepsilon\} \cup (\overline{Lang} \cap \Sigma^*\Sigma_d)$. The following definition presents the opacity property under the Orwellian projection map.

Definition 22: Let LG be a LTS and S a secret state. The secret is opaque under P_{Σ_o,Σ_d} if $\forall\, u \in D(Lang(LG))$, $C(u, Lang(LG))$ is opaque under P_{Σ_o}.

Example 12: Let LG be a LTS as shown in Figure 4, $\Sigma_o = \{a\}$ and $\Sigma_d = \{b\}$. If $S = \{q_3\}$ then S isn't opaque under P_{Σ_o,Σ_d}.

9. TIMED OPACITY WITH STATIC PROJECTION

The concept of opacity is expanded to temporal settings to explore the problem of language-based opacity [21]. Timed opacity is a fascinating extension of opacity that considers the measurement of time for an intruder, where the secret is a collection of specific locations. This characteristic guarantees that the system cannot definitively determine if this sequence is present in the secret or not.

Definition 23: Let A be a timed automaton, Σ_o be a set of observable actions and $S \subseteq$ L be secret actions. The secret S is timed opaque under TP_{Σ_o} if : $\forall\, u \in TL_S(A), \exists\, u_1 \in TL_{L-S}(A)$; $TP_{\Sigma_o}(u) = TP_{\Sigma_o}(u_1)$

A system is considered timed opaque if, for every timed word u, there exists another timed word u_1 that has the same projection as u and leads to non-secret locations. Language-opacity is defined as the state of being opaque for a real-time automaton, as stated in [21, 32]. An alternative formulation of this definition can be stated as:

Lemma 12: [32] The secret S is timed opaque under TP_{Σ_o} if $\forall\, u \in TP_{\Sigma_o}(TL_S(A)),\ [u]_{T_D} \notin S$

Lemma 13: [17] Language-opacity $TP_{\Sigma_o}(TL(A) \cap TL_S) \subseteq TP_{\Sigma_o}(TL(A) - TL_S)$

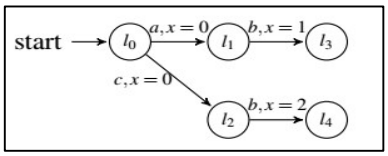

Figure 6: Example of TA in [32]

Example 13: Let A be a TA shown in Figure 6 where $\Sigma_o = \{b\}$ and $S = \{l_2\}$. Then, A isn't opaque. The outsider is certain that the system in l_2 when he

observes the projection of word b that occur at time 1 under TP_{Σ_o}.

10. VERIFICATION AND DECIDABITY OF OPACITY

The opacity property requires that a system has a hidden secret behavior from an intruder. This property is verified using different frameworks such that:

- Labeled Petri Nets (LPN) is used to verify ISO in [15, 26], CSO in [16] and Language-based opacity in [31].
- Symbolic Observation Graph (SOG) is used to verify simple opacity in [38], K-step weak and strong opacity in [15].
- Labeled Transition System (LTS) is followed in several research such as [2,7,9,18,20,22]. The opacity properties are verified on the building of the observer automaton.

To assess the opacity of these systems, it is essential to establish if a system is opaque in relation to a specific secret [11 13, 22, 24, 31]. Numerous studies have been conducted on the decidability of the property of opacity in DESs, as evidenced by the works in [4, 35]. For instance, the decidability of CSO, ICO, and language opacity in LTS has been demonstrated. ISO is decidable for bounded Petri nets in [4] and undecidable in Petri nets unbounded in [35]. The decidability and complexity results are synthesized, in [29], related to opacity problems for such discrete system model and projection map.

Timed opacity is generally undecidable for timed automata and event recording automata used in real-time systems. The problem of determining timed opacity using non-deterministic Timed Automata is impossible to solve, but it can be solved with Event Recording Automata. The problem of language opacity and the problem of starting opacity are determinable for Real-Time Automata, as stated in reference [17].

11. COMPARISON WITH EXIST WORKS

This section recapitulates the different definitions of opacity for both discrete and real-time systems. We present a comparative overview using Table 2. On one hand, the table shows established notions from previous research. On the other hand, it showcases the corresponding definitions based on our proposed lemma introduced earlier.

Table 2: Executions with Orwellian projection

Projection	Opacity properties	Existing works	Our work
Static projection	Language Based Opacity	[9, 11] *Definition 11*	Lemma 1
	weakly opaque	[11] *Definition 12*	Lemma 2
	No Opacity	[11] *Definition 13*	Lemma 3
	Current-State Opacity	[2,4,28,35] *Definition 14*	Lemma 4
	Initial State Opacity	[4,7,22] *Definition 15*	Lemma 5
	Initial-and-Final State Opacity	[13] *Definition 16*	Lemma 6
	K-step Weakly Opacity	[1,3,4,7] *Definition 17*	Lemma 7
	K-step Strongly Opacity	[3,4,7] *Definition 18*	Lemma 8
	Weakly Infinite Step Opacity	[18,20] *Definition 19*	Lemma 9
	Strongly Infinite Step Opacity	[18,20] *Definition 20*	Lemma 10
Dynamic projection	Opacity	[2] *Definition 21*	-
Orwellian projection	Opacity	[39] *Definition 22*	-
Timed Static projection	Opacity	[21,32] *Definition 23*	Lemma 12

Note that the complexity of those notions of opacity remains the same complexity because the verification of each opacity property is based on the verification of inclusion problem.

Figure 7 illustrates the relation between different publications addressing the opacity properties, described in Section 5. Each arrow, linking paper X to paper Y in the diagram, means that paper Y introduces a new notion of opacity based on obtained result in paper X.

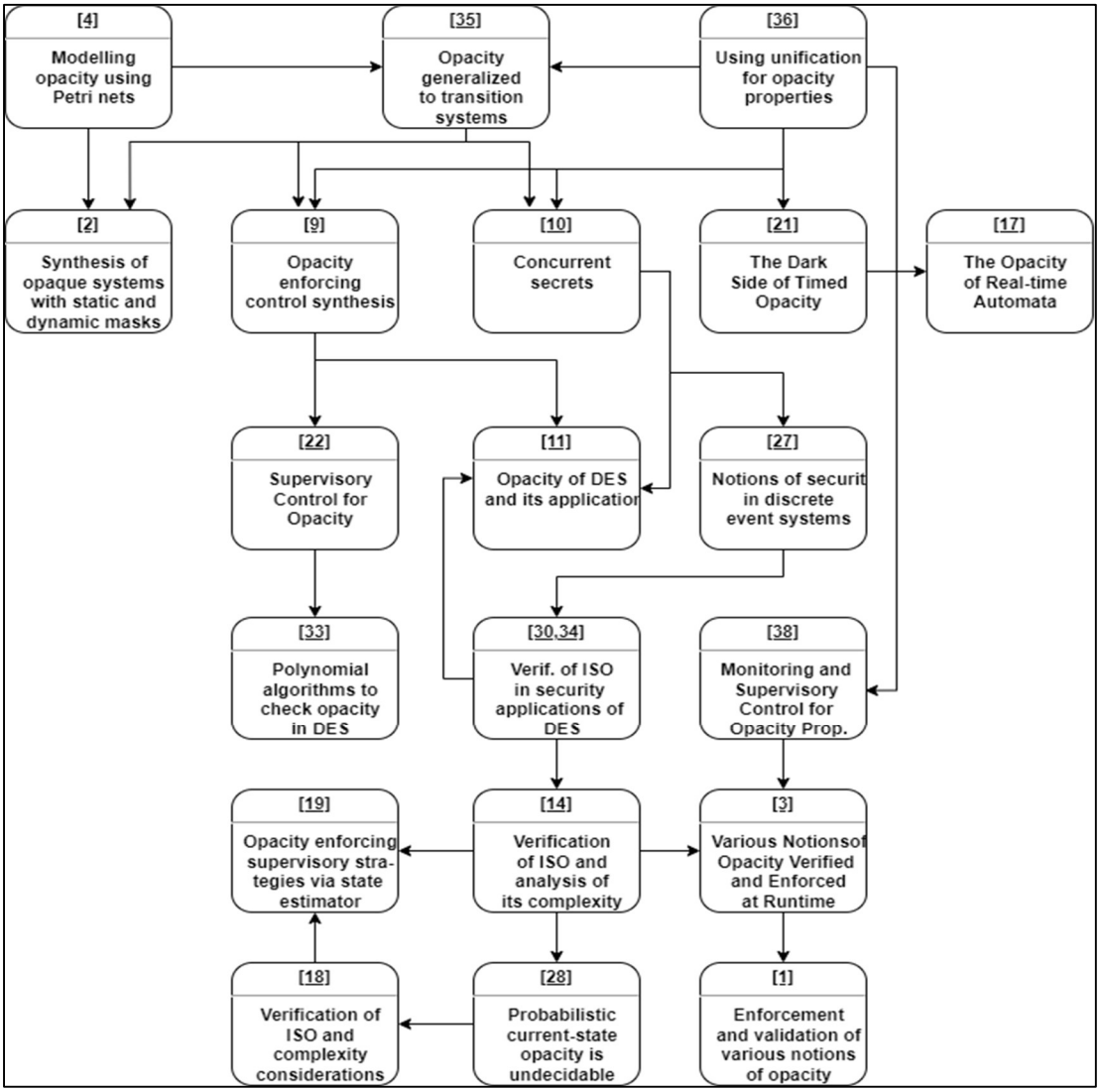

Figure 7: Reference Graph Between Opacity Notions

12. CONCLUSION

This paper presented a unified framework for defining opacity properties applicable to both discrete and real-time systems. This framework addresses a key challenge in the field of opacity research - the difficulty of comparing and analyzing opacity properties across various system models and observation scenarios. By leveraging language inclusion problems as a foundation, the framework allows researchers to analyze opacity properties in a consistent manner, regardless of the underlying system type or observation setting. This not only simplifies analysis but also facilitates the identification of connections between existing opacity formalisms. Furthermore, the paper establishes a foundation for future research by compiling existing decidability results for these unified opacity concepts and outlining potential avenues for exploring verification methods under various conditions.

In conclusion, this work offers a significant contribution to the field of opacity research. The proposed unified framework promotes a more comprehensive understanding of opacity properties in security systems. It empowers researchers to effectively compare different opacity concepts, paving the way for advancements in this crucial area of security analysis. It also opens exciting avenues for future exploration.

Building upon this foundation, future work can delve into areas such as:

- Extending the framework to incorporate additional opacity properties beyond those currently supported.

- Developing automated verification techniques specifically tailored to the unified opacity framework.
- Investigating the applicability of the framework to analyze opacity in even more complex system models, including distributed and hybrid systems.
- Exploring the potential for leveraging the framework in practical security analysis tools for real-world systems.

By pursuing these avenues, researchers can further refine and extend the power of the unified framework, leading to a deeper understanding of opacity and its role in securing complex systems.

REFERENCES

[1] Y. Falcone, H. Marchand, "Enforcement and validation (at runtime) of various notions of opacity". *Discrete Event Dynamic Systems: Theory and Applications*, 2014.

[2] F. Cassez, J. Dubreil, H. Marchand, "Synthesis of opaque systems with static and dynamic masks". *Formal Methods in System Design*, Vol. 40, 2012, pp. 88–115.

[3] Y. Falcone, H. Marchand, "Various notions of opacity verified and enforced at Runtime", *INRIA*, 2010.

[4] J. Bryans, M. Koutny, P. Ryan, "Modelling opacity using Petri nets", *Electronic Notes in Theoretical Computer Science*, Vol. 121, 2005, pp. 101–115.

[5] R. Alur, D. Dill, "A theory of timed automata", *Theoretical Computer Science* 2nd ed., vol. 3, 1994, pp. 183–235.

[6] R. Alur, P. Cerny, S. Zdancewic, "Preserving secrecy under refinement", *Automata, Languages and Programming*, 2006, pp. 107–118.

[7] A. Saboori, C. N. Hadjicostis, "Verification of initialstate opacity in security applications of DES", *Discrete Event Systems*, 2008, pp. 328–333.

[8] K. Kobayashi, K. Hiraishi, "Verification of opacity and diagnosability for pushdown systems", *Journal of Applied Mathematics*, 2013.

[9] J. Dubreil, P. Darondeau, H. Marchand, "Opacity enforcing control synthesis", *Discrete Event Systems*, 2008, pp. 28–35.

[10] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, P. Darondeau, "Concurrent secrets. Discrete Event Dynamic Systems", vol. 17, 2007, pp. 425–446.

[11] F. Lin, "Opacity of discrete event systems and its applications", *Automatica*, vol. 47, 2011, pp. 496–503.

[12] R.M. Kellery, "Formal Verification of Parallel Programs", *Communications of the ACM*, vol. 19, 1976, pp. 371–384.

[13] Y. Wu, S. Lafortune, "Comparative analysis of related notions of opacity in centralized and coordinated architectures", *Discrete Event Dynamic Systems*, pp. Saboori, C. N. Hadjicostis, "Notions of security and opacity in discrete event systems". IEEE, Conference on Decision and Control, Vol. 3, 2013, pp.307–339.

[14] A. Saboori, C. N. Hadjicostis. Verification of infite-step opacity and analysis of its complexity. Dependable control of discrete systems, Vol. 2, 2009, pp. 46–51.

[15] Y. Tong, Z. Li, C. Seatzu, A. Giua, "Decidability of opacity verification problems in labeled Petri net systems". *Automatica*, Vol. 80, 2017, pp. 48–53.

[16] X. Cong, M. P. Fanti, A. M. Mangini, Z. Li, "On-line verification of current-state opacity by Petri nets and integer linear programming", *Elsevier Science direct*, Vol. 94, 2018, pp. 205–213.

[17] L. Wang, N. Zhan, J. An, "The opacity of Real Time Automata", IEEE Transactions on Computer-Aided Design of Automata, Vol. 37, 2018, pp 2845–2856.

[18] A. Saboori, C. N. Hadjicostis, "Verification of infinite step opacity and complexity considerations", *IEEE Transactions on Automatic Control*, Vol. 57, 2012, pp. 1265–1269.

[19] A. Saboori, C. N. Hadjicostis, "Opacity enfonrcing supervisory strategies via state estimatorconstructions", *Automatic Control*, Vol. 57, 2012, pp. 1155–1165.

[20] X. Yina, S. Lafortune, "A new approach for the verification of infinite-step and K-step opacity using twoway observers", *Elsiver Automatica*, 2018.

[21] Franck Cassez. The Dark Side of Timed Opacity. International Conference on Information Security and Assurance, Springer, Vol. 5576, 2009, pp. 21–30.

[22] J. Dubreil, Philippe Darondeau and Herve Marchand. Supervisory Control for Opacity. IEEE, Transactions on Automatic Control, Vol. 55, 2010, pp. 1089–1100.

[23] S. Chedor, C. Morvan, S. Pinchinat, H. Marchand, al., "Analysis of partially observed

recursive tile systems", Workshop on Discrete Event Systems, 2012, pp.265-271.
[24] Y. Tong, Z. Li, C. Seatzu, A. Giua, "Verification of state-based opacity using Petri nets". Transactions on Automatic Control, 2017.
[25] F. Cassez, Tripakis S. "Fault diagnosis with static or dynamic diagnosers", Fundamenta Informatica, 2008, pp 97–540.
[26] Y. Tong, Z. Li, C. Seatzu, A. Giua, "Verification of Initial-State Opacity in Petri Nets", 54th IEEE Conference on Decision and Control (CDC), 2015.
[27] A. Saboori, C. N. Hadjicostis, "Notions of security and opacity in discrete event systems". IEEE, Conference on Decision and Control, 2007, pp.5056–5061.
[28] A. Saboori, C. N. Hadjicostis. Probabilistic current-state opacity is undecidable. International Symposium on Mathematical Theory of Networks and Systems MTNS, 2010, pp. 477–483.
[29] R. Jacob, J. J. Leasage, J. M. Faure, "Overview of DESs Opacity: models, validation and quantification", Annual Reviews in Control, 2016.
[30] A. Saboori, C. N. Hadjicostis, "Verification of initial state opacity in security applications of discrete event systems", Information Science, 2013, pp. 115–132.
[31] Y. Tong, Z. Li, C. Seatzu, A. Giua, "Verification of language-based opacity using Petri nets using verifier". American Control Conference, 2016, pp. 757-763.
[32] J. Bryans, M. Koutny, Mazare L., P. Ryan, "Opacity generalised to transition systems", International Workshop on Formal Aspects in Security and Trust, 2005, pp. 81–95.
[33] B. Zhang, S. Shu, F. Lin, "Polynomial algosurithms to check opacity in DESs", Control and Descion Conference, 2012, pp. 763-769.
[34] A. Saboori, C. N. Hadjicostis. Verification of initial-state opacity in security applications of discrete event systems. Information Science, 2013, pp. 115–132.
[35] J. W. Bryans, M. Koutny, L. Mazare, P. Y. Ryan, "Opacity generalised to transition systems", in International Journal of Information Security, 2008, pp. 421–435.
[36] L. Mazare, "Using unification for opacity properties", Proceedings of Workshop on Information Technology and Systems, Vol. 4, 2004, pp. 165–176.
[37] T. A. Henzinger, Z. Manna, A. "Pnueli. Timed transition systems", *Computer Science*, 1992, pp. 226–251.
[38] J. Dubreil, "Monitoriting and Supervisory Control for Opacity Properties. University of Rennes 1, 2009.
[39] John Mullins, Moez Yeddes. Opacity with Orwellian Observers and Intransitive Non-interference. Discrete Event Dynamic Systems, vol. 12, pp. 344–349, (2014).