mathematics

MDPI

Article

Security Quantification for Discrete Event Systems Based on the Worth of States

Sian Zhou [1], Jiaxin Yu [2], Li Yin [3,*] and Zhiwu Li [3]

[1] School of Computer Science and Engineering, Macau University of Science and Technology, Avenida Wai Long, Taipa, Macau SAR 999078, China; 1909853vii30001@student.must.edu.mo
[2] Hitachi Building Technology (Guangzhou) Co., Ltd., No. 2 Nanxiang 3rd Road, Guangzhou 510613, China; yujiaxin@hitachi-helc.com
[3] Institute of Systems Engineering, Macau University of Science and Technology, Avenida Wai Long, Taipa, Macau SAR 999078, China; zwli@must.edu.mo
* Correspondence: liyin@must.edu.mo

Abstract: This work addresses the problem of quantifying opacity for discrete event systems. We consider a passive intruder who knows the overall structure of a system but has limited observational capabilities and tries to infer the secret of this system based on the captured information flow. Researchers have developed various approaches to quantify opacity to compensate for the lack of precision of qualitative opacity in describing the degree of security of a system. Most existing works on quantifying opacity study specified probabilistic problems in the framework of probabilistic systems, where the behaviors or states of a system are classified as secret or non-secret. In this work, we quantify opacity by a state-worth function, which associates each state of a system with the worth it carries. To this end, we present a novel category of opacity, called worthy opacity, characterizing whether the worth of information exposed to the outside world during the system's evolution is below a threshold. We first provide an online approach for verifying worthy opacity using the notion of a run matrix proposed in this research. Then, we investigate a class of systems satisfying the so-called 1-cycle returned property and present a worthy opacity verification algorithm for this class. Finally, an example in the context of smart buildings is provided.

Keywords: discrete event systems; opacity; automata; smart building

MSC: 93C65

Citation: Zhou, S.; Yu, J.; Yin, L.; Li, Z. Security Quantification for Discrete Event Systems Based on the Worth of States. *Mathematics* **2023**, *11*, 3629. https://doi.org/10.3390/math11173629

Academic Editor: Cristina I. Muresan

Received: 30 June 2023
Revised: 11 August 2023
Accepted: 21 August 2023
Published: 22 August 2023

1. Introduction

With global urbanization, cities are growing in size and population, and the proportion of people living in urban areas is expected to grow to 68% by 2050 [1], creating inevitable challenges to urban residents due to limited resources and services. The concept of smart cities has been proposed to efficiently deploy public resources, improve social governance, and promote sustainable urban development. Currently, smart cities are being intensively established around the world; for example, there are already more than 500 in China [2].

Buildings provide the physical space required for people to carry out various social, economic, and cultural activities, and are one of the areas where the Internet of Things (IoT) would make significant impacts [3]. Smart buildings improve the energy efficiency, operational efficiency, security, and comfort of the living or working environment by integrating advanced technologies and systems [4,5]. For example, people use the model predictive control approach for temperature control in smart buildings, reducing operating costs and improving thermal comfort [6,7]. However, IoT devices in smart buildings generate a large amount of data, which can pose many potential threats. Deciding how to analyze the security of smart buildings is becoming an increasingly important issue and has attracted a lot of attention over the past few years [8–10].

In this article, we study an information flow security property of great interest: opacity, within the discrete event system (DES) framework, where DES can describe the state transition relationships between events in an IoT, thereby assisting us in understanding and analyzing IoT behavior. Opacity was originally introduced in 2004 for the analysis of cryptographic protocols [11]. Later, it is taken as a general information flow security framework for a DES interacting with a passive intruder. Many kinds of information flow security properties can be formulated as opacity, such as non-interference [12], anonymity [13], etc. Roughly speaking, opacity is a type of confidentiality property that characterizes whether certain secret information from a system can be deduced by outside observers that may be malicious.

In the context of DESs, opacity is classified into language-based and state-based opacity, depending on the defined secret [14]. Language-based opacity has been studied with models such as automata [13] and Petri nets [15], where the secret is defined as a language. In general, this type of opacity can be further classified as strong and weak opacity, as opposed to state-based opacity, which has a wide range of classifications depending on the state of interest. For example, current-state opacity [16,17] has been proposed to take the system's current state as a secret and determine whether the system reveals this secret during its evolution. Similarly, initial-state opacity [18] and initial-and-final-state opacity [19] are suggested when the secret is defined as a set of initial states or a set of secret state pairs. The notions of K-step opacity [20] and infinite-step opacity [21] are developed when the delayed information of a system is concerned, and pre-opacity [22] is formulated when the intention of a system needs to be kept secret.

All of the aforementioned notions of opacity describe a system in a binary way, where the system is either opaque or non-opaque. It has been noted that qualitative opacity is not accurate enough in assessing the information obtained by passive attackers [23]. For instance, a system that violates qualitative opacity with a very low probability and one that violates qualitative opacity with a very high probability are considered insecure. However, their degree of security is different [24]. The authors of [25] introduce game theory's ideas into probabilistic automata and propose a model of probabilistic resource automata based on which the current-state opacity is quantified.

The above-mentioned studies on quantification of opacity are basically based on probabilistic systems. However, a probabilistic model is much more difficult to be abstracted than a non-probabilistic version. The study in [26] is the first work that introduces the quantification of opacity into non-probabilistic DES. Considering that the essence of opacity is to explore the observational equivalent strings, the authors developed a method to calculate the opacity degree of a system by computing the dispersion among non-secret runs and secret equivalents.

In this work, we investigate the quantification of opacity from a new perspective by describing the worth of information that each system state carries in terms of a *state-worth function*. Then, we introduce a new type of opacity, called *worthy opacity*, to describe whether the worth of information exposed to the outside world in the system's evolution meets the security requirements. We consider an intruder who knows the whole structure of a system but has limited observation capabilities. Unlike the general notions of state-based opacity, we do not explicitly treat a secret as a specific sub-set of states since the state-worth function describes the importance of each system state. We show that current-state opacity is strictly weaker than worthy opacity. The proposed notion of worthy opacity is closely related to the notion of opacity degree in [26]. The contributions of this article can be summarized as follows:

- A novel notion of worthy opacity is proposed to quantitatively characterize the worth of a system available to an intruder;
- An online algorithm is provided to verify worthy opacity;
- A system property called 1-cycle returned is defined, and an offline verification algorithm for the system's worthy opacity satisfying this property is presented;
- It is shown that worthy opacity provides a more granular partitioning of the system than current-state opacity.

The rest of this article is organized as follows. Section 2 recalls the necessary preliminaries. The notion of worthy opacity is proposed in Section 3. In Section 4, two effective algorithms for the verification of worthy opacity are reported. Finally, Section 5 concludes this research.

2. Preliminaries

Symbols $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, $\mathbb{N}$, and $\mathbb{Z}^+$ are specified to denote the sets of real numbers, non-negative real numbers, non-negative integers, and positive integers, respectively. The ceiling function of a real number $x \in \mathbb{R}$, denoted as $\lceil x \rceil$, is defined as the smallest integer that is not smaller than x. For a vector $\boldsymbol{v}$, $[\boldsymbol{v}]_i$ is used to denote the i-th entry of $\boldsymbol{v}$. Given a matrix $\boldsymbol{M}$, its (i,j)-th entry is denoted as $[\boldsymbol{M}]_{i,j}$. The transpose of a vector $\boldsymbol{v}$ is denoted by $\boldsymbol{v}^T$. Given a k-dimensional vector $\boldsymbol{v}$, its *1-norm* is defined as the sum of the absolute values of each entry, i.e., $\|\boldsymbol{v}\| = \sum_{i=1}^{k} |[\boldsymbol{v}]_i|$. The *L1-normalization* of a vector $\boldsymbol{v}$, denoted by $\mathtt{norm}(\boldsymbol{v})$ in this article, is defined as $\mathtt{norm}(\boldsymbol{v}) = \boldsymbol{v}/\|\boldsymbol{v}\|$. Given an ordered pair $c = (a,b)$, we use $c(1)$ and $c(2)$ to denote its first and second components, namely, $c(1) = a$ and $c(2) = b$.

2.1. System Model

We define an *alphabet* as any non-empty finite set of events, denoted by E. A *string* over the alphabet E is a sequence of events taken out of E [27]. The length of a string s, written as $|s|$ (if X is a set, the notation $|X|$ denotes the cardinality of X. The distinction is usually clear from context), is the number of events contained in it, counting multiple occurrences of the same event. The string without any events is called the empty string and is denoted by ε with $|\varepsilon| = 0$. Given two strings s_1 and s_2, the *concatenation* of s_1 and s_2 is the sequence of events in s_1 followed by the sequence of events in s_2, denoted as $s_1 \cdot s_2$ or $s_1 s_2$. We use the superscript notation s^k to indicate that the string s is concatenated with itself k times.

Given an alphabet E, we denote by E^k the set of all strings of length k, particularly, $E^0 = \{\varepsilon\}$. The set consisting of all finite-length strings defined over E is denoted by E^*, i.e., $E^* = E^0 \cup E \cup E^2 \cup \cdots$. Given an alphabet E, a *language* is a sub-set of E^*. Given two languages L_1 and L_2, the *concatenation* of L_1 and L_2 is $L_1 L_2 = \{s_1 s_2 \mid s_1 \in L_1, s_2 \in L_2\}$. In this article, we model a DES as a *non-deterministic finite automaton* , formally defined as follows.

Definition 1 (Non-deterministic Finite Automaton)**.** *A non-deterministic finite automaton (NFA) is a four-tuple $G = (Q, E, f, Q_0)$, where*

- *$Q = \{q_1, q_2, \cdots, q_N\}$ is the finite set of states;*
- *$E = \{e_1, e_2, \cdots, e_M\}$ is the finite set of events associated with G;*
- *$f\colon Q \times E \to 2^Q$ (where 2^Q is the power set of Q) is the transition function, and $q' \in f(q,e)$ means that there is a transition labeled by e from state q to state q';*
- *$Q_0 \subseteq Q$ is the set of initial states.*

If Q_0 in G is a singleton and the transition function of G is a partially defined function $Q \times E \to Q$, then G is called a deterministic finite automaton (DFA). The transition function f can be extended recursively from the domain $Q \times E$ to the domain $Q \times E^*$: $f(q,\varepsilon) = \{q\}$ and $f(q,se) = \bigcup_{q' \in f(q,s)} f(q',e)$ for all states $q \in Q$, where $s \in E^*$ and $e \in E$. The language generated by G from state $q \in Q$ is $\mathcal{L}(G,q) = \{s \in E^* \mid f(q,s)!\}$, where $f(q,s)!$ indicates that $f(q,s)$ is defined, i.e., $f(q,s) \neq \emptyset$. The language generated by G from a set of states $Q' \subseteq Q$ is $\mathcal{L}(G,Q') = \bigcup_{q \in Q'} \mathcal{L}(G,q)$. Naturally, the language generated by G is $\mathcal{L}(G) = \mathcal{L}(G,Q_0)$.

Formally an NFA G can be equivalently represented by a directed graph, with a set of nodes Q denoting states and a set of edges $\{q \xrightarrow{e} q' \mid q' \in f(q,e)\}$ denoting transitions. A *run* in G starting from $q_{(0)} \in Q$ is a finite sequence of transitions $r\colon q_{(0)} \xrightarrow{e_{(1)}} q_{(1)} \xrightarrow{e_{(2)}}$

$\cdots q_{(k-1)} \xrightarrow{e_{(k)}} q_{(k)}$ (here, we use the numbers in parentheses to indicate the order in which states or events appear in the run, that is to say, $q_{(i)}$ is not necessarily equal to q_i and $e_{(i)}$ is not necessarily equal to e_i), and the events extracted from it in order form the string associated with it, denoted by $\phi(r) = e_{(1)}e_{(2)}\cdots e_{(k)}$ (we also say that r is a run on $\phi(r)$ and use $last(r) = q_{(k)}$ to denote the ending state). We define the *length* of a run r as the length of the string $\phi(r)$ associated with it, i.e., $|r| = |\phi(r)|$. Given a run from state q to q' on a string s, we denote it as $q \xrightarrow{s} q'$; if a specific string associated with the run is not of interest, we denote it as $q \to q'$. The set of all runs starting from state $q \in Q$ in G is denoted by $\Gamma(G, q)$; the set of all runs generated by G is $\Gamma(G) = \cup_{q\in Q_0}\Gamma(G, q)$. A run of length greater than zero that begins and ends at the same state is called a *cycle*.

2.2. Intruder Model and Opacity

In the general setting of studying opacity in DESs, it is assumed that the intruder is fully aware of the system's structure but only partially observes the system's behavior [14]. We follow this setting in our work and formalize it as the partial observability of events, where the set of events E is divided into an observable event set E_o and an unobservable event set E_{uo}, i.e., $E = E_o \dot{\cup} E_{uo}$. Given a string $s \in E^*$ generated by G, the intruder's observation is the output of the *natural projection function* $P\colon E^* \to E_o^*$, which is defined recursively as:

$$P(\varepsilon) = \varepsilon; P(e) = \begin{cases} e, & \text{if } e \in E_o, \\ \varepsilon, & \text{otherwise;} \end{cases}$$

$$P(se) = P(s) \cdot P(e) \text{ for } s \in E^*, e \in E.$$

The *unobservable reach* of a state $q \in Q$ in G, denoted by $UR(q)$, is $UR(q) = \{q' \in Q \mid \exists t \in E_{uo}^*\colon f(q, t)!, q' \in f(q, t)\}$. This definition can be extended to a set of states $Q' \subseteq Q$ by $UR(Q') = \bigcup_{q\in Q'} UR(q)$.

This work is an extension of state-based opacity, where the secret of a system is a sub-set of states $S \subseteq Q$. When a system generates a string $s \in E^*$, the intruder observes $P(s)$ and infers whether the system is in a secret state based on this observation and the system's structure.

Definition 2 (Current-State Opacity [14]). *Given a system $G = (Q, E, f, Q_0)$, a secret $S \subseteq Q$, and a set of observable events $E_o \subseteq E$, G is current-state opaque with respect to S and E_o, written as (S, E_o)-CSO, if*

$$\forall q \in Q_0, \forall s \in \mathcal{L}(G, q)[f(q, s) \subseteq S] \Rightarrow \exists q' \in Q_0, \exists s' \in \mathcal{L}(G, q')[P(s) = P(s'), f(q', s') \nsubseteq S].$$

In plain words, a system being current-state opaque means that an intruder cannot infer whether the current state belongs to the secret, regardless of the sequence of events occurring in the system. Given an observation $\omega \in P(\mathcal{L}(G))$, the *current-state estimate* associated with ω is $\mathcal{C}(\omega) = \{q \in Q \mid \exists q_0 \in Q_0\colon s \in \mathcal{L}(G, q_0), q \in f(q_0, s), P(s) = \omega\}$, and the set of *consistent strings* is $\mathcal{S}(\omega) = \{s \in \mathcal{L}(G) \mid P(s) = \omega\}$.

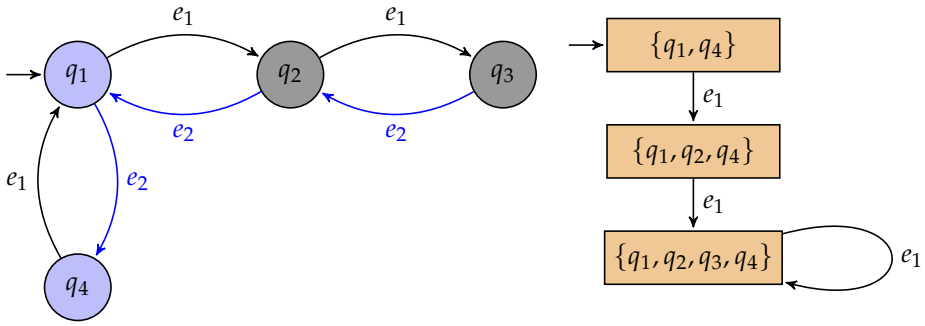
Lemma 1 ([28]). *Given a system $G = (Q, E, f, Q_0)$, a secret $S \subseteq Q$, and a set of observable events $E_o \subseteq E$, G is (S, E_o)-CSO if and only if for any observation $\omega \in P(\mathcal{L}(G))$, $\mathcal{C}(\omega) \nsubseteq S$.*

By Lemma 1, to verify current-state opacity, we can construct the *observer* G_{obs} of system G and check whether there exists a state in observer G_{obs} that is a sub-set of S [16].

Definition 3. *Given a system $G = (Q, E, f, Q_0)$ and a set of observable events $E_o \subseteq E$, its observer is a DFA $G_{obs} = (X, E_{obs}, f_{obs}, x_0)$, where*

- $X \subseteq 2^Q$, *with each state* $x \subseteq Q$ *being a state estimate generated by an intruder based on the evolution of* G;
- $E_{obs} = E_o$ *is the set of events that can be observed by an intruder;*
- $f_{obs}(x, e) = \bigcup_{q \in x} UR(f(q, e))$;
- $x_0 = UR(Q_0)$ *is the initial state estimate.*

Example 1. *Consider the system* G_1 *in Figure 1a, where* $S = \{q_2, q_3\}$ *and* $E_o = \{e_1\}$. *Clearly, by constructing the observer of* G_1 *as shown in Figure 1b, we can find that* G *is current-state opaque, since no state of* $G_{1,obs}$ *is a sub-set of* S.



(**a**) (**b**)

Figure 1. (**a**) A system G_1 and (**b**) the observer $G_{1,obs}$ with respect to G_1.

2.3. Some Counting Principles

As an essential part of combinatorics, counting objects with specific properties is indispensable for the study of DESs. Many different types of problems require us to count items. For instance, some notions of *diagnosability* are defined in terms of fault counting problems [29]. Counting is also required to determine the complexity of algorithms [30]. In addition, counting techniques are widely used in calculating the finite probability [31]. Our work is also inseparable from counting. This subsection recalls some critical counting principles [32].

Lemma 2 (Addition Principle)**.** *Suppose that a set* S *can be partitioned into pairwise disjoint parts* $S_1, S_2, \cdots, S_m$. *The number of elements in* S *is the sum of the number of elements in each of its part, i.e.,* $|S| = \sum_{i=1}^{m} |S_i|$.

We divide the problem into mutually exclusive cases for applying the addition principle. An alternative formulation of Lemma 2 is as follows: if there are m ways to complete a task, where the i-th way has k_i ($i \in \{1, 2, \cdots, m\}$) choices, then there are $\sum_{i=1}^{m} k_i$ choices for completing that task.

Example 2. *Consider the system* G_1 *in Example 1. Suppose we want to find the number of cycles of length two in* G_1. *We divide these cycles by listing their start and end states. The number of* 2*-length cycles starting and ending in states* q_1, q_2, q_3, *and* q_4 *are* 2, 2, 1, *and* 1, *respectively. Therefore, the number of cycles of length two in* G_1 *is* $2 + 2 + 1 + 1 = 6$.

Lemma 3 (Multiplication Principle)**.** *Let* S *be a set of* m*-tuples* $(S_1, S_2, \cdots, S_m)$, *where the* i*-th component* S_i *comes from a set of size* a_i. *The size of* S *is* $\prod_{i=1}^{m} a_i$.

The multiplication principle is a corollary of Lemma 2. A useful formulation of Lemma 3 is as follows: suppose that a task can be decomposed into m consecutive steps. If step i can be performed in a_i ways, and for each of these, step $i + 1$ can be performed in a_{i+1} ($i \in \{1, 2, \cdots, m - 1\}$) ways, then the task itself can be accomplished in $\prod_{i=1}^{m} a_i$ ways.

Example 3. *Consider the system G_1 in Example 1. The number of 2-length runs starting from state q_1, passing through state q_2, and ending at state q_3 is 1, as the number of 1-length runs from state q_1 to q_2 and state q_2 to q_3 is 1, respectively.*

Lemma 4 (Generalized Pigeonhole Principle [33])**.** *If p objects are placed into q boxes, then at least one box contains a minimum of $\lceil p/q \rceil$ objects.*

3. Notions of Worthy Opacity

In this section, we first provide the definition of worthy opacity for DESs, and then compare it with the widely studied current-state opacity. Existing notions of state-based opacity, both qualitative and quantitative, divide the set of states into secret or non-secret. However, various degrees of confidentiality exist for different states that are part of the same secret. For example, a spy trying to hide his/her tracks would wish neither his/her hiding place be exposed nor the convenience store he/she regularly visits be discovered. Nevertheless, it is clear that the secrecy of the hiding place is more valuable. To describe the degree of confidentiality of individual states, we introduce the notion of *state-worth function*.

Definition 4 (State-worth Function)**.** *Given a system $G = (Q, E, f, Q_0)$, a state-worth function is a mapping that assigns a non-negative number to a state, defined as $\Delta\colon Q \to \mathbb{R}_{\geq 0}$.*

Note that instead of describing the degree of confidentiality of each state in the secret (a sub-set of the state set), we define the state-worth function as assigning a worth to each state of the system. This is a more general approach than splitting the states into two categories. When linked to the conventional notion of secret, we can use this function to divide secrets: each state possesses a worth, and states carrying worth above a certain threshold are considered constituent elements of a secret.

Example 4. *Suppose that we have a bank account with a deposit limit of \$300 and a balance of \$100. The amount we deposit into our account or spend at a point of sale (POS) is \$100 each time. In addition, a private detective is lying in wait at the bank door and wants to know our financial situation, and he can only find out that we deposit but not that we use POS spending.*

Then, the above scenario can be modeled with the system of Example 1; event e_1 represents the deposit of \$100 while event e_2 represents the POS spending of \$100. Each state represents the balance of the bank account, and naturally, we can take the balance represented by each state as the value of the state-worth function, i.e., $\Delta(q_1) = 100$, $\Delta(q_2) = 200$, $\Delta(q_3) = 300$, and $\Delta(q_4) = 0$. If we treat a bank balance over \$100 as a secret, then it is $S = \{q \mid \Delta(q) > 100\} = \{q_2, q_3\}$, as shown in Example 1.

In addition, the existing notions of opacity are not sufficiently refined for depicting the evolution of a system. In fact, the most fundamental element of a system corresponding to an observation is the set of runs rather than the set of strings, as shown in Figure 2. Specifically, given a run $r \in \Gamma(G)$ of a system G, we can obtain the string $\phi(r) \in \mathcal{L}(G)$ by extracting the sequence of events occurring in it, also commonly considered as the *logical behavior* of a system, and then an intruder obtains the corresponding observation $\omega \in P(\mathcal{L}(G))$ based on the observability of the events.

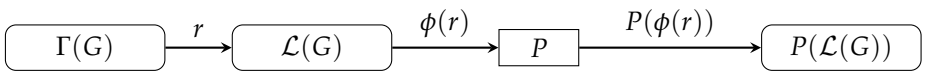


Figure 2. The generation of observations.

Definition 5. *Given a system $G = (Q, E, f, Q_0)$ with a set of observable events $E_o \subseteq E$ and an observation $\omega \in P(\mathcal{L}(G))$, the set of consistent runs and the set of consistent runs ending in state $q \in Q$ are defined as $\Phi(\omega) = \{r \in \Gamma(G) \mid \exists s \in \mathcal{S}(\omega)\colon \phi(r) = s\}$ and $\Phi_q(\omega) = \{r \in \Phi(\omega) \mid last(r) = q\}$, respectively.*

Intuitively, given an observation ω, *the set of consistent runs* $\Phi(\omega)$ is the set of all runs for which the system can generate that observation, and accordingly, the set $\Phi_q(\omega)$ is the set of all runs that cause the system to produce that observation and reach state q.

Definition 6 (Worthy Opacity)**.** *Given a system $G = (Q, E, f, Q_0)$ with state-worth function Δ, a set of observable events $E_o \subseteq E$, and a non-negative value $K \in \mathbb{R}_{\geq 0}$, an observation $\omega \in P(\mathcal{L}(G))$ is worthy opaque with respect to E_o, Δ, and K (denoted by (E_o, Δ, K)-WO) if*

$$\sum_{q \in Q} \alpha_\omega(q) \cdot \Delta(q) \leq K, \tag{1}$$

where $\alpha_\omega(q) = |\Phi_q(\omega)|/|\Phi(\omega)|$. System G is said to be worthy opaque with respect to E_o, Δ, and K ((E_o, Δ, K)-WO) if all observations $\omega \in P(\mathcal{L}(G))$ are (E_o, Δ, K)-WO.

Note that $\alpha_\omega(q)$ in Definition 6 can be viewed as the intruder's probability estimate of the current state of the system as q after observing ω, i.e., the probability of system G in state $q \in \mathcal{C}(\omega)$ when observation ω is generated is implicitly defined as the number of runs in which the system generates observation ω and ends up in state q divided by the number of all runs that can generate observation ω. Then, the left-hand side of (1) shows the worth the system is expected to expose for generating the observation ω. That is to say, a system is worthy opaque if the worth it exposes to the outside world during its evolution does not exceed a certain threshold.

Example 5. *Consider the situation in Example 4, which is modeled as the system G_1 of Example 1. Suppose that the private detective sees that we deposit \$100 in the bank, i.e., the system G_1 produces the observation e_1. We have $\Phi(e_1) = \{q_1 \xrightarrow{e_1} q_2 \xrightarrow{e_2} q_1, q_1 \xrightarrow{e_2} q_4 \xrightarrow{e_1} q_1, q_1 \xrightarrow{e_1} q_2, q_1 \xrightarrow{e_2} q_4 \xrightarrow{e_1} q_1 \xrightarrow{e_2} q_4, q_1 \xrightarrow{e_1} q_2 \xrightarrow{e_2} q_1 \xrightarrow{e_2} q_4\}$. Thus, we have $|\Phi_{q_1}(e_1)| = 2$, $|\Phi_{q_2}(e_1)| = 1$, $|\Phi_{q_4}(e_1)| = 2$, and $|\Phi(e_1)| = 5$, leading to $\alpha_{e_1}(q_1) = 2/5$, $\alpha_{e_1}(q_2) = 1/5$, and $\alpha_{e_1}(q_4) = 2/5$. By (1), we have $\sum_{q \in Q} \alpha_\omega(q) \cdot \Delta(q) = 2/5 \cdot 100 + 1/5 \cdot 200 + 2/5 \cdot 0 = 80$, implying that observation e_1 is $(E_o, \Delta, 80)$-WO. In plain words, the worth of information revealed to the outside world by observation e_1 is 80, which is a reasonable inference that the private detective can make.*

Notice that the probabilities implied in Definition 6 are based on a uniform distribution over the set of consistent runs, which implies that the set $\Phi(\omega)$ must be finite, since there is no uniform distribution over infinite countable sets; otherwise the additivity of the probability axioms [31] is violated. Therefore, we have the following assumption on system G.

Assumption 1. *There are no unobservable cycles in the system G, where an unobservable cycle c_u is a cycle such that $P(\phi(c_u)) = \varepsilon$.*

The above assumption ensures that, for each observation, the set of consistent runs is finite (as shown in Proposition 1), and this assumption is also a general one when the system is modeled as a Petri net and the problem is analyzed using the notion of a *basis reachability graph* [28].

Proposition 1. *Given a system $G = (Q, E, f, Q_0)$ with a set of observable events $E_o \subseteq E$, the set $\Phi(\omega)$ is finite for any observation $\omega \in P(\mathcal{L}(G))$ if there are no unobservable cycles in G.*

Proof. We first show that system G can generate no more than $N^{k+1}M^k$ runs of length k, where N is the number of states and M is the number of events in G. Note that we can consider a k-length run as a cross-arrangement of $k+1$ states and k events, provided that the transition function f is satisfied and $q_{(0)} \in Q_0$. The number of k-length runs is maximized by setting $Q_0 = Q$ and $f(q, e) = Q$ (for any $q \in Q$ and $e \in E$). In this way, each state and each event in a run can be arbitrarily selected from N states and M events, respectively. In other words, the number of k-length runs cannot exceed $N^{k+1}M^k$.

Then, we prove the proposition by contrapositive. Suppose that there exists an observation $\omega = o_{(1)}o_{(2)}\cdots o_{(l)}$ such that the set $\Phi(\omega)$ is infinite. The length of runs in set $\Phi(\omega)$ can be greater than any given positive integer L. Otherwise, the number of runs in $\Phi(\omega)$ cannot exceed $\sum_{i=1}^{L} N^{i+1}M^i$, which implies that $\Phi(\omega)$ is finite.

Now, let $L = N \cdot (l+2) - 1$, we can find a run

$$r_L = q_{(0)} \xrightarrow{e_{(1)}} q_{(1)} \xrightarrow{e_{(2)}} \cdots q_{(N\cdot(l+2)-1)} \xrightarrow{e_{(N\cdot(l+2))}} q_{(N\cdot(l+2))},$$

whose length is greater than L. By Lemma 4, we can obtain that there is a state q_d of G that is duplicated at least $l+2$ times in r_L. This implies that there are at least $l+1$ cycles in r_L that begin and end with state q_d. Since the length of ω is $l < l+1$, we conclude that at least one of these $l+1$ cycles is unobservable, which completes the proof. □

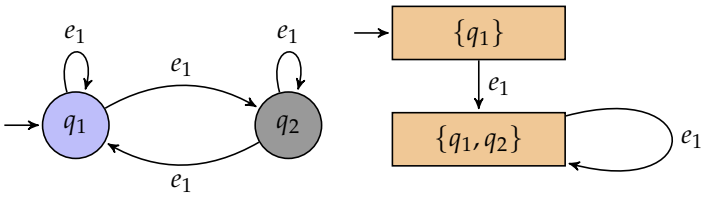
Recall that in the research of state-based opacity, the secret is defined as a sub-set of system states. In general, secret states are more valuable than non-secret states. With this consideration, we can relate current-state opacity to the proposed worthy opacity.

Proposition 2. *Given a system $G = (Q, E, f, Q_0)$ with state-worth function Δ, a secret $S = \{q \mid \Delta(q) > K\}$ ($K \in \mathbb{R}_{\geq 0}$), and a set of observable events $E_o \subseteq E$, G is (S, E_o)-CSO, if G is (E_o, Δ, K)-WO.*

Proof. By contrapositive, suppose that G is not (S, E_o)-CSO. By Lemma 1, there exists an observation ω such that $\mathcal{C}(\omega) \subseteq S$. That is, for any $q \in \mathcal{C}(\omega)$, $\Delta(q) > K$ holds, which leads to the fact that G is not (E_o, Δ, K)-WO. □

Note that the converse of the above proposition does not hold, as illustrated by the following example.

Example 6. *Consider the system G_2 in Figure 3a, where $E_o = E = \{e_1\}$. Let $\Delta(q_1) = 50$, $\Delta(q_2) = 200$, and $S = \{q \mid \Delta(q) > 100\} = \{q_2\}$. It is not difficult to verify that system G_2 is (S_2, E_o)-CSO. By analysis, we see that the evolution of G_2 satisfies Table 1. From Definition 6, we find that the system is $(E_o, \Delta, 125)$-WO, not $(E_o, \Delta, 100)$-WO.*



(**a**) (**b**)

Figure 3. (**a**) A system G_2 and (**b**) the observer $G_{2,obs}$ with respect to G_2.

Table 1. Number of runs in G_2.

Observation ω ($i \in \mathbb{Z}^+$)	$\lvert\Phi_{q_1}(\omega)\rvert$	$\lvert\Phi_{q_2}(\omega)\rvert$
ε	1	0
${e_1}^i$	2^{i-1}	2^{i-1}

4. Verifying Worthy Opacity

Intuitively, to verify the worthy opacity of a given system G, we need to check whether (1) holds for all $\omega \in P(\mathcal{L}(G))$, which means that the value of $\alpha_\omega(q)$ needs to be computed for all $q \in Q$. In general, this requires an exhaustive enumeration of all possible strings that can be generated by G, which may require infinite memory and thus render the problem unsolvable. In this section, we first provide an online verification procedure, and then propose an algorithm to identify a particular class of systems and verify their worthy opacity.

4.1. Online Verification of an Observation

Given an observation ω, we develop a notion of *run matrix* to compute the cardinality of a set $\Phi_q(\omega)$ (for any $q \in Q$) based on the *transition matrix* of automata.

Definition 7 (Transition Matrix [34]). *Given an event $e \in E$ of a system $G = (Q, E, f, Q_0)$, the transition matrix $\boldsymbol{T}_e$ is an $N \times N$ matrix, where the typical entry $[\boldsymbol{T}_e]_{i,j}$ is equal to one if $q_i \in f(q_j, e)$ and to zero otherwise.*

Example 7. *Consider the system in Example 1. The transition matrices associated with events e_1 and e_2 are, respectively:*

$$\boldsymbol{T}_{e_1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \boldsymbol{T}_{e_2} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

As runs are composed of transitions, we can utilize the transition matrix to calculate the number of runs between two states on a given string $s \in E^*$. Based on the idea of a transition matrix, we propose the notion of a *run matrix*.

Definition 8 (Run Matrix). *Given a system $G = (Q, E, f, Q_0)$ and a string $s \in E^*$, the run matrix $\boldsymbol{R}_s$ is an $N \times N$ matrix whose typical entry $[\boldsymbol{R}_s]_{i,j}$ is equal to the number of runs from q_j to q_i on string s.*

Recall that the values of transition function f of system G are sub-sets of states rather than multi-sets [35], which means that given a state q and an event e, the number of runs from state q to q' on the one-length string e is one if $q' \in f(q, e)$ and zero otherwise. This coincides with the definition of the transition matrix; hence, $\boldsymbol{R}_e = \boldsymbol{T}_e$ for all $e \in E$. For the empty string ε, to fit the extended definition of transition function, we have $\boldsymbol{R}_\varepsilon = \boldsymbol{I}$, where $\boldsymbol{I}$ is an identity matrix of size N.

Remark 1. *For a string $s \notin \mathcal{L}(G, Q)$, its corresponding run matrix is a null matrix, indicating no run on s in G from whatever state it starts in.*

Even though we specify the run matrices associated with strings of length zero and one for a given system G, it is still a problem to compute the run matrix associated with a string s of length greater than one, which is a more general case. Before we introduce the computation, we prove the following two properties of run matrices.

Proposition 3. *Given two strings s_1 and s_2 that have corresponding run matrices $\boldsymbol{R}_{s_1}$ and $\boldsymbol{R}_{s_2}$, respectively,*

(1) In the case that s_1 and s_2 are not identical, the number of runs on string s_1 or s_2 can be described by matrix $\boldsymbol{R}_{s_1+s_2} = \boldsymbol{R}_{s_1} + \boldsymbol{R}_{s_2}$, i.e., the number of runs from state q_j to q_i on string s_1 or s_2 is $[\boldsymbol{R}_{s_1+s_2}]_{i,j}$;

(2) The number of runs on string $s_1 s_2$ can be described by matrix $\boldsymbol{R}_{s_1 s_2} = \boldsymbol{R}_{s_2} \cdot \boldsymbol{R}_{s_1}$, i.e., the number of runs from state q_j to q_i on string $s_1 s_2$ is $[\boldsymbol{R}_{s_1 s_2}]_{i,j}$.

Proof. (1) As $s_1 \neq s_2$, a system cannot generate strings s_1 and s_2 simultaneously, whereas the way the system generates a string is the number of runs it can generate on that string. By Lemma 2, we have that the number of runs on string s_1 or s_2 is equal to the sum of the number of runs on s_1 and the number of runs on s_2, which is $[\boldsymbol{R}_{s_1}]_{i,j} + [\boldsymbol{R}_{s_2}]_{i,j} = [\boldsymbol{R}_{s_1+s_2}]_{i,j}$.

(2) Clearly, a system generates a string $s_1 s_2$ in the order that first yields s_1 and then generates s_2. By Lemma 3, we have that the number of runs from state q_j to q_i on $s_1 s_2$ is

equal to the product of the number of runs from state q_j to any state $q_k \in Q$ and the number of runs from that state q_k to state q_i, which is $\sum_{1\leq k\leq N}[\boldsymbol{R}_{s_1}]_{k,j} \cdot [\boldsymbol{R}_{s_2}]_{i,k} = [\boldsymbol{R}_{s_1 s_2}]_{i,j}$. □

Remark 2. *Given any string $s \in E^*$, we have $\boldsymbol{R}_s = \boldsymbol{R}_s \cdot \boldsymbol{R}_\varepsilon = \boldsymbol{R}_\varepsilon \cdot \boldsymbol{R}_s$, which also meets the definition of the concatenation of strings.*

Based on the second part of the above proposition, we have the following corollary, which can be used to calculate the run matrix associated with a string $s \in E^*$ of length greater than zero.

Corollary 1. *The run matrix associated with a string $s = e_{(1)}e_{(2)}\cdots e_{(k)}$ $(k > 0)$ is*

$$\boldsymbol{R}_s = \boldsymbol{T}_{e_{(k)}}\boldsymbol{T}_{e_{(k-1)}}\cdots\boldsymbol{T}_{e_{(2)}}\boldsymbol{T}_{e_{(1)}}. \tag{2}$$

We can also extend the notion of run matrices to languages, i.e., given a language L, the run matrix $\boldsymbol{R}_L$ associated with it is an $N \times N$ matrix whose typical entry $[\boldsymbol{R}_L]_{i,j}$ is equal to the total number of runs on all strings in L from q_j to q_i. It is not difficult to conclude that $\boldsymbol{R}_L = \sum_{s\in L}\boldsymbol{R}_s$. Moreover, the extended run matrix has the following properties.

Proposition 4. *Given two finite languages L_1 and L_2 that have corresponding run matrices $\boldsymbol{R}_{L_1}$ and $\boldsymbol{R}_{L_2}$, respectively,*

(1) In the case that L_1 and L_2 are disjoint, $\boldsymbol{R}_{L_1\cup L_2} = \boldsymbol{R}_{L_1} + \boldsymbol{R}_{L_2}$.
(2) $\boldsymbol{R}_{L_1\cdot L_2} = \boldsymbol{R}_{L_2} \cdot \boldsymbol{R}_{L_1}$.

Proof. (1) As L_1 and L_2 are disjoint, the string in $L_1 \cup L_2$ either belongs to L_1 or to L_2. Therefore, we have $\boldsymbol{R}_{L_1\cup L_2} = \sum_{s\in L_1}\boldsymbol{R}_s + \sum_{s\in L_2}\boldsymbol{R}_s = \boldsymbol{R}_{L_1} + \boldsymbol{R}_{L_2}$.

(2) By $L_1L_2 = \{s_1s_2 \mid s_1 \in L_1, s_2 \in L_2\}$, and the definition of extended run matrices, we have $\boldsymbol{R}_{L_1\cdot L_2} = \sum_{s_1\in L_1, s_2\in L_2}\boldsymbol{R}_{s_2}\cdot\boldsymbol{R}_{s_1} = (\sum_{s_2\in L_2}\boldsymbol{R}_{s_2})\cdot(\sum_{s_1\in L_1}\boldsymbol{R}_{s_1}) = \boldsymbol{R}_{L_2}\cdot\boldsymbol{R}_{L_1}$, where the first equal sign is due to the second part of Proposition 3. □

Remark 3. *For any finite language $L \subseteq E^*$, we have $\boldsymbol{R}_L = \boldsymbol{R}_{L\cap\mathcal{L}(G,Q)}$, as for any $s \in L \setminus \mathcal{L}(G,Q)$, we have $\boldsymbol{R}_s = \boldsymbol{O}$ by Remark 1, where $\boldsymbol{O}$ is a null matrix.*

Proposition 5. *Given a system $G = (Q, E, f, Q_0)$, define an N-dimensional column vector $\boldsymbol{\pi}$ such that $[\boldsymbol{\pi}]_i = 1$ if $q_i \in Q_0$ and $[\boldsymbol{\pi}]_i = 0$ otherwise. Then, $[\boldsymbol{R}_L \cdot \boldsymbol{\pi}]_i$ is the total number of runs generated by system G on strings in language L, where the ending state of these runs is q_i.*

Proof. It can be directly obtained from the definition of the multiplication of matrices and the meaning of run matrices. □

Based on the above discussions, we can use matrix operations to calculate the number of runs between different states on observations. It is not difficult to find that given an observation $\omega = o_{(1)}o_{(2)}\cdots o_{(l)}$, we have $\mathcal{S}(\omega) = E_{uo}^*\{o_{(1)}\}E_{uo}^*\{o_{(2)}\}E_{uo}^*\cdots E_{uo}^*\{o_{(l)}\}E_{uo}^* \cap \mathcal{L}(G)$. Based on Definition 5, we know that the cardinality of set $\Phi_q(\omega)$ is the total number of runs generated by the system on all strings in language $\mathcal{S}(\omega)$ that can reach state q, i.e., $|\Phi_{q_i}(\omega)| = [\boldsymbol{R}_{\mathcal{S}(\omega)}\cdot\boldsymbol{\pi}]_i$ $(1 \leq i \leq N)$. With Remark 3, once the run matrix corresponding to the language $E_{uo}^*\{o_{(1)}\}E_{uo}^*\{o_{(2)}\}E_{uo}^*\cdots E_{uo}^*\{o_{(l)}\}E_{uo}^*$ is obtained, the cardinality of set $\Phi_q(\omega)$ for any $q \in Q$ is calculated.

Clearly, for any $e \in E_o$, we have $\boldsymbol{R}_{\{e\}} = \boldsymbol{R}_e = \boldsymbol{T}_e$. The remaining problem is how to determine the run matrix associated with E_{uo}^*. Considering the set of unobservable events E_{uo} as a language, we have $\boldsymbol{R}_{E_{uo}} = \sum_{e\in E_{uo}}\boldsymbol{R}_e = \sum_{e\in E_{uo}}\boldsymbol{T}_e$. By the second part of Proposition 4, we have $\boldsymbol{R}_{E_{uo}^i} = \boldsymbol{R}_{E_{uo}}^i$ for $i > 0$. Note that by $E_{uo}^0 = \{\varepsilon\}$, there is $\boldsymbol{R}_{E_{uo}^0} = \boldsymbol{R}_\varepsilon = \boldsymbol{I}$.

Theorem 1. *Given a system $G = (Q, E, f, Q_0)$ with no unobservable cycles, the run matrix $\boldsymbol{R}_{E^*_{uo}}$ associated with E^*_{uo} is $(\boldsymbol{I} - \boldsymbol{R}_{E_{uo}})^{-1}$.*

Proof. Since $E^*_{uo} = \{\varepsilon\} \cup E_{uo} \cup E^2_{uo} \cup \cdots$ and any two sets in $\{E^i_{uo} \mid i \in \mathbb{N}\}$ are disjoint, by Proposition 4, we have $\boldsymbol{R}_{E^*_{uo}} = \sum_{i\in\mathbb{N}} \boldsymbol{R}_{E^i_{uo}}$. Suppose that there exists a run r of length N generated by G with $N+1$ states, where N is the number of states in G. By Lemma 4, we can find a duplicate state q_d in r. This indicates the existence of an unobservable cycle in r, which violates the premise that there are no unobservable cycles in G. Therefore, the length of unobservable runs generated by G cannot be greater than N. Naturally, we have $\boldsymbol{R}_{E^i_{uo}} = \boldsymbol{O}$ for all $i > N$. From this, we obtain $\boldsymbol{R}_{E^*_{uo}} = \sum_{i=0}^{N} \boldsymbol{R}_{E^i_{uo}}$.

$$\boldsymbol{R}_{E^*_{uo}} \cdot (\boldsymbol{I} - \boldsymbol{R}_{E_{uo}}) = (\boldsymbol{I} + \boldsymbol{R}_{E_{uo}} + \boldsymbol{R}^2_{E_{uo}} + \cdots + \boldsymbol{R}^N_{E_{uo}}) \cdot (\boldsymbol{I} - \boldsymbol{R}_{E_{uo}}) = \boldsymbol{I} - \boldsymbol{R}_{E^{N+1}_{uo}} = \boldsymbol{I} \quad (3)$$

Furthermore, by (3), we conclude that the matrix $(\boldsymbol{I} - \boldsymbol{R}_{E_{uo}})$ is invertible and $\boldsymbol{R}_{E^*_{uo}} = (\boldsymbol{I} - \boldsymbol{R}_{E_{uo}})^{-1}$. □

Remark 4. *In the following, we will notate the run matrix associated with E^*_{uo} as $\boldsymbol{R}_{uo}$ for brevity of notation. If there are no unobservable events in the system G, then $\boldsymbol{R}_{E_{uo}} = \boldsymbol{O}$, leading to $\boldsymbol{R}_{uo} = \boldsymbol{I}$ without affecting the results below.*

In light of the above discussion, it is natural to present Algorithm 1 for the online verification of worthy opacity. Lines 1 to 6 are the initialization of the marker variable $flag$ and the count vector $\boldsymbol{\pi}$, where $flag$ is set to be *True* to indicate that the system is worthy opaque, and the i-th entry of $\boldsymbol{\pi}$ indicates the number of system-generated runs ending in state q_i, inferred by an intruder from the observation. Lines 7 to 21 are the intruder's online judgment of worthy opacity, whereas lines 8 to 12 are calculations of the expected worth of the currently revealed information. Once the expected worth exceeds K, which means that the system is not worthy opaque, set the $flag$ to be *False* and stop observing as shown in lines 13 to 16; otherwise, continue the observation as shown in lines 17 to 20. As regards the complexity of Algorithm 1, we note that the complexity of the recursive step k (observe the kth event) is $\mathcal{O}(k \cdot N)$.

Example 8. *Consider the system G_1 in Example 1, where its state-worth function is shown in Example 4. Given an observation $\omega = e_1^2$, we verify online the $(E_o, \Delta, 100)$-worthy opacity of ω using Algorithm 1. Initially, ω is set to be the empty string, i.e., $\omega = \varepsilon$. By Algorithm 1, we have that the expected worth for this system is 50, which is less than 100, implying that the null observation ε is $(E_o, \Delta, 100)$-WO. Similarly, after observing e_1, the value of worth becomes $80 < 100$, so e_1 is also $(E_o, \Delta, 100)$-WO. Continuing to run the algorithm, after observing e_1 again, we have worth $= 100$, implying that e_1^2 is $(E_o, \Delta, 100)$-WO.*

Algorithm 1 Online verification of worthy opacity

Input: A system $G = (Q, E, f, Q_0)$ with $E = E_o \dot{\cup} E_{uo}$, a state-worth function Δ, and a non-negative value K

Output: (E_o, Δ, K)-WO of ω upon observing an event e

$flag \leftarrow True$, $\boldsymbol{\pi} \leftarrow \mathbf{0}$
for $i \leftarrow 1$ to N **do**
  if $q_i \in Q_0$ **then**
    $[\boldsymbol{\pi}]_i \leftarrow 1$
  end if
end for
while $flag$ **do**
  $\boldsymbol{\pi} \leftarrow \boldsymbol{R}_{uo} \cdot \boldsymbol{\pi}$, $worth \leftarrow 0$
  $\boldsymbol{\pi}_n \leftarrow \mathtt{norm}(\boldsymbol{\pi})$
  for $i \leftarrow 1$ to N **do**
    $worth \leftarrow worth + \Delta(q_i) \cdot [\boldsymbol{\pi}_n]_i$
  end for
  if $worth > K$ **then**
    $flag \leftarrow False$
  end if
  Output $flag$
  if $flag$ **then**
    Wait until a new event e is observed
    $\boldsymbol{\pi} \leftarrow \boldsymbol{T}_e \cdot \boldsymbol{\pi}$
  end if
end while

4.2. Run Status Recorder and 1*-Cycle Returned*

It is not difficult to find that for any observation ω, there is $\Phi_q(\omega) = \emptyset$ if $q \notin \mathcal{C}(\omega)$, which leads to $\alpha_\omega(q) = 0$ if $q \notin \mathcal{C}(\omega)$, and implies that we can narrow the computation from $\{\alpha_\omega(q) \mid q \in Q\}$ to $\{\alpha_\omega(q) \mid q \in \mathcal{C}(\omega)\}$. We call the set $\{\alpha_\omega(q) \mid q \in \mathcal{C}(\omega)\}$ the *current-state probability distribution estimate* associated with ω, denoted by $\mathcal{A}(\omega)$. Note that the observer mentioned in Definition 3 represents, in a compact structure, all possible current state estimates during the evolution of a system. Inspired by this idea, it seems that we can also represent all current-state probability distribution estimates in a finite structure. Unfortunately, as the system evolves, there may be an infinite number of sets $\mathcal{A}(\omega)$, as the sets $\mathcal{A}(\omega)$ and $\mathcal{C}(\omega)$ do not correspond one-to-one.

By analyzing the evolution of a system, we find that if there is no cycle in its observer. Then, the observations generated by that system are finite, which means that the number of set $\mathcal{A}(\omega)$ is also finite. Thus, the verification of worthy opacity can be achieved by traversing all observations to determine whether (1) holds. In contrast, if there is a cycle in the observer of a system, then this system can generate an infinite number of observations.

Given a cycle c_o of an observer, the system associated with that observer can generate an infinite number of observations of the form ωc_o^i ($i \in \mathbb{N}$), assuming that the observation when the observer first enters the cycle is ω. If $\mathcal{A}(\omega) = \mathcal{A}(\omega c_o)$, we say that the cycle is 1-cycle returned (1-CR), and if all cycles in an observer of a system are 1-CR, then we say that this system is 1-CR. Fortunately, if a system G is 1-CR, then the current-state probability distribution estimate that can be generated by G is finite, i.e., the worthy opacity of system G is verifiable, as shown in Example 6.

Given a system G, we check whether G is 1-CR by Algorithm 2 and, if so, construct a run status recorder, which is an automaton that enumerates all possible current-state probability distribution estimates generated by system G (see Figure 4). The core idea of Algorithm 2 comes from Proposition 6.

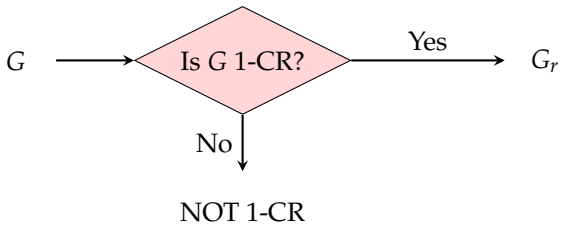


Figure 4. Input and output of Algorithm 2.

Proposition 6. *Given two N-dimensional column vectors* $\boldsymbol{\pi}_1$ *and* $\boldsymbol{\pi}_1$ *that satisfy* $\texttt{norm}(\boldsymbol{\pi}_1) = \texttt{norm}(\boldsymbol{\pi}_2)$, for any $N \times N$ matrix $\boldsymbol{R}$, it holds that $\texttt{norm}(\boldsymbol{R} \cdot \boldsymbol{\pi}_1) = \texttt{norm}(\boldsymbol{R} \cdot \boldsymbol{\pi}_2)$.

Proof. Let $\texttt{norm}(\boldsymbol{\pi}_1) = \texttt{norm}(\boldsymbol{\pi}_2) = \boldsymbol{\pi}$. We have $\boldsymbol{\pi}_1 = k_1\boldsymbol{\pi}$ and $\boldsymbol{\pi}_2 = k_2\boldsymbol{\pi}$, where k_1 and k_2 are 1-norms of $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$, respectively. Hence, one obtains

$$\texttt{norm}(\boldsymbol{R} \cdot \boldsymbol{\pi}_1) = \frac{\boldsymbol{R} \cdot k_1\boldsymbol{\pi}}{\|\boldsymbol{R} \cdot k_1\boldsymbol{\pi}\|} = \frac{k_1\boldsymbol{R} \cdot \boldsymbol{\pi}}{k_1\|\boldsymbol{R} \cdot \boldsymbol{\pi}\|} = \frac{k_2\boldsymbol{R} \cdot \boldsymbol{\pi}}{k_2\|\boldsymbol{R} \cdot \boldsymbol{\pi}\|} = \frac{\boldsymbol{R} \cdot k_2\boldsymbol{\pi}}{\|\boldsymbol{R} \cdot k_2\boldsymbol{\pi}\|} = \texttt{norm}(\boldsymbol{R} \cdot \boldsymbol{\pi}_2).$$

This completes the proof. □

Algorithm 2 Construction of the run status recorder and and verification of the 1-CR of system G

Input: A system $G = (Q, E, f, Q_0)$ with $E = E_o \dot{\cup} E_{uo}$
Output: Run status recorder $G_r = (Y, E_o, f_r, y_0)$

for $i \leftarrow 1$ to N **do**
  if $q_i \in Q_0$ **then**
    $[\boldsymbol{\pi}]_i \leftarrow 1$
  end if
end for
$\boldsymbol{\pi} \leftarrow \boldsymbol{R}_{uo} \cdot \boldsymbol{\pi}$
$y_0 \leftarrow (\boldsymbol{\pi}, \texttt{norm}(\boldsymbol{\pi}))$
$Y \leftarrow \{y_0\}, ns \leftarrow \{\texttt{norm}(\boldsymbol{\pi})\}$
$un \leftarrow \{y_0\}$
while $un \neq \emptyset$ **do**
  Select a state $y \in un$
  for all $e \in E_o$ **do**
    $\boldsymbol{\pi} \leftarrow \boldsymbol{R}_{uo} \cdot \boldsymbol{T}_e \cdot y(1)$
    if $\boldsymbol{\pi} \neq \boldsymbol{0}$ and $\texttt{norm}(\boldsymbol{\pi}) \notin ns$ **then**
      if $\exists y' \rightarrow y$ such that $\texttt{eig}(y'(1)) = \texttt{eig}(\boldsymbol{\pi})$ **then**
        return NOT 1-CR
      end if
      $ns \leftarrow ns \cup \{\texttt{norm}(\boldsymbol{\pi})\}$
      $y_n \leftarrow (\boldsymbol{\pi}, \texttt{norm}(\boldsymbol{\pi})), Y \leftarrow Y \cup \{y_n\}$
      $un \leftarrow un \cup \{y_n\}$
    else
      Select $y_n \in Y$ with $y_n(2) = \texttt{norm}(\boldsymbol{\pi})$
    end if
    $f_r(y, e) \leftarrow y_n$
  end for
  $un \leftarrow un \setminus \{y\}$
end while

Each state in the run status recorder G_r is an ordered pair of two vectors, whose first component is the count vector mentioned in Algorithm 1, denoting the number of runs that reach each state after the system generates a certain observation. Its second component is the L1-normalization of the first vector. Proposition 6 shows that we can merge count

vectors that have the same L1-normalization because we are only interested in the current-state probability distribution estimates, which is why we use the second component of the state to identify the different states.

Algorithm 2 works as follows: Lines 1 to 9 are the initialization of the key parameters of the algorithm, where set *ns* includes all possible current-state probability distribution estimates generated by the input system and set *un* contains the states generated by the algorithm for which subsequent states are not computed. Function $\texttt{eig}\colon \mathbb{N}^N \to \mathbb{N}^N$ in line 15 of Algorithm 2 is defined as $[\texttt{eig}(\boldsymbol{\pi})]_i = 1$ if $[\boldsymbol{\pi}]_i \neq 0$ otherwise $[\texttt{eig}(\boldsymbol{\pi})]_i = 0$, representing a current-state estimate of system G. Once the judgment condition in line 15 is satisfied, it means that the newly generated current-state probability distribution estimate, which different from some previously calculated current-state probability distribution estimate, corresponds to the same current-state estimate, i.e., this system is not 1-CR.

The construction of a run status recorder shows that the number of its states is related to the number of cycles in the observer of the input system G. The number of cycles in an observer of system G containing i different states is $\binom{2^N}{i}(i-1)!$, where the observer is considered as the worst case. Then, these $\binom{2^N}{i}(i-1)!$ cycles can produce $i \cdot \binom{2^N}{i}(i-1)! = (2^N)!/(2^N - i)!$ different states in G_r in the worst case. Therefore, the complexity of Algorithm 2 is $\mathcal{O}(2^N!)$. Although the complexity of Algorithm 2 looks formidable, there are few systems that can reach the worst case that we analyze. If there is no cycle in the observer of a system, then the complexity of Algorithm 2 is reduced to $\mathcal{O}(2^N)$.

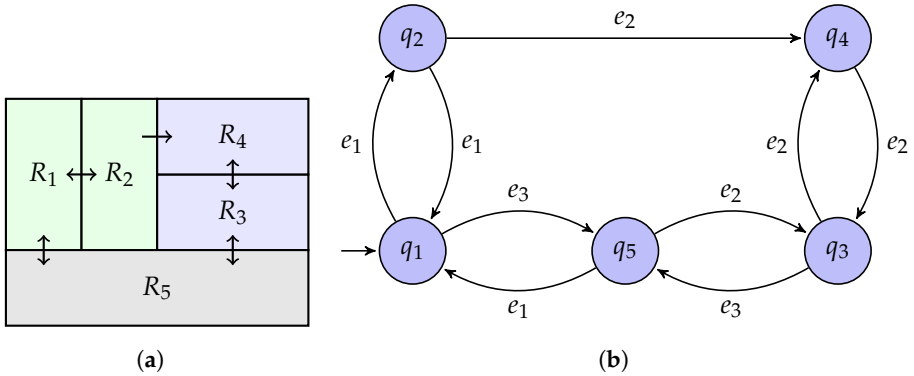
Theorem 2. *Given a system $G = (Q, E, f, Q_0)$ with state-worth function Δ, a set of observable events $E_o \subseteq E$, and a non-negative value $K \in \mathbb{R}_{\geq 0}$, if system G is 1-CR, then construct the run status recorder $G_r = (Y, E_o, f_r, y_0)$ as in Algorithm 2. System G is worthy opaque with respect to E_o, Δ and K if and only if for any states $y \in Y$, $\boldsymbol{\Delta} \cdot y(2) \leq K$ holds, where $\boldsymbol{\Delta}$ is an N-dimensional row vector with $[\boldsymbol{\Delta}]_i = \Delta(q_i)$.*

Proof. (Sufficiency) By contrapositive, assume that there exists a state y in G_r such that $\boldsymbol{\Delta} \cdot y(2) > K$. Let ω be the observation that leads to y from y_0. By the construction of G_r, we have that, for the observation ω, $\sum_{q \in Q} \alpha_\omega(q) \cdot \Delta(q) > K$ holds. By Definition 6, G is not worthy opaque with respect to E_o, Δ and K.

(Necessity) Also by contrapositive, assume that G is not worthy opaque with respect to E_o, Δ and K. By Definition 6, there exists an observation ω' such that $\sum_{q \in Q} \alpha_{\omega'}(q') \cdot \Delta(q) > K$. By the construction of G_r, we conclude that there is a state $y' \in Y$ with $\boldsymbol{\Delta} \cdot y(2) > K$. □

Example 9. *Consider a single-story smart building with five rooms marked with R_1, R_2, R_3, R_4, and R_5, as shown in Figure 5a. A bi-directional door exists between R_1 and R_2, R_3 and R_4, R_3 and R_5, R_1 and R_5, while R_2 to R_4 have a uni-directional door. And these five rooms are provided to three different departments according to R_1 and R_2, R_3 and R_4, and R_5. A smart vehicle provides different services to staff depending on the department, with sensors inside the vehicle that identify the current department. Five levels of confidential documents need to be stored in these five rooms, ranging in value from 1 to 5. Now, suppose that an intruder can read the sensor data of a smart vehicle; how does one allot the five rooms to store confidential documents such as to minimize the worth of information exposed by the vehicle?*

The vehicle's trajectory can be represented by the model shown in Figure 5b. The five states q_1 to q_5 correspond to the five rooms R_1 to R_5, respectively. Events e_1 to e_3 represent the sensor readings of the vehicle reaching each room, respectively. Since the intruder does not know the exact location of the vehicle but can obtain the sensor readings, the initial state of this model is $Q_0 = Q = \{q_1, q_2, q_3, q_4, q_5\}$ and the observable event set is $E_o = E = \{e_1, e_2, e_3\}$. Now, running Algorithm 2 with system G_3 as input, we can obtain the corresponding run status recorder $G_{r,3}$ as shown in Figure 6.



(**a**) (**b**)

Figure 5. (**a**) A single-story building with five rooms and (**b**) the constructed automaton model G_3.

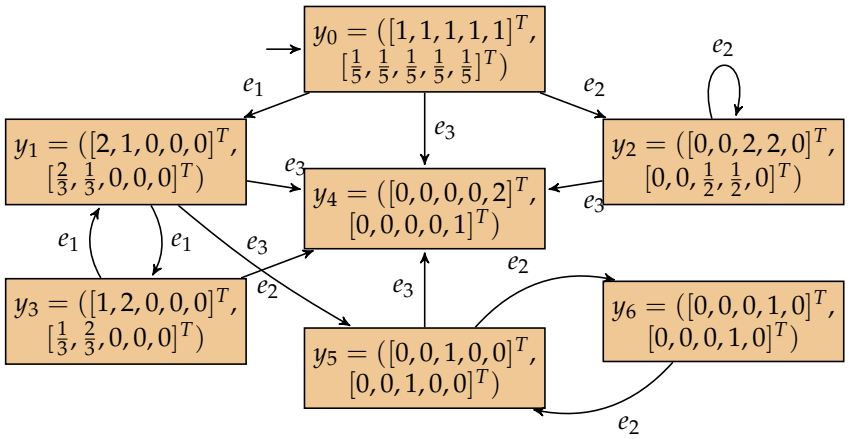


Figure 6. The run status recorder of G_3.

By assumption, the state-worth function Δ of system G_3 is a one-to-one mapping from Q to $\{1,2,3,4,5\}$. Let $\mathbf{\Delta}$ be an N-dimensional row vector with $[\mathbf{\Delta}]_i = \Delta(q_i)$. Then, the problem of allotting the rooms where the confidential documents are to be stored is transformed into how to define the function Δ such that the maximum value in $\{\mathbf{\Delta} \cdot y_i(2) \mid i = 0,1,2,3,4,5,6\}$ is minimized. Based on states y_4, y_5, and y_6, we know that the two documents with the highest level of confidentiality can only be placed in rooms R_1 and R_2, respectively. Once this is done, it is not difficult to verify that $\max\{\mathbf{\Delta} \cdot y_i(2) \mid i = 0,1,2,3,4,5,6\} = \mathbf{\Delta} \cdot y_0(2) = 3$*, which means the system G_3 is $(E_0, \Delta, 3)$-WO.*

In plain words, we only need to place the documents with the highest and second-highest level of confidentiality in rooms R_1 and R_2, respectively; then, no matter how the sensor data of the vehicle is exposed, the worth of the information it reveals to the outside world will not exceed 3.

5. Conclusions

In this article, we introduce a notion of worthy opacity to quantify the worth of information released by a partially observed DES modeled with an automaton. We propose an online verification algorithm that considers an intruder who waits and observes the occurrence of observable events and determines whether the observation is worthy opaque. We also present a 1-CR system to verify the worthy opacity offline.

We believe that there are multiple interesting directions for future work related to the notion of worthy opacity. An attractive direction is to synthesize a supervisor that enforces worthy opacity when the verification result is negative. Also, we would like to apply the notion of worthy opacity to more complex, realistic environments.

Author Contributions: Conceptualization, S.Z. and L.Y.; methodology, S.Z.; software, S.Z.; validation, J.Y. and L.Y.; formal analysis, Z.L.; investigation, J.Y.; resources, Z.L.; writing— original draft preparation, S.Z.; writing—review and editing, Z.L.; visualization, S.Z.; supervision, L.Y.; project administration, L.Y. and Z.L.; funding acquisition, L.Y. and Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Guangzhou Innovation and Entrepreneurship Leading Team Project Funding under grant No. 202009020008 and the Science and Technology Fund, FDCT, Macau SAR, under grant No. 0101/2022/A.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DES	Discrete Event System
DFA	Deterministic Finite Automaton
IoT	Internet of Things
NFA	Non-deterministic Finite Automaton
1-CR	1-Cycle Returned

References

1. Khor, N.; Arimah, B.; Otieno, R.; Oostrum, M.; Mutinda, M.; Martins, J. World Cities Report 2022: Envisaging the Future of Cities. 2022. Available online: https://unhabitat.org/sites/default/files/2022/06/wcr_2022.pdf (accessed on 29 June 2022).
2. Yang, J.; Lee, T.Y.; Zhang, W. Smart cities in China: A brief overview. *IT Prof.* **2021**, *23*, 89–94. [CrossRef]
3. Jia, M.; Komeily, A.; Wang, Y.; Srinivasan, R.S. Adopting Internet of Things for the development of smart buildings: A review of enabling technologies and applications. *Autom. Constr.* **2019**, *101*, 111–126. [CrossRef]
4. Verma, A.; Prakash, S.; Srivastava, V.; Kumar, A.; Mukhopadhyay, S.C. Sensing, controlling, and IoT infrastructure in smart building: A Review. *IEEE Sens. J.* **2019**, *19*, 9036–9046. [CrossRef]
5. Shaikh, P.H.; Nor, N.B.M.; Nallagownden, P.; Elamvazuthi, I.; Ibrahim, T. A review on optimized control systems for building energy and comfort management of smart sustainable buildings. *Renew. Sustain. Energy Rev.* **2014**, *34*, 409–429. [CrossRef]
6. Carli, R.; Cavone, G.; Dotoli, M.; Epicoco, N.; Scarabaggio, P. Model predictive control for thermal comfort optimization in building energy management systems. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 2608–2613.
7. Ascione, F.; Bianco, N.; De Stasio, C.; Mauro, G.M.; Vanoli, G.P. Simulation-based model predictive control by the multi-objective optimization of building energy performance and thermal comfort. *Energy Build.* **2016**, *111*, 131–144. [CrossRef]
8. Komninos, N.; Philippou, E.; Pitsillides, A. Survey in smart grid and smart home security: Issues, challenges and countermeasures. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1933–1954. [CrossRef]
9. Wendzel, S. How to increase the security of smart buildings? *Commun. ACM* **2016**, *59*, 47–49. [CrossRef]
10. Hu, J.; Zhang, Z.; Lu, J.; Yu, J.; Cao, J. Demand response control of smart buildings integrated with security interconnection. *IEEE Trans. Cloud Comput.* **2022**, *10*, 43–55. [CrossRef]
11. Mazaré, L. Using unification for opacity properties. In Proceedings of the 4th IFIP WG 1.7, ACM SIGPLAN and GI FoMSESS Workshop on Issues in the Theory of Security, Barcelona, Spain, 3–4 April 2004.
12. Bryans, J.; Koutny, M.; Mazaré, L.; Ryan, P. Opacity generalised to transition systems. *Int. J. Inf. Secur.* **2008**, *7*, 421–435. [CrossRef]
13. Lin, F. Opacity of discrete event systems and its applications. *Automatica* **2011**, *47*, 496–503. [CrossRef]
14. Jacob, R.; Lesage, J.J.; Faure, J.M. Overview of discrete event systems opacity: Models, validation, and quantification. *Annu. Rev. Control* **2016**, *41*, 135–146. [CrossRef]
15. Tong, Y.; Ma, Z.; Li, Z.; Seatzu, C.; Giua, A. Verification of language-based opacity in Petri nets using verifier. In Proceedings of the 2016 American Control Conference, Boston, MA, USA, 6–8 July 2016.
16. Saboori, A.; Hadjicostis, C.N. Notions of security and opacity in discrete event systems. In Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, LA, USA, 12–14 December 2007.
17. Dong, Y.; Li, Z.; Wu, N. Symbolic verification of current-state opacity of discrete event systems using Petri nets. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 7628–7641. [CrossRef]
18. Saboori, A.; Hadjicostis, C.N. Verification of initial-state opacity in security applications of discrete event systems. *Inf. Sci.* **2013**, *246*, 115–132. [CrossRef]
19. Wu, Y.C.; Lafortune, S. Comparative analysis of related notions of opacity in centralized and coordinated architectures. *Discret. Event Dyn. Syst.* **2013**, *23*, 307–339. [CrossRef]

20. Saboori, A.; Hadjicostis, C.N. Verification of *K*-step opacity and analysis of its complexity. *IEEE Trans. Autom. Sci. Eng.* **2011**, *8*, 549–559. [CrossRef]
21. Saboori, A.; Hadjicostis, C.N. Verification of infinite-step opacity and complexity considerations. *IEEE Trans. Autom. Control* **2011**, *57*, 1265–1269. [CrossRef]
22. Yang, S.; Yin, X. Secure Your Intention: On Notions of Pre-Opacity in Discrete-Event Systems. *IEEE Trans. Autom. Control* **2023**, *68*, 4754–4766. [CrossRef]
23. Bérard, B.; Mullins, J.; Sassolas, M. Quantifying opacity. *Math. Struct. Comput. Sci.* **2015**, *25*, 361–403. [CrossRef]
24. Saboori, A.; Hadjicostis, C.N. Current-state opacity formulations in probabilistic finite automata. *IEEE Trans. Autom. Control* **2014**, *59*, 120–133. [CrossRef]
25. Li, D.; Yin, L.; Wang, J.; Wu, N. Game current-state opacity formulation in probabilistic resource automata. *Inf. Sci.* **2022**, *613*, 96–113. [CrossRef]
26. Bourouis, A.; Klai, K.; Hadj-Alouane, N.B. Measuring opacity for non-probabilistic DES: A SOG-based approach. In Proceedings of the 24th International Conference on Engineering of Complex Computer Systems, Guangzhou, China, 10–13 November 2019.
27. Cassandras, C.G.; Lafortune, S. *Introduction to Discrete Event Systems*; Springer Nature: Cham, Switzerland, 2021.
28. Tong, Y.; Li, Z.; Seatzu, C.; Giua, A. Verification of state-based opacity using Petri nets. *IEEE Trans. Autom. Control* **2017**, *62*, 2823–2837. [CrossRef]
29. Jiang, S.; Kumar, R.; Garcia, H.E. Diagnosis of repeated/intermittent failures in discrete event systems. *IEEE Trans. Robot. Autom.* **2003**, *19*, 310–323. [CrossRef]
30. Reinhardt, K. Counting as Method, Model and Task in Theoretical Computer Science. Habilitation Thesis, University of Tübingen, Tübingen, Germany, 2005.
31. Bertsekas, D.P.; Tsitsiklis, J.N. *Introduction to Probability*; Athena Scientific: Nashua, New Hampshire, 2008.
32. Brualdi, R. *Introductory Combinatorics*; Pearson Education: Upper Saddle River, NJ, USA, 2010.
33. Rosen, K. *Discrete Mathematics and Its Applications*; McGraw-Hill: New York, NY, USA, 2019.
34. Hadjicostis, C.N. *Estimation and Inference in Discrete Event Systems: A Model-Based Approach with Finite Automata*; Springer: New York, NY, USA, 2020.
35. Blizard, W. Multiset theory. *Notre Dame J. Form. Log.* **1989**, *30*, 36–66. [CrossRef]