Automatica 140 (2022) 110238

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

A framework for current-state opacity under dynamic information release mechanism☆

Junyao Hou, Xiang Yin *, Shaoyuan Li

Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China

ARTICLE INFO

Article history:
Received 7 December 2020
Received in revised form 23 August 2021
Accepted 20 January 2022
Available online xxxx

Keywords:
Opacity
Discrete-event systems
Security
Information release

ABSTRACT

Opacity is an important information-flow security property that characterizes the plausible deniability of a dynamic system for its "secret" against eavesdropping attacks. As an information-flow property, the underlying observation model is the key in the modeling and analysis of opacity. In this paper, we investigate the verification of current-state opacity for discrete-event systems under Orwellian-type observations, i.e., the system is allowed to re-interpret the observation of an event based on its future suffix. First, we propose a new Orwellian-type observation model called the dynamic information release mechanism (DIRM). In the DIRM, when to release previous "hold on" events is state-dependent. Then we propose a new definition of opacity based on the notion of history-equivalence rather than the standard projection-equivalence. This definition is more suitable for observations that are not prefix-closed. Finally, we show that by constructing a new structure called the DIRM-observer, current-state opacity can be effectively verified under the DIRM. Computational complexity analysis as well as illustrative examples for the proposed approach is also provided. Compared with the existing Orwellian-type observation model, the proposed framework is more general in the sense that the information-release-mechanism is state-dependent, information is partially released and the corresponding definition of opacity is more suitable for non-prefix-closed observations.

1. Introduction

Security and privacy issues are becoming pervasive in safety-critical cyber–physical systems as computational devices nowadays are connected by networks which may lead to information leakage. For dynamic systems, an important angle for understanding the security-level of a system is to analyze what information of importance can revealed via information-flow. Opacity is one of the most widely adopted information-flow security properties for dynamic systems whose information-flow is available to eavesdroppers or passive observers that are potentially malicious. Essentially, opacity captures the plausible deniability of the system for its "secret" behavior by requiring that the secret behavior can never be identified unambiguously by the intruder based on the online information-flow and the dynamic of the system.

In this paper, we investigate opacity in the context of discrete event systems (DES), which has drawn considerable attentions in the last decade; see, e.g., the recent surveys (Jacob, Lesage, & Faure, 2016; Lafortune, Lin, & Hadjicostis, 2018) and the textbook (Hadjicostis, 2020). The basic concept of opacity was originally introduced by Bryans, Koutny, and Ryan (2005) and Bryans, Koutny, Mazaré, and Ryan (2008) for dynamic systems modeled as transition systems. Depending on different secret requirements, different notions of opacity were proposed in the literature, including, e.g., initial-state opacity (Cong, Fanti, Mangini, & Li, 2019; Saboori & Hadjicostis, 2013), current-state opacity (Wu & Lafortune, 2013) and K/infinite-step opacity (Saboori & Hadjicostis, 2011, 2012; Yin & Lafortune, 2017). For example, current-state opacity requires that the outsider/intruder can never determine for sure that the system is at a secret state based on the information-flow. When the original system is not opaque, many different approaches have also been proposed for the enforcement of opacity; see, e.g., Barcelos and Basilio (2018), Dubreil, Darondeau, and Marchand (2010), Falcone and Marchand (2015), Ji, Yin, and Lafortune (2019), Liu, Mei, and Lu (2020), Mohajerani, Ji, and Lafortune (2020), Takai and Oka (2008) and Zinck, Ricker, Marchand, and Hélouët (2020). More recently, the concept of opacity has been further generalized to continuous dynamic systems with infinite state–spaces and time-driven dynamics; see, e.g., An and Yang (2020), Liu and Zamani

☆ This work was supported by the National Natural Science Foundation of China (62061136004, 62173226, 61833012) and the National Key R&D Program, China (2018AAA0101700). The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Christoforos Hadjicostis under the direction of Editor Christos G. Cassandras.

* Corresponding author.
E-mail addresses: houjunyao@sjtu.edu.cn (J. Hou), yinxiang@sjtu.edu.cn (X. Yin), syli@sjtu.edu.cn (S. Li).

https://doi.org/10.1016/j.automatica.2022.110238
0005-1098/

(2020), Ramasubramanian, Cleaveland, and Marcus (2020) and Yin, Zamani, and Liu (2021). In this work, we will study the verification of current-state opacity for finite systems.

Since the essence of opacity is an information-flow security property, the underlying observation model of the system becomes the key in its modeling and analysis. In the DES literature, there are three different types of observation models that have been investigated: static observation, dynamic observation and Orwellian observation. These three types were originally categorized in Bryans et al. (2008). Here we summarize and explain them in detail as follows.

- Static Observation: In this setting, it is assumed that the event set is partitioned as observable and unobservable events, and the outsider can observe observable events immediately upon their occurrences. Such an observation model is called static because whether or not an event can be observed is fixed. Hence, the information-flow of a generated internal string is essentially its natural projection. This is the most simple but probably also the most widely investigated observation model for the analysis of opacity; see, e.g., Balun and Masopust (2019), Chen, Ibrahim, and Kumar (2017), Keroglou and Hadjicostis (2017), Lin (2011), Mohajerani and Lafortune (2020), Saadaoui, Li, and Wu (2020), Tong, Li, Seatzu, and Giua (2017) and Yin, Li, Wang, and Li (2019).
- Dynamic Observation: In this setting, it is assumed that whether or not the occurrence of an event can be observed is not fixed a priori and may depend on the *prefix* string up to the current instant. This model can describe the scenario where the observability of each event is "controlled" by an active information acquisition module. Such an information acquisition module is also referred to as the sensor activation policy (Lin, Chen, Wang, & Wang, 2020; Thorsley & Teneketzis, 2007; Wang, Lafortune, Girard, & Lin, 2010; Yin & Lafortune, 2019), the dynamic mask (Cassez, Dubreil, & Marchand, 2012; Yin & Li, 2020) or the information release module (Behinaein, Lin, & Rudie, 2019; Zhang, Shu, & Lin, 2015) in the literature depending on the context of the underlying problem.
- Orwellian Observation: Compared with the dynamic observation, in the setting of Orwellian observation, the observability of an event not only depends on the prefix up to the current instant, but also depends on the future *suffix*. In other words, the system is allowed to re-interpret the observation of an internal string. In general, however, deciding opacity under an arbitrary Orwellian observation is undecidable even for finite systems (Bryans et al., 2008). In Mullins and Yeddes (2014) and Yeddes (2016), the authors formulated a simple Orwellian-type observation using the notion of downgrading events, and showed that the corresponding opacity verification problem is decidable by relating it to the notion of intransitive non-interference (Ben Hadj-Alouane, Lafrance, Lin, Mullins, & Yeddes, 2005a, 2005b). In Bérard and Mullins (2014), the notion of rational observation is proposed, which further generalizes the Orwellian projection proposed in Mullins and Yeddes (2014).

In this paper, we are interested in the verification of current-state opacity under Orwellian-type observation. As we discussed above, the main feature of Orwellian observation is that it allows to re-interpret the information of a previous event. Therefore, it is extremely useful in the modeling and analysis of information-flow security related to *information declassification*. For example, the user can classify the occurrence of an event and declassify/release it in the future when some particular conditions are fulfilled. This is a very common way how secure information is processed. However, existing Orwellian-type observations as well as their associated definitions of opacity cannot fully capture this scenario. For example, in the definition of Orwellian projection in Mullins and Yeddes (2014), those "on hold" information are released when a downgrading event occurs. However, in a more general setting, when to release those "on hold" information may be determined by a "controller" having its own logic rather than simply depending on the event of the original system. Furthermore, in Mullins and Yeddes (2014), it is assumed that once a downgrading event occurs, the entire trajectory is revealed, i.e., the outsider knows the current state precisely. In practice, however, it is possible that only partial history information is declassified and the outsider may still have ambiguity after the declassification.

More importantly, existing definitions of current-state opacity under Orwellian-type observations still follow the standard projection-based definition by requiring that for any secret string, there exists a non-secret string such that they have the *same projection*, e.g., Bérard and Mullins (2014) and Mullins and Yeddes (2014). However, this definition is not exactly suitable for Orwellian-type observation since the observation sequence may not be prefix-closed. In particular, that two strings have the same projection does not necessarily imply that their prefixes also have the same projection. If the prefixes of two projection-equivalent strings are not projection-equivalent, then the intruder may still be able to determine that the system is currently at a secret state even when the standard projection-based opacity condition holds. This scenario will be explained in more detail in Section 3. Therefore, a new definition of opacity as well as the associated verification algorithms is needed for Orwellian-type observation.

Motivated by the above discussions, in this paper, we propose a new Orwellian-type observation framework for information-flow security and investigate the underlying opacity verification problem. Compared with the existing works, the main contributions of this work are as follows.

- First, we propose a new Orwellian-type observation model called the *Dynamic Information Release Mechanism* (DIRM). The DIRM is motivated by Mullins and Yeddes (2014) but has the following main difference. Compared with Mullins and Yeddes (2014) where downgrading events are used to trigger information release, in the DIRM, when to release those "hold on" information is *state-dependent*. This model captures the scenario where information declassification can be controlled by another agent that is not embedded in the original plant model. Furthermore, we investigate the partial information release setting, which is more general than the full information release setting in Mullins and Yeddes (2014), where the entire trajectory will be available when the information is released.
- Second, we propose a new definition of current-state opacity that is more suitable for Orwellian-type observations. Specifically, we introduce the concept of observation history as the set of all projections along the prefixes of a string. Then we say that a system is current-state opaque if for any secret string, there exists a non-secret string such that they have the same *observation history*. The new history-based definition is more suitable than the standard projection-based definition, since the observation history may not be prefix-closed for Orwellian-type observations.
- Finally, we propose an effective algorithm for the verification of the proposed notion of current-state opacity under the DIRM. Our approach is based on the notion of augmented system and a new information structure called the DIRM-observer. Specifically, the augmented system augments the original state–space by an additional binary information that tracks whether or not there is information to

be released on hold. The DIRM-observer essentially tracks both the current-state estimate and the information that is held on. When an information-release-state is reached, it updates the current-state-estimate by effectively fusing these two parts of information together. Then we show that the opacity verification problem can be solved by a simple reachability search within the DIRM-observer.

The rest of this paper is organized as follows. In Section 2, some necessary preliminaries are introduced. In Section 3, we present the dynamic information release mechanism as the underlying Orwellian-type observation model and define current-state opacity under the DIRM projection. To verify opacity, we first introduce the concept of augment system in Section 4 and show that the verification problem can be investigated equivalently for the augmented system. In Section 5, we present the DIRM-observer structure and show how to use it to verify opacity. A case study on security issues in federated cloud computing systems is provided in Section 6. Finally, we conclude the paper in Section 7.

2. Preliminaries

2.1. System model

Let Σ be a finite set of events. A string is a finite sequence of events and we denote by Σ^* the set of all strings over Σ including the empty string ϵ. For any string $s = \sigma_1\sigma_2\ldots\sigma_n$, we denote by $|s|$ the length of string s, i.e., $|s| = n$ and $|\epsilon| = 0$; we also denote by $s[i,j]$ the sub-string of s from the ith event to the jth event, i.e., $s[i,j] = \sigma_i\sigma_{i+1}\ldots\sigma_j$. In particular, we define $s[i,j] = \epsilon$ if $j < i$ and define $s[i,j] = s[i,|s|]$ if $|s| < j$. A language $L \subseteq \Sigma^*$ is a set of strings. For any language $L \subseteq \Sigma^*$, we denote by $\bar{L}$ its *prefix-closure*, i.e., $\bar{L} = \{t \in \Sigma^* : \exists w \in \Sigma^* \text{ s.t. } tw \in L\}$.

We consider a DES modeled as a deterministic finite-state automaton (DFA)

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

where X is a finite set of states, Σ is a finite set of events, $\delta : X \times \Sigma \to X$ is the partial transition function, and $x_0 \in X$ is the initial state. For any $x, x' \in X$, $\sigma \in \Sigma$, $\delta(x,\sigma) = x'$ means that there exists a transition from state x to state x' labeled with event σ. For the sake of simplicity, we write $\delta(x,s)$ as $\delta(s)$ when $x = x_0$. The transition function is also extended to $\delta : X \times \Sigma^* \to X$ recursively by: $\delta(x,\epsilon) = x$ and $\delta(x,s\sigma) = \delta(\delta(x,s),\sigma)$. The language generated by G is $\mathcal{L}(G) = \{s \in \Sigma^* : \delta(x_0,s)!\}$, where "!" means "is defined".

2.2. Current-state opacity under natural projection

In the analysis of information-flow security, it is often assumed that the event set is partitioned as $\Sigma = \Sigma_o \dot\cup \Sigma_{uo}$, where Σ_o is the set of observable events and Σ_{uo} is the set of unobservable events. Let $\hat\Sigma \subseteq \Sigma$ be a set of events. Then the natural projection from Σ to $\hat\Sigma$ is a mapping $P_{\hat\Sigma} : \Sigma^* \to \hat\Sigma^*$ defined recursively by: for any $s \in \Sigma^*$, $\sigma \in \Sigma$, we have

$$P_{\hat\Sigma}(\epsilon) = \epsilon \quad \text{and} \quad P_{\hat\Sigma}(s\sigma) = \begin{cases} P_{\hat\Sigma}(s)\sigma & \text{if } \sigma \in \hat\Sigma \\ P_{\hat\Sigma}(s) & \text{if } \sigma \notin \hat\Sigma \end{cases}.$$

In the context of opacity, we assume the underlying system has some "secret" modeled as a set of secret states $X_S \subset X$. Furthermore, there is a passive intruder modeled as an observer that knows the system model and can eavesdrop the projected information flow, i.e., it can observe the occurrences of events in Σ_o. Then current-state opacity requires that the intruder should never know for sure that the system is currently at a secret state, which is defined as follows.

Definition 1. Given system G and secret states $X_S \subset X$, we say system G is *current-state opaque* (w.r.t. X_S and Σ_o), if for any string $s \in \mathcal{L}(G)$ such that $\delta(s) \in X_S$, there exists another string $s' \in \mathcal{L}(G)$ such that $\delta(s') \in X \setminus X_S$ and $P_{\Sigma_o}(s) = P_{\Sigma_o}(s')$.

3. Opacity under dynamic information release mechanism

3.1. Dynamic information release mechanism

In the standard analysis of opacity under natural projection, the observation of each observable event is assumed to be *instant* in the sense that the occurrence of each observable event can be observed immediately and the occurrence of each unobservable event can never be observed. In some applications, however, the event set cannot be simply partitioned as observable and unobservable. In practice, the occurrence of some event may be classified until the system decides to *release (or declassify)* it. This is a very common scenario in information-flow security problem, e.g., some secure documents will be declassified only when certain conditions are satisfied. This leads to the *dynamic information release mechanism* (DIRM) defined as follows.

Formally, we assume that the event set is partitioned as

$$\Sigma = \Sigma_o \dot\cup \Sigma_{uo} \dot\cup \Sigma_r,$$

where

- Σ_o is the standard set of observable events whose occurrences can be observed instantly;
- Σ_{uo} is the standard set of unobservable events whose occurrences can never be observed;
- Σ_r is the set of events whose occurrences may not be observed instantly but can be *released* in the future.

To formally describe how the information is released, we consider a *state-based* dynamic information release mechanism specified by a function

$$R : X \to \{\mathbf{u}, \mathbf{r}\},$$

where "**r**" and "**u**" stand for "release" and "unrelease", respectively. One can image that the system has an "information release button" for events in Σ_r and the release button is pressed when a state x such that $R(x) = \mathbf{r}$ is encountered. In such a case, all events in Σ_r along the trajectory that have not yet be observed will be released at state x in the sense that the observer knows when they occurred.

Remark 1. Here we use function R to emphasize the fact that the information release mechanism is *state dependent*. Equivalently, we can also define $X_r = \{x \in X : R(x) = \mathbf{r}\}$ as the set of states at which Σ_r can be released. Hereafter, we will only use X_r instead of function R and also refer to X_r as the DIRM, for the sake of simplicity. Also, without loss of generality but for the sake of simplicity, an event will not be released immediately after its occurrence, i.e., $\forall s \in \mathcal{L}(G), \sigma \in \Sigma_r$: $\delta(s\sigma) \notin X_r$.

Remark 2. The dynamic information release mechanism defined here is state-based. In general, it can be a language-based function specified by a finite-state transducer. In this case, one can take the product of the transducer with the plant to refine the state–space such that the release mechanism becomes state-based.

3.2. The DIRM projection

Let $s \in \mathcal{L}(G)$ be a string generated by the system. We denote by $0 \le \imath_s \le |s|$ the latest instant when events in Σ_r are released, i.e.,

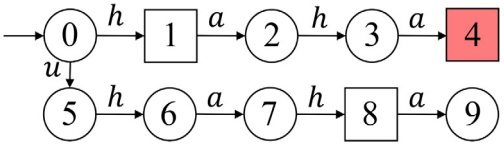


Fig. 1. System G with $\Sigma_o = \{a\}$, $\Sigma_r = \{h\}$, $\Sigma_{uo} = \{u\}$, $X_r = \{1, 4, 8\}$ and $X_S = \{4\}$.

- $\delta(x_0, s[1, \iota_s]) \in X_r$; and
- $\forall \iota_s < i \leq |s| : \delta(x_0, s[1, i]) \notin X_r$.

In the DIRM, the observation of a string s depends on its release status. We define the corresponding DIRM projection $P_R(s)$ of it as follows.

Definition 2. Given system G with partition $\Sigma = \Sigma_o \dot\cup \Sigma_{uo} \dot\cup \Sigma_r$ and DIRM $X_r \subseteq X$, the *DIRM projection* of string s, denote by $P_R(s)$, is defined by

$$P_R(s) = P_{\Sigma_o \cup \Sigma_r}(s[1, \iota_s]) P_{\Sigma_o}(s[\iota_s + 1, |s|]) \tag{1}$$

The intuition of the above definition is as follows. Recall that ι_s is the latest information release instant when s is executed. Therefore, events in Σ_r that occur before instant ι_s are released, which is captured by $P_{\Sigma_o \cup \Sigma_r}(s[1, \iota_s])$, while events in Σ_r that occur after ι_s are still unobservable, which is captured by $P_{\Sigma_o}(s[\iota_s + 1, |s|])$. Note that, when $\iota_s = |s|$, by definition we have $s[1, \iota_s] = s$ and $s[\iota_s + 1, |s|] = \epsilon$. Then in this case, we have $P_R(s) = P_{\Sigma_o \cup \Sigma_r}(s)$ as all events in Σ_r are released. Also, when $\iota_s = 0$, it means that there is no information released along the path. Therefore, by definition, we have $s[1, \iota_s] = \epsilon$ and $s[\iota_s + 1, |s|] = s$, which means that $P_R(s) = P_{\Sigma_o}(s)$. Also, the above definition implicitly assumes the DIRM has infinite memory for those unreleased information. We illustrate the DIRM projection by the following example.

Example 1. Let us consider system G shown in Fig. 1, where $X_r = \{1, 4, 8\}$, $\Sigma_o = \{a\}$, $\Sigma_r = \{h\}$, $\Sigma_{uo} = \{u\}$ and $X_S = \{4\}$. We use rectangles to denote states in X_r. For string $s = uh$, we have $\iota_s = 0$ since $\forall s' \in \overline{\{s\}} : \delta(x_0, s') \notin X_r$, i.e., no state in X_r is visited along s from the initial state. Therefore, we have

$$P_R(uh) = P_{\Sigma_o \cup \Sigma_r}(s[1, \iota_s]) P_{\Sigma_o}(s[\iota_s + 1, |s|]) = P_{\Sigma_o}(uh) = \epsilon,$$

i.e., no event is observed upon the occurrence of uh.

Comparatively, for string $s = h$, we have $\iota_s = 1$ since string h reaches state $1 \in X_r$. Then we have

$$P_R(h) = P_{\Sigma_o \cup \Sigma_r}(s[1, \iota_s]) P_{\Sigma_o}(s[\iota_s + 1, |s|]) = P_{\Sigma_o \cup \Sigma_r}(h) = h.$$

This is because the occurrence of h is released when state $1 \in X_r$ is reached. Note that events in Σ_r that occur after reaching X_r will become unobservable again until a new state in X_r is reached again. For example, for string $s = hah$, we have still have $\iota_s = 1$, which gives

$$\begin{aligned} P_R(hah) =& P_{\Sigma_o \cup \Sigma_r}(s[1, \iota_s]) P_{\Sigma_o}(s[\iota_s + 1, |s|]) \\ =& P_{\Sigma_o \cup \Sigma_r}(s[1, 1]) P_{\Sigma_o}(s[2, 3]) \\ =& P_{\Sigma_o \cup \Sigma_r}(h) P_{\Sigma_o}(ah) \\ =& ha \end{aligned}$$

That is, only the first occurrence of h is released, while the second is not. □

Remark 3. According to the definition of the DIRM projection, once an event in Σ_r is released, the outside observer not only knows the occurrence of this event, but also knows *when* it occurred. This is different from the case of delayed observations, where events are "held on" physically, e.g., in communication channels, and the released events are observed following the existing observation. However, in our DIRM, the current observation can be overwritten after information release because we know when these previous events occurred.

3.3. Opacity under DIRM projection

In order to extend current-state opacity from the standard natural projection to our DIRM projection, one may think it suffices to replace P_{Σ_o} in Definition 1 by P_R. However, the following example shows that such a definition is not suitable for the case of dynamic information release mechanism.

Example 2. Again, let us still consider system G shown in Fig. 1. For string $s = haha$, there exists another string $s' = uhaha$ such that $P_R(s) = P_R(s') = haha$. If we simply replace P_{Σ_o} in Definition 1 by P_R, then we will assert that the system is current-state opaque because for string $s = haha$, which is the only string that leads to secret state 4, there exists another string $s' = uhaha$ that leads to a non-secret and they have the same projection.

However, this definition is not suitable here as the intruder can still assert that the system is currently at secret state 4 when string $s = haha$ is executed. To see this point more clearly, let us consider what the intruder can observe when string $s = haha$ is generated. Initially, it observes h when string h, which reaches $1 \in X_r$, is executed. Then it observes ha when a occurs. Finally, the intruder observes $haha$ when state $4 \in X_r$ is reached. On the other hand, for string $s' = uhaha$, the intruder will not observe h when uh is executed since this string leads to $6 \notin X_r$. Therefore, the intruder will first observe a when string uha is executed. When the second h is executed, which leads to state $8 \in X_r$, the intruder observes hah, i.e., the first occurrence of h was "held on" until this point. Therefore, the information histories available to the intruder are different when strings s and s' are executed. In other words, once the intruder observes h directly, it knows for sure that the system is currently at state 1. Therefore, if it further observes $haha$, it can still determine that the system is at secret state 4, which makes the system non-opaque. □

The above example reveals an important feature in the DIRM projection. That is, for any two strings $s, s' \in \mathcal{L}(G)$, $P_R(s) = P_R(s')$ does not necessarily imply that s and s' generate the same information-flow. This is because they may have different *intermediate observations* and hence, these two strings can still be distinguished. Note that such an issue does not arise in the standard natural projection as two strings having the same projection must have the same projection for their prefixes with the same length. However, it is not the case for our DIRM projection, which is the key technical issue we need to handle.

To correctly capture the information-flow of a string $s \in \mathcal{L}(G)$, we define

$$H_R(s) = \{P_R(s') : s' \in \overline{\{s\}}\}$$

as the *history* of string s. For the case of natural projection, history equivalence and projection equivalence are the same. However, in our setting, history equivalence is stronger than projection equivalence as it requires that the projections of all prefixes, i.e., all intermediate observations, are also equivalent.

Remark 4. Here, a history is defined as an unordered set rather than an order sequence according to the ordering of each observation. This definition is without loss of generality as the length of $P_R(s')$ is always non-decreasing when the length of the prefix s' increases. Therefore, one can always recover the ordering of the observation, if needed, based on the length of each element in the history set $H_R(s)$.

Therefore, to define current-state opacity, one can replace $P_{\Sigma_o}(\cdot)$ in Definition 1 by $H_R(\cdot)$. Here, we provide an equivalent definition using the notion of current-state estimate (CSE) for the sake of future developments. Formally, for any string $s \in \mathcal{L}(G)$, the current-state estimate upon history $H_R(s)$ is defined by

$$\hat{X}(H_R(s)) = \{\delta(s') \in X : \exists s' \in \mathcal{L}(G) \text{ s.t. } H_R(s) = H_R(s')\}.$$

Then we introduce current-state opacity under DIRM using the CSE as follows.

Definition 3. Given system G with secret states $X_S \subset X$ and DIRM $X_r \subseteq X$, we say system G is *current-state opaque* (w.r.t. X_S and DIRM X_r), if

$$\forall s \in \mathcal{L}(G) : \hat{X}(H_R(s)) \not\subseteq X_S.$$

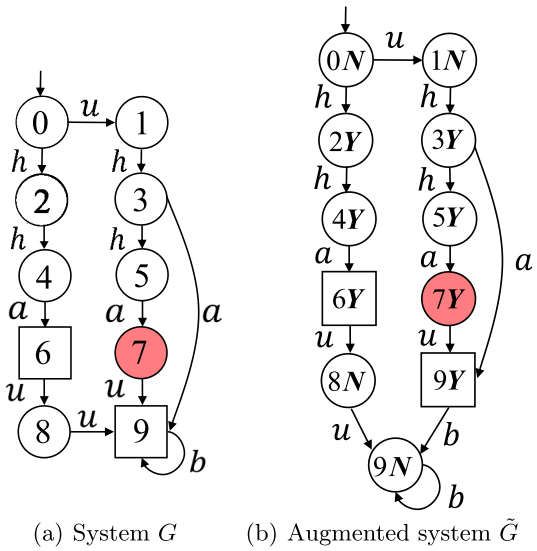
According to the definition of opacity, we implicitly assume that the intruder is aware of the DIRM of the underlying system. We illustrate this definition by the following example.

Example 3. Let us still consider system G in Fig. 1. Then for string $s = haha$ leading to secret state 4, we have $H_R(s) = \{\epsilon, h, ha, haha\}$. Note that s is the only string that generates history $H_R(s)$, e.g., for string $s' = uhaha$, we have $H_R(s') = \{\epsilon, a, hah, haha\} \neq H_R(s)$. Therefore, we have $\hat{X}(H_R(s)) = \{4\} \subseteq X_S$, which means that system G is not current-state opaque with respect to DIRM X_r and secret X_S.

Comparatively, suppose that $X_S = \{2\}$, then the system becomes opaque. This is because $t = ha$ is the only string that leads to secret state 2. However, we also have $t' = hah$ such that $H_R(t) = H_R(t') = \{\epsilon, h, ha\}$ and $\delta(t') = 3$. Therefore, we have $\hat{X}(H_R(t)) = \{2, 3\} \not\subseteq X_S$. □

Remark 5. Our DIRM projection is closely related to the so called *Orwellian projection* proposed in Mullins and Yeddes (2014), where a *downgrading event* is used to trigger information release. Our DIRM projection is different from the Orwellian projection in the following aspects. First, we consider partial information release while the Orwellian projection considers full information release. Specifically, under the Orwellian projection, once the information release mechanism is triggered, the outsider knows immediately the precise state of the system. This is not the case of our DIRM projection since we assume that events in Σ_{uo} are always unobservable, i.e., even when information release is triggered, the intruder may still have information uncertainty. Also, we consider a *state-based* information release mechanism rather than using a downgrading event to trigger the release as the case of Mullins and Yeddes (2014). We believe such a modeling is more natural in practice since, e.g., when to release information can be controlled by another transducer that does not need to be associated with events in the original plant model. Furthermore, the state-based mechanism also brings new technical challenges. In particular, due to the use of downgrading event, two strings having the same projection necessarily have the same history. However, for the general case of Orwellian-type observation, we need to consider histories rather than projections, which is the most significant technical challenge in our setting.

Note that, in the standard natural projection setting, the current-state estimate can be computed easily by constructing the standard *observer automaton* that essentially recursively updates the belief of the observer on-the-fly; see, e.g., Cassandras and Lafortune (2008). However, the computation of $\hat{X}(H_R(s))$, which is the key to the opacity verification problem, is much more challenging as delayed information is involved. In the following sections, we will elaborate on how to compute $\hat{X}(H_R(s))$.



(a) System G (b) Augmented system $\tilde{G}$

Fig. 2. System G and corresponding augmented system $\tilde{G}$, where $\Sigma_o = \{a, b\}$, $\Sigma_r = \{h\}$, and $\Sigma_{uo} = \{u\}$.

4. Augmenting the system with the trajectory information

4.1. Augmented systems

We note that, not every visit of an information-release-state in X_r will release some previous information; it depends on whether or not the trajectory visiting X_r contains events in Σ_r since the last visit of X_r, i.e., whether or not there are events in Σ_r that are "on hold". To capture the "true" information-release-states, we augment the original plant by an additional binary information $\{N, Y\}$ to obtain a new DFA

$$\tilde{G} = (\tilde{X}, \Sigma, \tilde{\delta}, \tilde{x}_0),$$

where

- $\tilde{X} \subseteq X \times \{N, Y\}$ is the set of augmented states;
- $\tilde{x}_0 = (x_0, N)$ is the initial augmented state;
- $\tilde{\delta} : \tilde{X} \times \Sigma \to \tilde{X}$ is the augmented transition function defined by: for any $\tilde{x} = (x, l) \in \tilde{X}$ and $\sigma \in \Sigma$ such that $\delta(x, \sigma)!$, we have

$$\tilde{\delta}(\tilde{x}, \sigma) = \begin{cases} (\delta(x, \sigma), Y) & \text{if } [\sigma \in \Sigma_r] \vee [x \notin X_r \wedge l = Y] \\ (\delta(x, \sigma), N) & \text{otherwise} \end{cases}$$

Intuitively, a state is augmented with labeled Y if it is either visited directly via an event in Σ_r or visited by a string that contains event Σ_r since the last visit of information-release-states X_r. Therefore, a state augmented with Y is a state at which some historical events can be released. When the current state x is in X_r and the upcoming event σ is not in Σ_r, label Y will be reset to N, which means that the previous information has been released (at state x) and there is no new information to be released in the future added. We illustrate the augmented system $\tilde{G}$ by the following example.

Example 4. Let us consider DFA $G = (X, \Sigma, \delta, x_0)$ shown in Fig. 2(a), where $\Sigma_o = \{a, b\}$, $\Sigma_r = \{h\}$, and $\Sigma_{uo} = \{u\}$. The DIRM is specified by information-release-states $X_r = \{6, 9\}$. Then the augmented system $\tilde{G}$ of G is depicted in Fig. 2(b). For the sake of simplicity, in the figure, we omit brackets and comma for each state in $\tilde{X}$, e.g., state $(1, Y)$ is simplified as $1Y$.

Specifically, the initial state is $0N$ and when event u occurs, we reach state $1N$. However, when r occurs, we reach state $2Y$

since $h \in \Sigma_r$. From state $6Y$, when event u occurs, we reach state $8N$ since $6 \in X_r$ and $u \notin \Sigma_r$. Note that state 9 in G is split as $9N$ and $9Y$ in $\tilde{G}$ depending on how it is visited. □

The augmented system $\tilde{G}$ contains at most $2 \cdot |X|$ states which is linear in the size of the original system. Furthermore, since for any state $(x, l) \in \tilde{X}$, $\tilde{\delta}((x, l), \sigma)$ is defined if and only if $\delta(x, \sigma)$ is defined, we also have

$$\mathcal{L}(\tilde{G}) = \mathcal{L}(G),$$

i.e., $\tilde{G}$ is essentially a state–space refinement of G. Hereafter, we will analyze opacity based on the augmented system $\tilde{G}$ instead of G. To this end, we define

$$\tilde{X}_S = \{(x, l) \in \tilde{X} : x \in X_S\}$$

as the set of secret states in $\tilde{G}$. Also, we define

$$\tilde{X}_r = \{(x, l) \in \tilde{X} : x \in X_r \text{ and } l = Y\}$$

as the set of information-release-states in $\tilde{G}$.

To distinguish with notations for G, before the end of this subsection, we first use $\tilde{P}_R(\cdot)$, $\tilde{H}_R(\cdot)$, $\hat{\tilde{X}}(\tilde{H}_R(s))$ to represent, respectively, as the DIRM projection, the history and the current-state estimate w.r.t. $\tilde{G}$ and $\tilde{X}_r$. The following result shows that considering the augmented system $\tilde{G}$ and the corresponding DIRM $\tilde{X}_r$ is the same as the original system.

Lemma 1. *For any string $s \in \mathcal{L}(G) = \mathcal{L}(\tilde{G})$, we have $H_R(s) = \tilde{H}_R(s)$.*

Proof. By definition, we have $H_R(s) = \{P_R(s') : s' \in \overline{\{s\}}\}$ and $\tilde{H}_R(s) = \{\tilde{P}_R(s') : s' \in \overline{\{s\}}\}$, where for any $s' \in \overline{\{s\}}$, we have $P_R(s') = P_{\Sigma_o \cup \Sigma_r}(s'[1, \iota_{s'}])P_{\Sigma_o}(s'[\iota_{s'} + 1, |s'|])$ and $\tilde{P}_R(s') = P_{\Sigma_o \cup \Sigma_r}(s'[1, \tilde{\iota}_{s'}])P_{\Sigma_o}(s'[\tilde{\iota}_{s'}+1, |s'|])$. Here we use $\iota_{s'}$ and $\tilde{\iota}_{s'}$ to denote the latest instant when s' reaches X_r in G and the latest instant when s' reaches $\tilde{X}_r$ in $\tilde{G}$, respectively. Since $\tilde{X}_r \subseteq X_r \times \{Y\}$, we have $\iota_{s'} \geq \tilde{\iota}_{s'}$.

To show that $H_R(s) = \tilde{H}_R(s)$, it suffices to show that $P_R(s') = \tilde{P}_R(s')$ for any $s' \in \overline{\{s\}}$. Note that, if $\tilde{\iota}_{s'} = \iota_{s'}$, we have immediately that $P_R(s') = \tilde{P}_R(s')$. Therefore, we consider the case of $\iota_{s'} > \tilde{\iota}_{s'}$ hereafter. For this case, we can write $P_R(s')$ and $\tilde{P}_R(s')$ in the forms of

$$P_R(s') = P_{\Sigma_o \cup \Sigma_r}(s'_1)P_{\Sigma_o \cup \Sigma_r}(s'_2)P_{\Sigma_o}(s'_3)$$
$$\tilde{P}_R(s') = P_{\Sigma_o \cup \Sigma_r}(s'_1)P_{\Sigma_o}(s'_2)P_{\Sigma_o}(s'_3),$$

where $s'_1 = s'[1, \tilde{\iota}_{s'}]$, $s'_2 = s'[\tilde{\iota}_{s'} + 1, \iota_{s'}]$ and $s'_3 = s'[\iota_{s'} + 1, |s'|]$. Therefore, it remains to show that $P_{\Sigma_o}(s'_2) = P_{\Sigma_o \cup \Sigma_r}(s'_2)$, i.e., s'_2 does not contain events in Σ_r.

Let $s'_2 = \sigma_1 \dots \sigma_m$ and let $x_i = \tilde{\delta}(s'_1\sigma_1 \dots \sigma_i)$ be the ith state visited from $\tilde{\delta}(s'_1) \in \tilde{X}_r$ in $\tilde{G}$. Assume for the sake of contradiction that s'_2 contains an event in Σ_r and let $k \in \{1, \dots, m\}$ be the largest number such that $\sigma_k \in \Sigma_r$. Furthermore, let $p \in \{k, \dots, m\}$ be the smallest instant such that $x_p \in X_r$. Note that x_p is well-defined since $x_m = \tilde{\delta}(s'_1s'_2) \in X_r$. Then, by the definition of $\tilde{G}$, we have $x_p \in X_r \times \{Y\} = \tilde{X}_r$. However, this violates the fact that $\tilde{\iota}_{s'}$ is the latest instant when s' reaches $\tilde{X}_r$ in $\tilde{G}$. Therefore, s'_2 does not contain any event in Σ_r, which completes the proof. □

Then the following theorem shows that to check opacity for G, it suffices to check opacity for $\tilde{G}$.

Theorem 1. *G is current-state opaque w.r.t. secret states X_S and DIRM X_r, if and only if, $\tilde{G}$ is current-state opaque w.r.t. secret states $\tilde{X}_S$ and DIRM $\tilde{X}_r$.*

Proof. For any string $s \in \mathcal{L}(G) = \mathcal{L}(\tilde{G})$, the current-state estimate of G by observing $H_R(s)$ is

$$\hat{X}(H_R(s)) = \{\delta(s') \in X : \exists s' \in \mathcal{L}(G) \text{ s.t. } H_R(s) = H_R(s')\}$$

and the current-state estimate of $\tilde{G}$ by observing $\tilde{H}_R(s)$ is

$$\hat{\tilde{X}}(\tilde{H}_R(s)) = \{\tilde{\delta}(s') \in \tilde{X} : \exists s' \in \mathcal{L}(\tilde{G}) \text{ s.t. } \tilde{H}_R(s) = \tilde{H}_R(s')\}.$$

By Lemma 1 and the construction of $\tilde{G}$, we have

$$\hat{\tilde{X}}(\tilde{H}_R(s)) \subseteq \hat{X}(H_R(s)) \times \{N, Y\}.$$

Therefore, by the definition of $\tilde{X}_S$, we have that

$$\hat{\tilde{X}}(\tilde{H}_R(s)) \subseteq \tilde{X}_S \Leftrightarrow \hat{X}(H_R(s)) \subseteq X_S.$$

Hence, $\tilde{G}$ is opaque if and only if G is opaque. □

Based on the above discussion, hereafter, we will only perform analysis based on the augmented system $\tilde{G}$. For the sake of simplicity, hereafter, we still use notations $\hat{X}(\cdot)$to represent the CSE under DIRM $\tilde{X}_r$ for system $\tilde{G}$.

4.2. Observations in $\tilde{G}$

In order to investigate how to estimate states in $\tilde{G}$, we first classify different types of observations. Under the DIRM, there are actually two different types of observations:

- the standard instant observation for an observable event: this observation is available when an observable event occurs;
- the release of some previous events in Σ_r: this observation is available when a state in $\tilde{X}_r$ is reached.

Note that the above two types of observations may occur simultaneously when $\tilde{X}_r$ is reached by an observable event $\sigma \in \Sigma_o$.

Therefore, events in Σ_r is unobservable before reaching a state in $\tilde{X}_r$. On the other hand, once events in Σ_r are released, we know precisely when they occurred. To handle this issue, we introduce two observation views: high-level view $\mathcal{O}(\tilde{x})$ and low-level view $\mathcal{O}_L(\tilde{x})$.

The high-level view assumes that events in $\Sigma_o \cup \Sigma_r$ are always observable. From this view, an event $\sigma \in \Sigma$ such that $\tilde{\delta}(\tilde{x}, \sigma)!$ is considered as an "observable" event at augmented state $\tilde{x} \in \tilde{X}$ if (i) $\sigma \in \Sigma_o \cup \Sigma_r$; or (ii) $\tilde{\delta}(\tilde{x}, \sigma) \in \tilde{X}_r$. Therefore, we define

$$\mathcal{O}(\tilde{x}) := \{\sigma \in \Sigma : \tilde{\delta}(\tilde{x}, \sigma)! \wedge [\sigma \in \Sigma_o \cup \Sigma_r \vee \tilde{\delta}(\tilde{x}, \sigma) \in \tilde{X}_r]\}$$

as the set of high-level "observable event" defined at state $\tilde{x} \in \tilde{X}$ assuming events in $\Sigma_o \cup \Sigma_r$ are observable.

On the other hand, the low-level view is from the observer/intruder's actual observation considering that we cannot see all events in $\mathcal{O}(\tilde{x})$ immediately since $\sigma \in \Sigma_r$ is not observable until it is released. Instead, we define

$$\mathcal{O}_L(\tilde{x}) := \{\sigma \in \Sigma : \tilde{\delta}(\tilde{x}, \sigma)! \wedge [\sigma \in \Sigma_o \vee \tilde{\delta}(\tilde{x}, \sigma) \in \tilde{X}_r]\}$$

as the set of low-level "observable events" defined at state $\tilde{x} \in \tilde{X}$ under the assumption that only event in Σ_o are instantly observable.

Let $q \in 2^{\tilde{X}}$ be a set of augmented states. In order to capture all observation-equivalent states of q, from the high-level view, we define $\widetilde{\mathrm{UR}}(q)$ as the *unobservable reach w.r.t.* $\mathcal{O}(\tilde{x})$. Unlike the standard unobservation reachable in static observation, this unobservation reach cannot be written in a closed-form since unobservable events are *state-dependent*, e.g., the transition is also observable when $\sigma \in \Sigma_{uo}$ and $\tilde{\delta}(\tilde{x}, \sigma) \in \tilde{X}_r$. Alternatively, we define $\widetilde{\mathrm{UR}}(q)$ inductively as follows:

- $q \subseteq \widetilde{\mathrm{UR}}(q)$;
- For any $\tilde{x} \in q, \sigma \in \Sigma$ such that $\tilde{\delta}(\tilde{x}, \sigma)!$, we have

  $\tilde{\delta}(\tilde{x}, \sigma) \in \widetilde{\mathrm{UR}}(q) \Leftrightarrow \sigma \notin \mathcal{O}(\tilde{x})$.

Also, from the observer's actual observation, we define $\widetilde{\mathrm{UR}}_L(q)$ as the *unobservable reach* with respective to the low-level view $\mathcal{O}_L(\tilde{x})$ by assuming one can only observe Σ_o instantly and events in Σ_r are subjected to the DIRM. Similarly, we define $\widetilde{\mathrm{UR}}_L(q)$ inductively as follows:

- $q \subseteq \widetilde{\mathrm{UR}}_L(q)$;
- For any $\tilde{x} \in q, \sigma \in \Sigma$ such that $\tilde{\delta}(\tilde{x}, \sigma)!$, we have

$$\tilde{\delta}(\tilde{x}, \sigma) \in \widetilde{\mathrm{UR}}_L(q) \Leftrightarrow \sigma \notin \mathcal{O}_L(\tilde{x}).$$

Next, we define how a state estimate $q \in 2^{\tilde{X}}$ evolves according to different types of observations. As we discussed above, the observer may observe an new event occurrence or an information release or both together. Therefore, if an observable event $\sigma \in \Sigma_o \cup \Sigma_r$ from the high-level view (or $\sigma \in \Sigma_o$ from the low-level view) occurs without information release, then the set of states reached immediately is defined by

$$\widetilde{\mathrm{NX}}_\sigma(q) = \{\tilde{x} \in \tilde{X} : \exists \tilde{x}' \in q \text{ s.t. } \tilde{x} = \tilde{\delta}(\tilde{x}', \sigma) \notin \tilde{X}_r\}.$$

If an unobservable event triggers an information release, then the set of states reached immediately is defined by

$$\widetilde{\mathrm{NX}}_r(q) = \{\tilde{x} \in \tilde{X} : \exists \tilde{x}' \in q, \sigma \in \Sigma_{uo} \text{ s.t. } \tilde{x} = \tilde{\delta}(\tilde{x}', \sigma) \in \tilde{X}_r\}.$$

If an observable event $\sigma \in \Sigma_o \cup \Sigma_r$ occurs and it also triggers an information release, then the set of states reached immediately is defined by

$$\widetilde{\mathrm{NX}}_{\sigma,r}(q) = \{\tilde{x} \in \tilde{X} : \exists \tilde{x}' \in q \text{ s.t. } \tilde{x} = \tilde{\delta}(\tilde{x}', \sigma) \in \tilde{X}_r\}.$$

We illustrate the above operators by the following example.

Example 5. Let us still consider $\tilde{G}$ shown in Fig. 2(b). For state set $\{0N\}$, from the high-level view, we have $\widetilde{\mathrm{UR}}(\{0N\}) = \{0N, 1N\}$ since only unobservable event u can be executed. However, from the low-level view, event h is "unobservable" until reaching a state in $\tilde{X}_r$. Therefore, we have $\widetilde{\mathrm{UR}}_L(\{0N\}) = \{0N, 1N, 2Y, 3Y, 4Y, 5Y\}$. Note that, within this unobservable reach, observable event a can happen from states $3Y$, $4Y$ and $5Y$; each yields a different observation. For example, from state $5Y$, one can just observe the occurrence of a without information release since the successor state reached, i.e., $7Y$, is not in $\tilde{X}_r$. Therefore, the corresponding observable reach is

$$\widetilde{\mathrm{NX}}_a(\{0N, 1N, 2Y, 3Y, 4Y, 5Y\}) = \{7Y\}$$

However, from states $3Y$ or $4Y$, the occurrence of a not only generates an instant observation but also releases some historical information. This corresponds to the following observable reach

$$\widetilde{\mathrm{NX}}_{a,r}(\{0N, 1N, 2Y, 3Y, 4Y, 5Y\}) = \{6Y, 9Y\}.$$

Note that, from the intruder's point of view, it can still distinguish between state $6Y$ and state $9Y$ since by reaching $6Y$ it will observe *hha* and by reaching $9Y$ it will observe *ha*. Therefore, only the unobservable and observable reach still cannot compute the precise state estimate of the system. We need additional information that tracks those unreleased information to correct this state estimate. This will be discussed in the next section. □

5. Verification of opacity using DIRM-observer

In this section, we discuss how to compute the current-state estimate, which is the key to the verification of current-state opacity, under the dynamic information release mechanism. To this end, we propose the DIRM-observer, which is a new DFA

$$Obs(\tilde{G}) = (X_{obs}, \Sigma, f, x_{obs,0}),$$

where

- $X_{obs} \subseteq \tilde{X} \times 2^{\tilde{X}} \times 2^{\tilde{X}}$ is the set states;
- $x_{obs,0} = (x_0, \widetilde{\mathrm{UR}}(\{x_0\}), \widetilde{\mathrm{UR}}_L(\{x_0\}))$ is the initial state;
- $f : X_{obs} \times \Sigma \to X_{obs}$ is the partial transition function defined by: for any $x_{obs} = (\tilde{x}, \hat{q}, q) \in X_{obs}$ and $\sigma \in \Sigma$, we have
  - $f(x_{obs}, \sigma)$ is defined if and only if $\tilde{\delta}(\tilde{x}, \sigma)$ is defined;
  - if $f(x_{obs}, \sigma)$ is defined, then we have $f((\tilde{x}, \hat{q}, q), \sigma) = (\tilde{x}', \hat{q}', q')$, where

$$\tilde{x}' = \tilde{\delta}(\tilde{x}, \sigma) \tag{2}$$

$$\hat{q}' = \begin{cases} \hat{q} & \text{if } \sigma \notin \mathcal{O}(\tilde{x}) \\ \widetilde{\mathrm{UR}}(\hat{q}_{mid}) & \text{if } \sigma \in \mathcal{O}(\tilde{x}) \end{cases} \tag{3}$$

$$q' = \begin{cases} q & \text{if } \sigma \notin \mathcal{O}_L(\tilde{x}) \\ \widetilde{\mathrm{UR}}_L(\widetilde{\mathrm{NX}}_\sigma(q)) & \text{if } \sigma \in \mathcal{O}_L(\tilde{x}) \wedge \tilde{x}' \notin \tilde{X}_r \\ \widetilde{\mathrm{UR}}_L(\hat{q}_{mid}) & \text{if } \sigma \in \mathcal{O}_L(\tilde{x}) \wedge \tilde{x}' \in \tilde{X}_r \end{cases} \tag{4}$$

where

$$\hat{q}_{mid} = \begin{cases} \widetilde{\mathrm{NX}}_\sigma(\hat{q}) & \text{if } \sigma \in \Sigma_o \cup \Sigma_r \wedge \tilde{x}' \notin \tilde{X}_r \\ \widetilde{\mathrm{NX}}_r(\hat{q}) & \text{if } \sigma \in \Sigma_{uo} \wedge \tilde{x}' \in \tilde{X}_r \\ \widetilde{\mathrm{NX}}_{\sigma,r}(\hat{q}) & \text{if } \sigma \in \Sigma_o \cup \Sigma_r \wedge \tilde{x}' \in \tilde{X}_r \end{cases} \tag{5}$$

For the sake of simplicity, we only consider the reachable part of $Obs(\tilde{G})$. Also, we note that each state in $Obs(\tilde{G})$ is in the form of $x_{obs} = (\tilde{x}, \hat{q}, q)$, where the first component is an augmented state while the second and the third components are sets of augmented states representing state estimates. Then for each state x_{obs}, we denote by $X_1(x_{obs})$, $X_2(x_{obs})$ and $X_3(x_{obs})$ its first, second and third components, respectively.

The intuition of each component of the DIRM-observer is explained as follows. The first component essentially tracks in *actual state* of the augmented system. Furthermore, we note that, at each state $x_{obs} = (\tilde{x}, \hat{q}, q)$, $f(x_{obs}, \sigma)$ is defined if and only if $\tilde{\delta}(\tilde{x}, \sigma)$ is defined. Therefore, we have $\mathcal{L}(Obs(\tilde{G})) = \mathcal{L}(\tilde{G}) = \mathcal{L}(G)$, i.e., $Obs(\tilde{G})$ exactly generates the same language of the original plant.

The second and third components estimate the state of the system from the high-level view and the low-level view, respectively. Specifically, the second component tracks all possible states the system could be in assuming the observer has the knowledge of the occurrences of events in Σ_r. Specifically, if $\sigma \notin \mathcal{O}(\tilde{x})$, i.e., there is no observation upon the occurrence of σ from the high-level view, then the state estimate should remain the same. If $\sigma \in \mathcal{O}(\tilde{x})$, then we need to first update the estimation upon the observation via observable reach and obtain an intermediate estimate denoted by $\hat{q}_{mid}$. Then we compute the unobservable reach of $\hat{q}_{mid}$ to complete the information update. As discussed earlier, there are three different types of observation for $\sigma \in \mathcal{O}(\tilde{x})$. Therefore, $\hat{q}_{mid}$ is computed accordingly for each case as follows:

- If $\sigma \in \Sigma_o \cup \Sigma_r$, but $\tilde{x}' \notin \tilde{X}_r$, i.e., from the high-level view of the second component, one can observe an instant observation in $\Sigma_o \cup \Sigma_r$ but there is no historical information released, then the state estimate is updated by observable reach $\widetilde{\mathrm{NX}}_\sigma(\hat{q})$ since it can track instant observation σ with no historical information released;
- If $\sigma \in \Sigma_{uo}$, but $\tilde{x}' \in \tilde{X}_r$, i.e., from the high-level view of the second component, there is no instant observation in $\Sigma_o \cup \Sigma_r$ but some historical information released, then the state estimate is updated first by $\widetilde{\mathrm{NX}}_r(\hat{q})$ since it tracks no instant observation and there is historical information released;

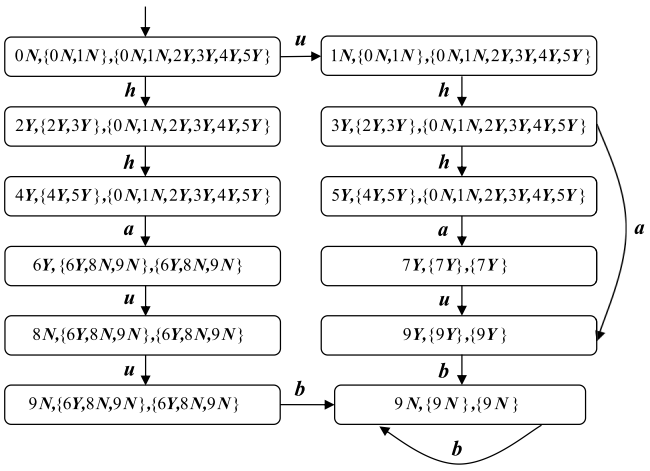


Fig. 3. DIRM-observer $Obs(\tilde{G})$ of G.

- If $\sigma \in \Sigma_o$, but $\tilde{x}' \in \tilde{X}_r$, i.e., from the high-level view of the second component, there are both instant observation in $\Sigma_o \cup \Sigma_r$ and some historical information released, then the state estimate is updated by observable reach $\widetilde{\mathrm{NX}}_{\sigma,r}(\hat{q})$.

After the above update using observable reach, we take the unobservable reach $\widetilde{\mathrm{UR}}(\hat{q}_{mid})$ w.r.t. $\mathcal{O}(\tilde{x})$ to capture all states that cannot be distinguished with $\hat{q}_{mid}$. Note that, the estimated information in the second component cannot be used directly because the actual observer does not directly observe events in Σ_r. The information in this component is used to construct the third component when information release is triggered. This is because, only in this case, all previous $\sigma \in \Sigma_r$ are released and one can then leverage the information in the high-level view to correct the information in the low-level view.

The third component aims to represent the current-state estimate under the DIRM projection, which is of our main interest. The information update of this component is based on the actual low-level view of the intruder. Specifically, if $\sigma \notin \mathcal{O}_L(\tilde{x})$, i.e., there is no actual observation of the intruder, then the state estimate should remain the same. On the other hand, for $\sigma \in \mathcal{O}_L(\tilde{x})$, depending on whether or not there is historical information released, we have the following two cases:

- If $\tilde{x}' \notin \tilde{X}_r$, then the intruder only has an instant observation σ but there is no historical information released. Therefore, the state estimate is updated by observable reach $\widetilde{\mathrm{NX}}_\sigma(q)$ and the low-level view unobservable reach $\widetilde{\mathrm{UR}}_L(q)$ w.r.t. $\mathcal{O}_L(\tilde{x})$;
- If $\tilde{x}' \in \tilde{X}_r$, then there is historical information released. As we explained for the second component, the intermediate estimate $\hat{q}_{mid}$ in the second component actually captures what the intruder knows if it has the high-level view. Since the information has been released, this information becomes available to the low-level intruder. Therefore, instead of taking the observable reach of the previous estimate in the third component, this part is "switched" to $\hat{q}_{mid}$ to utilize the released information. Next, since the third component captures the actual state estimate of the intruder, which cannot observe events in Σ_r before the next information release, we again take the low-level unobservable reach of $\hat{q}_{mid}$ from the intruder's actual point of view to complete the information update.

We use the following example to illustrate the DIRM-observer.

Example 6. Let us still consider DFA $G = (X, \Sigma, \delta, x_0)$ in Fig. 2(a) and its augmented system $\tilde{G}$ in Fig. 2(b). The DIRM-observer for $\tilde{G}$ is shown in Fig. 3. The initial state is $x_{obs,0} = (0N, \{0N, 1N\}, \{0N, 1N, 2Y, 3Y, 4Y, 5Y\})$ since $\widetilde{\mathrm{UR}}(\{0N\}) = \{0N, 1N\}$ and $\widetilde{\mathrm{UR}}_L(\{0N\}) = \{0N, 1N, 2Y, 3Y, 4Y, 5Y\}$. Therefore, by observing nothing, the state estimate of the intruder is $\hat{X}(\{\epsilon\}) = X_3(x_{obs,0}) = \{0N, 1N, 2Y, 3Y, 4Y, 5Y\}$.

For the initial-state, if event h occurs, we have $f(h) = (2Y, \{2Y, 3Y\}, \{0N, 1N, 2Y, 3Y, 4Y, 5Y\})$. Note that the first component is updated to $2Y = \tilde{\delta}(0N, h)$ and the second component is updated to $\widetilde{\mathrm{UR}}(\widetilde{\mathrm{NX}}_\sigma(\{0N, 1N\})) = \{2Y, 3Y\}$ since $h \in \Sigma_r$. However, the third component remains unchanged since $h \notin \mathcal{O}_L(0N) = \emptyset$. Similarly, when h occurs again, we have $f(hh) = (4Y, \{4Y, 5Y\}, \{0N, 1N, 2Y, 3Y, 4Y, 5Y\})$ since $h \notin \mathcal{O}_L(2Y) = \emptyset$, i.e., the third component is still not updated.

Now, let us consider string hha, the first component is updated to $6Y = \tilde{\delta}(4Y, a)$ and the second component is updated to $\widetilde{\mathrm{UR}}(\widetilde{\mathrm{NX}}_{a,r}(\{4Y, 5Y\})) = \{6Y, 8N, 9N\}$. Since $a \in \Sigma_o$ and $6Y \in \tilde{X}_r$, the third component is updated to $\widetilde{\mathrm{UR}}_L(\hat{q}_{mid}) = \{6Y, 8N, 9N\}$, where $\hat{q}_{mid} = \widetilde{\mathrm{NX}}_{a,r}(\{4Y, 5Y\}) = \{6Y\}$. This corresponds to the last case of Eq. (4).

For state $(5Y, \{4Y, 5Y\}, \{0N, 1N, 2Y, 3Y, 4Y, 5Y\})$, when event a occurs, it reaches $(7Y, \{7Y\}, \{7Y\})$, where since $a \in \Sigma_o$ but $7Y \notin \tilde{X}_r$, the third component is updated according to the second case of Eq. (4) by

$$\widetilde{\mathrm{UR}}_L(\widetilde{\mathrm{NX}}_a(\{0N, 1N, 2Y, 3Y, 4Y, 5Y\}))$$
$$= \widetilde{\mathrm{UR}}_L(\{7Y\}) = \{7Y\}$$

If event u occurs from $(7Y, \{7Y\}, \{7Y\})$, then we move to $(9Y, \{9Y\}, \{9Y\})$. Since $u \in \Sigma_{uo}$ but $9Y \in \tilde{X}_r$, the second component is updated according to the second case of Eq. (5) by $\widetilde{\mathrm{UR}}(\widetilde{\mathrm{NX}}_r(\{7Y\})) = \{9Y\}$; the third component is updated according to the third case of Eq. (4) by $\widetilde{\mathrm{UR}}_L(\hat{q}_{mid}) = \widetilde{\mathrm{UR}}_L(\{9Y\}) = \{9Y\}$, where $\hat{q}_{mid} = \widetilde{\mathrm{NX}}_r(\{7Y\}) = \{9Y\}$. □

Next, we show the properties of the DIRM-observer and establish its correctness. First, we show that, for a string s and any state in the component $X_2(f(s))$, there exists a string having the same history with string s. Its proof is provided in the appendix.

Lemma 2. *Let $Obs(\tilde{G})$ be the DIRM-observer of $\tilde{G}$. Then, for any $s \in \mathcal{L}(\tilde{G})$, we have*

$$\forall x \in X_2(f(s)), \exists\, t \in \mathcal{L}(\tilde{G}) : x = \tilde{\delta}(t) \wedge H_R(t) = H_R(s)$$

Using the above result, we can obtain a precise characterization for the second component of the DIRM-observer.

Proposition 1. *Let $Obs(\tilde{G})$ be the DIRM-observer of $\tilde{G}$. Then for any $s\sigma \in \mathcal{L}(\tilde{G})$ such that $\tilde{\delta}(s\sigma) \in \tilde{X}_r$, we have*

$$X_2(f(s\sigma)) = \left\{ \begin{array}{c} s' = t\sigma'\omega \in \mathcal{L}(\tilde{G}),\, \omega \in \Sigma_{uo}^* \wedge \\ \tilde{\delta}(s') : H_R(t) = H_R(s) \wedge \tilde{\delta}(t\sigma') \in \tilde{X}_r \wedge \\ P_{\Sigma_o \cup \Sigma_r}(t\sigma') = P_{\Sigma_o \cup \Sigma_r}(s\sigma) \end{array} \right\}$$

Also, we characterize the third component of the DIRM-observer by showing that indeed tracks the current-state estimate of system under DIRM projection.

Proposition 2. *Let $Obs(\tilde{G})$ be the DIRM-observer of $\tilde{G}$. Then, we have*

$$\forall s \in \mathcal{L}(\tilde{G}) : X_3(f(s)) = \hat{X}(H_R(s)) \tag{6}$$

By putting the above results together, we finally provide a theorem to show how to verify current-state opacity under DIRM based on the DIRM-observer.

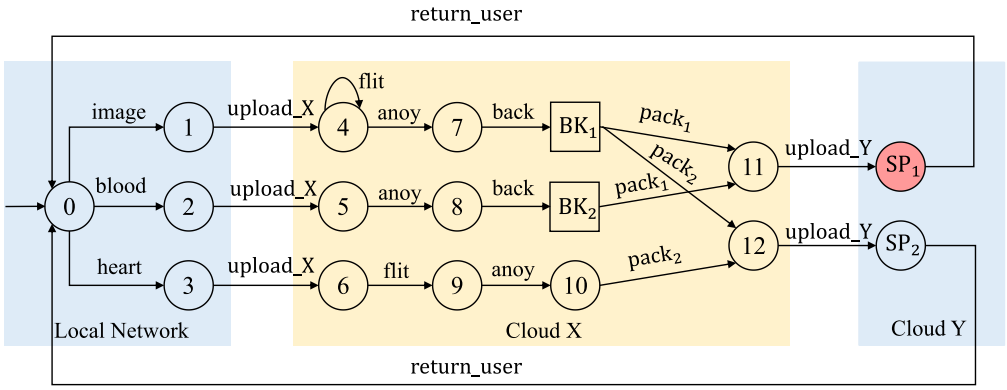


Fig. 4. The operation flow in a cloud based medical service system.

Theorem 2. *Let $Obs(\tilde{G})$ be the DIRM-observer for system G. Then system G is current-state opaque w.r.t. X_S and X_r if and only if*

$$\forall x_{obs} \in X_{obs} : X_3(x_{obs}) \not\subseteq \tilde{X}_S \tag{7}$$

Proof. (if) Suppose that $\forall x_{obs} \in X_{obs} : X_3(x_{obs}) \not\subseteq \tilde{X}_S$. By Proposition 2 and $\mathcal{L}(\tilde{G}) = \mathcal{L}(Obs(\tilde{G}))$, we have that $\forall s \in \mathcal{L}(\tilde{G}) : X_3(f(s)) = \hat{X}(H_R(s)) \not\subseteq \tilde{X}_S$. Then by Definition 3, we have that $\tilde{G}$ is current-state opaque w.r.t. $\tilde{X}_S$ and $\tilde{X}_r$. Since $\mathcal{L}(G) = \mathcal{L}(\tilde{G})$, the definition of $\tilde{X}_S$ and $\tilde{X}_r$, and Theorem 1, we have that G is current-state opaque w.r.t. X_S and X_r.

(only if) By contradiction. Suppose that, for G that is current-state opaque w.r.t. secret states X_S and DIRM X_r and the corresponding $\tilde{G}$, $\exists x_{obs} \in X_{obs} : X_3(x_{obs}) \subseteq \tilde{X}_S$. Based on Proposition 2 and the fact that $\mathcal{L}(G) = \mathcal{L}(\tilde{G}) = \mathcal{L}(Obs(\tilde{G}))$, we have $\exists s \in \mathcal{L}(\tilde{G}) : X_3(f(s)) = \hat{X}(H_R(s)) \subseteq \tilde{X}_S$. By Definition 3 and Theorem 1, it can conclude that $\exists s \in \mathcal{L}(G) : \hat{X}(H_R(s)) \subseteq X_S$, which contradicts with the assumption. □

The theorem shows that current-state opacity can be checked by a reachability search in the DIRM-observer. We illustrate Theorem 2 by the following example.

Example 7. Let us still consider DFA $G = (X, \Sigma, \delta, x_0)$ in Fig. 2(a) and its augmented system $\tilde{G}$ in Fig. 2(b). The DIRM-observer for $\tilde{G}$ has been shown in Fig. 3, where for $x = (7Y, \{7Y\}, \{7Y\})$, we have $X_3(x) = \{7Y\} \subseteq \tilde{X}_S$. This state is reached by *uhha* with $H_R(uhha) = \{\epsilon, a\}$, which is the unique string having this history. Therefore, the intruder knows for sure that the system is at secret state 7 and also, by Theorem 2, we know that G is not current-state opaque with respect to X_r and X_S. □

Remark 6. Finally, we discuss the complexity of proposed approach for verifying current-state opacity under DIRM. First, the augmented system $\tilde{G}$ contains at most $2 \cdot |X|$ states which is linear in the size of the original system. The DIRM-observer contains at most $|\tilde{X}| \cdot 2^{|\tilde{X}|} \cdot 2^{|\tilde{X}|}$ states and $|\Sigma| \cdot |\tilde{X}| \cdot 2^{|\tilde{X}|} \cdot 2^{|\tilde{X}|}$ transitions. Therefore, the overall complexity for verifying current-state opacity is exponential in the number of states in the original system. However, this complexity seems to be unavoidable since verifying current-state opacity under the standard natural projection, which is a special case of the DIRM, is already shown to be PSPACE-hard.

6. Case study on medical cloud computing services

In this section, we apply the proposed framework to a cloud-computing-based medical data processing example. This example is adopted from Zeng and Koutny (2019) with some modifications and simplifications. The reader is referred to Zeng and Koutny (2019) for details on how to model medical cloud computing systems using DES models.

Specifically, we consider a scenario, where a user wants to analyze some medical data of patients by cloud computing services. The entire process involves a local network and two clouds X and Y as shown in Fig. 4; each of them works as follows:

- Initially, the user in the local network extracts different health data for later processing. Here, we consider three data extraction operations: `image` representing the CT image data of the patient, `blood` representing the blood text data of the patient and `heart` representing the heart rate sequence data of the patient. Then the extracted data are uploaded to Cloud X for further pre-processing represented by operation `upload_X`.
- Cloud X aims to pre-process data received from the local network. Depending on different types of patient data, different pre-processing services are provided, including data filtering operation `filt` and data anonymization operation `anoy`. As shown in Fig. 4, image data may need to be filtered for multiple times, while blood text data does not need to be filtered. The pre-processed data will be packaged to the sender nodes, where the data are sent to service providers SP_1 or SP_2 in Cloud Y for final process under operation `upload_Y`. We assume that the image data can be processed by either SP_1 or SP_2, while the blood data and the heart rate data can only be processed by SP_1 and SP_2, respectively. Furthermore, before packaging the data, the CT image data and the blood text data also need to be backuped represented by operation `back`.
- Cloud Y further analyzes the data pre-processed by Cloud X either by service providers SP_1 or by SP_2 in it. Then the final results will be send back to the user in the local network by operation `return_user`.

Although clouding computing provides a powerful way for processing big local data, one of its main concerns is the privacy issue. Here we consider the following information release scenario. We assume that the communication between the local network, Cloud X and Cloud Y are available to the outsider, i.e., we have

$$\Sigma_o = \{\texttt{upload\_X}, \texttt{upload\_Y}, \texttt{return\_user}\}$$

We assume that the operations in within each cloud and the local network are not available directly to the outsider. However, Cloud X is subject to the so called *log attacks* (Chu et al., 2013; Sanamrad & Kossmann, 2013) at the backup locations BK_i, $i = 1, 2$. Specifically, the attacker is able to recover operations that has been executed in the cloud with the right order by checking its logs. Therefore, pre-processing operations `filt` and `anoy`

are events that are initial unobservable but will be released at locations $\mathtt{BK}_i$, $i = 1, 2$. Formally, we have

$\Sigma_{uo} = \{\mathtt{image}, \mathtt{blood}, \mathtt{heart}, \mathtt{pack}_1, \mathtt{pack}_2\}$

and releasable events and the release states are

$\Sigma_r = \{\mathtt{filt}, \mathtt{anoy}, \mathtt{back}\}$ and $X_r = \{\mathtt{BK}_1, \mathtt{BK}_2\}$.

Here we consider a privacy constraint requiring that the intruder should never be able to discover that the user is utilizing service provider $\mathtt{SP}_1$ in Cloud Y. This is because $\mathtt{SP}_1$ can provide more detailed analysis than $\mathtt{SP}_2$, and therefore, it is used when the pre-processing procedures in Cloud X find that the patients have high risk of diseases. Here, we can convert this privacy requirement as a current-state opacity problem by considering $X_S = \{\mathtt{SP}_1\}$. To verify current-state opacity, one can construct the DIRM-observer, which is omitted here, and we can conclude easily that the entire system is current-state opacity under DIRM. For example, for execution

$s = \mathtt{blood} \cdot \mathtt{upload\_X} \cdot \mathtt{anoy} \cdot \mathtt{back} \cdot \mathtt{pack}_1 \cdot \mathtt{upload\_Y}$

that utilizes service provider $\mathtt{SP}_1$, there exists another execution

$t = \mathtt{image} \cdot \mathtt{upload\_X} \cdot \mathtt{anoy} \cdot \mathtt{back} \cdot \mathtt{pack}_2 \cdot \mathtt{upload\_Y}$

that utilizes service provider $\mathtt{SP}_2$, which does not indicate sensitive information of patients.

Note that this example cannot be modeled using the standard natural projection. This is because whether or not the executions of operations `filt` and `anoy` will be discovered by the intruder depends on whether the system will visit release state $\mathtt{BK}_i$ in the future. Therefore, the proposed DIRM is used to capture the information-flow in this log-attack scenario.

7. Conclusion

In this paper, we proposed a new Orwellian-type observation model called the dynamic information release mechanism that allows to release history information generated dynamically based on the current state. A new history-based definition of current-state opacity was proposed that better captures the feature of non-prefix-closed observations, which is the main feature of Orwellian-type observations. Then we proposed a new information structure called the DIRM-observer that effectively fuses the released previous information to update the current-state estimate. Based on the DIRM-observer, we showed that the current-state opacity verification problem can be effectively solved. An illustrative example on medical cloud computing is provided to illustrate our framework.

There are many interesting future directions under the proposed DIRM framework. First, it is interesting to investigate other types of opacity, e.g., initial-state opacity or infinite/K-step opacity, under the DIRM. Furthermore, in this work, it is assumed that the information release policy is already given (encoded as a state-based function) and we want to verify opacity under a given release policy. It is also very interesting to investigate how to synthesize an information release policy that releases historical information as frequently as possible but still preserves opacity.

Appendix A. Proofs not contained in main body

Proof of Lemma 2. We prove it by induction on the length of s.

Induction Basis: Suppose that $|s| = 0$, i.e., $s = \epsilon$ and $H_R(\epsilon) = \{\epsilon\}$. Then we easily know that $X_2(f(\epsilon)) = \widetilde{\mathrm{UR}}(\{\tilde{x}_0\}) = H_R(\epsilon)$. Therefore, the induction basis holds.

Induction Step: Now, let us assume that, for string $s \in \mathcal{L}(\tilde{G})$ with $|s| = k$, we have $\forall x \in X_2(f(s)), \exists\, t \in \mathcal{L}(\tilde{G}) : x = \tilde{\delta}(t) \wedge H_R(t) = H_R(s)$. Then we consider $s\sigma \in \mathcal{L}(\tilde{G})$, $\sigma \in \Sigma$.

For any $x \in X_2(f(s\sigma))$, by the definition of $X_2(f(s\sigma))$, we have $\tilde{\delta}(t) \in X_2(f(s))$, then there exist $t' = t\sigma'\omega \in \mathcal{L}(\tilde{G})$ and $\omega \in \Sigma_{uo}^*$ such that $x = \tilde{\delta}(t')$. As the construction of X_2, we know the events in $\Sigma_o \cup \Sigma_r$ are observable. Hence, by the definition of the specified observable reach of $X_2(f(s\sigma))$ and the induction hypothesis $H_R(t) = H_R(s)$, we have

$$P_{\Sigma_o \cup \Sigma_r}(t') = P_{\Sigma_o \cup \Sigma_r}(s\sigma) \tag{A.1}$$

$$P_R(t) = P_R(s) \tag{A.2}$$

Next, we consider different cases.

Case 1: $\sigma \in \Sigma_o \wedge \tilde{x}' \in \tilde{X}_r$

For this case, $X_2(f(s\sigma)) = \widetilde{\mathrm{UR}}(\widetilde{\mathrm{NX}}_{\sigma,r}(X_2(f(s))))$, then we have $\sigma = \sigma' \in \Sigma_o$ and $\tilde{\delta}(t\sigma') \in \tilde{X}_r$. Furthermore, the above makes that

$$P_R(s\sigma) = P_{\Sigma_o \cup \Sigma_r}(s\sigma) \wedge P_R(t') = P_R(t\sigma') = P_{\Sigma_o \cup \Sigma_r}(t\sigma') \tag{A.3}$$

By the induction hypothesis, Eqs. (A.1) and (A.3), we have $H_R(s) \cup P_R(s\sigma) = H_R(t) \cup P_R(t')$, i.e., $H_R(s\sigma) = H_R(t')$. Therefore, for $\sigma \in \Sigma_o \wedge \tilde{x}' \in \tilde{X}_r$, we have $\forall x \in X_2(f(s\sigma)), \exists\, t' \in \mathcal{L}(\tilde{G}) : x = \tilde{\delta}(t') \wedge H_R(t') = H_R(s\sigma)$, i.e., the hypothesis holds in this case.

Case 2: $\sigma \in \Sigma_o \cup \Sigma_r \wedge \tilde{x}' \notin \tilde{X}_r$

For this case, $X_2(f(s\sigma)) = \widetilde{\mathrm{UR}}(\widetilde{\mathrm{NX}}_{\sigma}(X_2(f(s))))$, then we have $\sigma = \sigma'$ and $\tilde{\delta}(t\sigma') \notin \tilde{X}_r$. Furthermore, the above makes that,

when $\sigma \in \Sigma_r$,

$$P_R(s\sigma) = P_R(s) \wedge P_R(t') = P_R(t) \tag{A.4}$$

when $\sigma \in \Sigma_o$,

$$P_R(s\sigma) = P_R(s)\sigma \wedge P_R(t') = P_R(t\sigma') = P_R(t)\sigma' \tag{A.5}$$

By the induction hypothesis, Eqs. (A.2), (A.4) or (A.5), we have $H_R(s) \cup P_R(s\sigma) = H_R(t) \cup P_R(t')$, i.e., $H_R(s\sigma) = H_R(t')$. Therefore, the hypothesis holds in this case.

Case 3: $\sigma \in \Sigma_{uo} \wedge \tilde{x}' \in \tilde{X}_r$

For this case, $X_2(f(s\sigma)) = \widetilde{\mathrm{UR}}(\widetilde{\mathrm{NX}}_r(X_2(f(s))))$, then we have $\sigma, \sigma' \in \Sigma_{uo}$ and $\tilde{\delta}(t\sigma') \in \tilde{X}_r$. Furthermore, the above makes that

$$P_R(s\sigma) = P_{\Sigma_o \cup \Sigma_r}(s\sigma) \wedge P_R(t') = P_R(t\sigma') = P_{\Sigma_o \cup \Sigma_r}(t\sigma') \tag{A.6}$$

By the induction hypothesis, Eqs. (A.1) and (A.6), we have $H_R(s) \cup P_R(s\sigma) = H_R(t) \cup P_R(t')$, i.e., $H_R(s\sigma) = H_R(t')$. Therefore, the hypothesis holds in this case.

Case 4: $\sigma \in \Sigma_{uo} \wedge \tilde{x}' \notin \tilde{X}_r$

For this case, we have $X_2(f(s\sigma)) = X_2(f(s))$, $t' = t\omega \in \mathcal{L}(\tilde{G})$ and $\omega \in \Sigma_{uo}^*$. Further, we easily have that, $H_R(s\sigma) = H_R(s) = H_R(t) = H_R(t')$. Therefore, the hypothesis holds in this case.

Based the above cases, this proposition holds. □

Proof of Proposition 1. ($\subseteq$) According to Lemma 2 and $\tilde{\delta}(s\sigma) \in \tilde{X}_r$, we have that $\forall x \in X_2(f(s\sigma)), \exists\, t\sigma' \in \mathcal{L}(\tilde{G}) : x = \tilde{\delta}(t\sigma') \wedge H_R(t\sigma') = H_R(s\sigma)$ and $P_{\Sigma_o \cup \Sigma_r}(t\sigma') = P_{\Sigma_o \cup \Sigma_r}(s\sigma)$. Also, for $t\sigma'$, we have $t'\omega = t\sigma'$, $\omega \in \Sigma_{uo}^*$ such that $\tilde{\delta}(t') \in \tilde{X}_r$. Therefore, the right set includes the left set.

($\supseteq$) By contradiction. Suppose that there exists string $t\sigma' \in \mathcal{L}(\tilde{G})$ which satisfies the right conditions, but $\tilde{\delta}(t\sigma') \notin X_2(f(s\sigma))$. Since the specified observable reach of the definition of components X_2 in $Obs(\tilde{G})$ and the right conditions, for t, there only can be $\tilde{\delta}(t) \notin X_2(f(s))$ and $P_{\Sigma_o \cup \Sigma_r}(t) = P_{\Sigma_o \cup \Sigma_r}(s)$. Similarly, we inductively deduce to that there must be $t' \in \Sigma_{uo}^* \cap \overline{\{t\}}$ such that $\tilde{\delta}(t') \notin X_2(f(\epsilon))$. It contradicts with the initial states definition of X_2, since $\tilde{\delta}(t')$ must be in $\widetilde{\mathrm{UR}}(\{x_0\})$, but $X_2(f(\epsilon)) = \widetilde{\mathrm{UR}}(\{x_0\})$. Therefore, any state in the right also includes in the left. □

Proof of Proposition 2. We prove it by induction on the length of s.

Induction Basis: Suppose that $|s| = 0$, i.e., $s = \epsilon$ and $H_R(\epsilon) = \{\epsilon\}$. Then we know that

$$\begin{aligned}&\hat{X}(\{\epsilon\}) \qquad\qquad (A.7)\\ &= \left\{\tilde{\delta}(s') \in \tilde{X} : s' \in \mathcal{L}(\tilde{G}) \wedge H_R(s') = \{\epsilon\}\right\}\\ &= \left\{\tilde{\delta}(s') \in \tilde{X} : s' \in \mathcal{L}(\tilde{G}) \wedge (\forall s'' \in \overline{\{s'\}})[P_R(s'') = \epsilon]\right\}\\ &= \left\{\tilde{\delta}(s') \in \tilde{X} : \begin{array}{c} s' = \sigma_1 \ldots \sigma_n \in \mathcal{L}(\tilde{G}) \wedge \\ (\forall i \leq n)[\sigma_i \notin \mathcal{O}_L(\tilde{\delta}(\sigma_1 \ldots \sigma_{i-1}))] \end{array}\right\}\end{aligned}$$

Since $\tilde{x}_0$ is included in both $\hat{X}(\{\epsilon\})$ and $\widetilde{\mathrm{UR}}_L(\{\tilde{x}_0\})$, by a simple inductive argument according to Eq. (A.7) and the definition of $\widetilde{\mathrm{UR}}_L$, one can easily conclude that $\hat{X}(\{\epsilon\}) = \widetilde{\mathrm{UR}}_L(\{\tilde{x}_0\})$, i.e., the induction basis holds.

Induction Step: Now, let us assume that, for string $s \in \mathcal{L}(\tilde{G})$ with $|s| = k$, we have $X_3(f(s)) = \hat{X}(H_R(s))$. Then we consider $s\sigma \in \mathcal{L}(\tilde{G})$, where $\sigma \in \Sigma$, for the following two cases.

Case 1: $\tilde{\delta}(s\sigma) \notin \tilde{X}_r$.

For this case, we have $H_R(s\sigma) = H_R(s) \cup \{P_R(s\sigma)\}$. If $\sigma \in \Sigma_o$, then

$$\begin{aligned}&\hat{X}(H_R(s\sigma))\\ &= \left\{\tilde{\delta}(s') : s' \in \mathcal{L}(\tilde{G}) \wedge H_R(s') = H_R(s) \cup \{P_R(s)\sigma\}\right\}\\ &= \left\{\tilde{\delta}(s') : \begin{array}{c} s' = t\sigma\sigma_1 \ldots \sigma_n \in \mathcal{L}(\tilde{G}) \wedge \\ H_R(t) = H_R(s) \wedge \tilde{\delta}(t\sigma) \notin \tilde{X}_r \wedge \\ (\forall i \leq n)[\sigma_i \notin \mathcal{O}_L(\tilde{\delta}(t\sigma\sigma_1 \ldots \sigma_{i-1}))] \end{array}\right\}\\ &= \left\{\tilde{\delta}(x, \sigma w) : \begin{array}{c} w = \sigma_1 \ldots \sigma_n \wedge \\ x \in \hat{X}(H_R(s)) \wedge \tilde{\delta}(x, \sigma) \notin \tilde{X}_r \wedge \\ \forall i \leq n : \sigma_i \notin \mathcal{O}_L(\tilde{\delta}(x, \sigma\sigma_1 \ldots \sigma_{i-1})) \end{array}\right\}\\ &= \left\{\tilde{\delta}(x', w) : \begin{array}{c} w = \sigma_1 \ldots \sigma_n \wedge \\ x' \in \widetilde{\mathrm{NX}}_\sigma(\hat{X}(H_R(s))) \wedge \\ \forall i \leq n : \sigma_i \notin \mathcal{O}_L(\tilde{\delta}(x', \sigma_1 \ldots \sigma_{i-1})) \end{array}\right\}\\ &= \widetilde{\mathrm{UR}}_L(\widetilde{\mathrm{NX}}_\sigma(\hat{X}(H_R(s))))\end{aligned}$$

Still, by the induction hypothesis and the update function for the third component in Eq. (4), we have

$$\begin{aligned}X_3(f(s\sigma)) &= \widetilde{\mathrm{UR}}_L(\widetilde{\mathrm{NX}}_\sigma(X_3(f(s)))) \qquad (A.8)\\ &= \widetilde{\mathrm{UR}}_L(\widetilde{\mathrm{NX}}_\sigma(\hat{X}(H_R(s)))) = \hat{X}(H_R(s\sigma))\end{aligned}$$

If $\sigma \notin \Sigma_o$, then we have $H_R(s\sigma) = H_R(s)$, which means that $\hat{X}(H_R(s\sigma)) = \hat{X}(H_R(s))$. Therefore, we also have

$$X_3(f(s\sigma)) = X_3(f(s)) = \hat{X}(H_R(s)) = \hat{X}(H_R(s\sigma)) \qquad (A.9)$$

Case 2: $\tilde{\delta}(s\sigma) \in \tilde{X}_r$.

For this case, we have $H_R(s\sigma) = H_R(s) \cup \{P_{\Sigma_o \cup \Sigma_r}(s\sigma)\}$. Therefore, we have

$$\begin{aligned}&\hat{X}(H_R(s\sigma))\\ &= \left\{\tilde{\delta}(s') : s' \in \mathcal{L}(\tilde{G}) \wedge H_R(s') = H_R(s) \cup \{P_{\Sigma_o \cup \Sigma_r}(s\sigma)\}\right\}\\ &= \left\{\tilde{\delta}(s') : \begin{array}{c} s' = t\sigma'\sigma_1 \ldots \sigma_n \in \mathcal{L}(\tilde{G}) \wedge \\ H_R(t) = H_R(s) \wedge \tilde{\delta}(t\sigma') \in \tilde{X}_r \wedge \\ P_{\Sigma_o \cup \Sigma_r}(t\sigma') = P_{\Sigma_o \cup \Sigma_r}(s\sigma) \wedge \\ \forall i \leq n : \sigma_i \notin \mathcal{O}_L(\tilde{\delta}(t\sigma'\sigma_1 \ldots \sigma_{i-1})) \end{array}\right\}\end{aligned}$$

Since $\widetilde{\mathrm{UR}}(q) \subseteq \widetilde{\mathrm{UR}}_L(q)$, then by Proposition 1, we further have

$$\hat{X}(H_R(s\sigma)) = \widetilde{\mathrm{UR}}_L(X_2(f(s\sigma))) = \widetilde{\mathrm{UR}}_L(\hat{q}_{mid})$$

Then, by the induction hypothesis and the update function for the third component in Eq. (4), we have

$$\hat{X}(H_R(s\sigma)) = X_3(f(s\sigma))$$

This completes the proof since $X_3(f(s)) = \hat{X}(H_R(s))$ holds for each case. □

References

An, L., & Yang, G.-H. (2020). Opacity enforcement for confidential robust control in linear cyber-physical systems. *IEEE Transactions on Automatic Control*, *65*(3), 1234–1241.

Balun, J., & Masopust, T. (2019). On opacity verification for discrete-event systems. arXiv:1912.07314.

Barcelos, R. J., & Basilio, J. C. (2018). Enforcing current-state opacity through shuffle in event observations. *IFAC-PapersOnLine*, *51*(7), 100–105.

Behinaein, B., Lin, F., & Rudie, K. (2019). Optimal information release for mixed opacity in discrete-event systems. *IEEE Transactions on Automation Science and Engineering*, *16*(4), 1960–1970.

Ben Hadj-Alouane, N., Lafrance, S., Lin, F., Mullins, J., & Yeddes, M. (2005a). Characterizing intransitive noninterference for 3-domain security policies with observability. *IEEE Transactions on AAutomatic Control*, *50*(6), 920–925.

Ben Hadj-Alouane, N., Lafrance, S., Lin, F., Mullins, J., & Yeddes, M. (2005b). On the verification of intransitive noninterference in mulitlevel security. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *35*(5), 948–958.

Bérard, B., & Mullins, J. (2014). Verification of information flow properties under rational observation. arXiv:1409.0871.

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

Bryans, J. W., Koutny, M., & Ryan, P. (2005). Modelling opacity using Petri nets. *Electronic Notes in Theoretical Computer Science*, *121*, 101–115.

Cassandras, C. G., & Lafortune, S. (2008). *Introduction to discrete event systems* (2nd ed.). Springer.

Cassez, F., Dubreil, J., & Marchand, H. (2012). Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, *40*(1), 88–115.

Chen, J., Ibrahim, M., & Kumar, R. (2017). Quantification of secrecy in partially observed stochastic discrete event systems. *IEEE Transactions on Automation Science and Engineering*, *14*(1), 185–195.

Chu, C., Zhu, W., Han, J., Liu, J. K., Xu, J., & Zhou, J. (2013). Security concerns in popular cloud storage services. *IEEE Pervasive Computing*, *12*(4), 50–57. http://dx.doi.org/10.1109/MPRV.2013.72.

Cong, X., Fanti, M. P., Mangini, A. M., & Li, Z. (2019). On-line verification of initial-state opacity by Petri nets and integer linear programming. *ISA Transactions*, *93*, 108–114.

Dubreil, J., Darondeau, P., & Marchand, H. (2010). Supervisory control for opacity. *IEEE Transactions on Automatic Control*, *55*(5), 1089–1100.

Falcone, Y., & Marchand, H. (2015). Enforcement and validation (at runtime) of various notions of opacity. *Discrete Event Dynamic Systems*, *25*(4), 531–570.

Hadjicostis, C. N. (2020). *Estimation and inference in discrete event systems*. Springer.

Jacob, R., Lesage, J.-J., & Faure, J.-M. (2016). Overview of discrete event systems opacity: Models, validation, and quantification. *Annual Review of Control*, *41*, 135–146.

Ji, Y., Yin, X., & Lafortune, S. (2019). Enforcing opacity by insertion functions under multiple energy constraints. *Automatica*, *108*, Article 108476.

Keroglou, C., & Hadjicostis, C. N. (2017). Probabilistic system opacity in discrete event systems. *Discrete Event Dynamic Systems*, 1–26.

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

Lin, F. (2011). Opacity of discrete event systems and its applications. *Automatica*, *47*(3), 496–503.

Lin, F., Chen, W., Wang, W., & Wang, F. (2020). Information control in networked discrete event systems and its application to battery management systems. *Discrete Event Dynamic Systems*, 1–26.

Liu, R., Mei, L., & Lu, J. (2020). *K*-memory-embedded insertion mechanism for opacity enforcement. *Systems & Control Letters*, *145*, Article 104785.

Liu, S., & Zamani, M. (2020). Verification of approximate opacity via barrier certificates. *IEEE Control Systems Letters*.

Mohajerani, S., Ji, Y., & Lafortune, S. (2020). Compositional and abstraction-based approach for synthesis of edit functions for opacity enforcement. *IEEE Transactions on Automatic Control*, *65*(8), 3349–3364.

Mohajerani, S., & Lafortune, S. (2020). Transforming opacity verification to nonblocking verification in modular systems. *IEEE Transactions on Automatic Control*, *65*(4), 1739–1746.

Mullins, J., & Yeddes, M. (2014). Opacity with orwellian observers and intransitive non-interference. In *12th int. workshop on discrete event systems* (pp. 344–349).

Ramasubramanian, B., Cleaveland, W. R., & Marcus, S. (2020). Notions of centralized and decentralized opacity in linear systems. *IEEE Transactions on Automatic Control*, *265*(4), 1442–1455.
Saadaoui, I., Li, Z., & Wu, N. (2020). Current-state opacity modelling and verification in partially observed Petri nets. *Automatica*, *116*, Article 108907.
Saboori, A., & Hadjicostis, C. N. (2011). Verification of *K*-step opacity and analysis of its complexity. *IEEE Transactions on Automation Science and Engineering*, *8*(3), 549–559.
Saboori, A., & Hadjicostis, C. N. (2012). Verification of infinite-step opacity and complexity considerations. *IEEE Transactions on Automatic Control*, *57*(5), 1265–1269.
Saboori, A., & Hadjicostis, C. N. (2013). Verification of initial-state opacity in security applications of discrete event systems. *Information Sciences*, *246*, 115–132.
Sanamrad, T., & Kossmann, D. (2013). Query log attack on encrypted databases. In *Workshop on secure data management* (pp. 95–107). Springer.
Takai, S., & Oka, Y. (2008). A formula for the supremal controllable and opaque sublanguage arising in supervisory control. *SICE Journal of the Control, Measurement & Systems Integration*, *1*(4), 307–311.
Thorsley, D., & Teneketzis, D. (2007). Active acquisition of information for diagnosis and supervisory control of discrete event systems. *Discrete Event Dynamic Systems*, *17*(4), 531–583.
Tong, Y., Li, Z., Seatzu, C., & Giua, A. (2017). Verification of state-based opacity using Petri nets. *IEEE Transactions on Automatic Control*, *62*(6), 2823–2837.
Wang, W., Lafortune, S., Girard, A. R., & Lin, F. (2010). Optimal sensor activation for diagnosing discrete event systems. *Automatica*, *46*(7), 1165–1175.
Wu, Y.-C., & Lafortune, S. (2013). Comparative analysis of related notions of opacity in centralized and coordinated architectures. *Discrete Event Dynamic Systems*, *23*(3), 307–339.
Yeddes, M. (2016). Enforcing opacity with orwellian observation. In *2016 13th international workshop on discrete event systems (WODES)* (pp. 306–312). IEEE.
Yin, X., & Lafortune, S. (2017). A new approach for the verification of infinite-step and *K*-step opacity using two-way observers. *Automatica*, *80*, 162–171.
Yin, X., & Lafortune, S. (2019). A general approach for optimizing dynamic sensor activation for discrete event systems. *Automatica*, *105*, 376–383.
Yin, X., & Li, S. (2020). Synthesis of dynamic masks for infinite-step opacity. *IEEE Transactions on Automatic Control*, *65*(4), 1429–1441.
Yin, X., Li, Z., Wang, W., & Li, S. (2019). Infinite-step opacity and *K*-step opacity of stochastic discrete-event systems. *Automatica*, *99*, 266–274.
Yin, X., Zamani, M., & Liu, S. (2021). On approximate opacity of cyber-physical system. *IEEE Transactions on Automatic Control*, *66*(4), 1630–1645.
Zeng, W., & Koutny, M. (2019). Quantitative analysis of opacity in cloud computing systems. *IEEE Transactions on Cloud Computing*.
Zhang, B., Shu, S., & Lin, F. (2015). Maximum information release while ensuring opacity in discrete event systems. *IEEE Transactions on Automation Science and Engineering*, *12*(4), 1067–1079.
Zinck, G., Ricker, L., Marchand, H., & Hélouët, L. (2020). Enforcing opacity in modular systems. In *IFAC world congress*.

Junyao Hou was born in Hunan, China, in 1993. He received the B.S. degree from Southwest University in 2015, the M.S. degree from Xidian University in 2018, all in automation. He is pursuing the Ph.D. degree at Shanghai Jiao Tong University in the Department of Automation. His research interests focus on the security of cyber–physical systems and abstraction-based synthesis in formal methods.

Xiang Yin was born in Anhui, China, in 1991. He received the B.Eng. degree from Zhejiang University in 2012, the M.S. degree from the University of Michigan, Ann Arbor, in 2013, and the Ph.D. degree from the University of Michigan, Ann Arbor, in 2017, all in electrical engineering. Since 2017, he has been with the Department of Automation, Shanghai Jiao Tong University, where he is an Associate Professor. His research interests include formal methods, discrete-event systems and cyber–physical systems. Dr. Yin is serving as the co-chair of the *IEEE CSS Technical Committee on Discrete Event Systems*, an Associate Editor for the *Journal of Discrete Event Dynamic Systems: Theory & Applications*, and a member of the *IEEE CSS Conference Editorial Board*. Dr. Yin received the IEEE Conference on Decision and Control (CDC) Best Student Paper Award Finalist in 2016.

Shaoyuan Li was born in Hebei, China, in 1965. He received the B.S. and M.S. degrees in automation from the Hebei University of Technology, Tianjin, China, in 1987 and 1992, respectively, and the Ph.D. degree from Nankai University, Tianjin, in 1997. Since 1997, he has been with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, where he is currently a chair Professor. His current research interests include model predictive control, dynamic system optimization, and cyber–physical systems. He is the vice-president of the Chinese Association of Automation.