Verification of trajectory-dependent opacity properties via fault diagnosis

Virginia Maria Alterio[1], Tianyu Liu[1], Carla Seatzu[1], Alessandro Giua[1]

***Abstract*— In this paper, we focus on two properties related to the privacy of partially observable discrete event systems: state-trajectory opacity and initial-state opacity. Both properties are trajectory-dependent, meaning that if a trajectory violates the property, all of its continuations will also violate it. Our contribution lies in demonstrating that the verification of these two properties can be addressed using the notion of fault diagnoser. Specifically, we define two ad hoc automata — one for each property — and show that, by analyzing their respective diagnosers, the properties can be verified. While the complexity of our method is the same as that of other state of the art procedures, our approach opens the door to tackling complex problems by combining approaches developed within two areas of opacity and diagnosis.**

I. Introduction

Opacity is a fundamental security property of discrete event systems (DES) that prevents external observers from inferring sensitive information about the system's behavior.

The literature distinguishes four primary variants—*current state opacity* (CSO), *initial state opacity* (ISO), *K-step opacity*, and *infinite-step opacity* (INSO)—each defined in both *weak* and *strong* forms; detailed definitions and theoretical foundations are provided in [7], [8].

In this paper, we propose a framework that converts opacity analysis into a fault-diagnosis problem, addressing the strong version of both *infinite-step opacity* and *initial-state opacity*. However, we prefer to call *state-trajectory opacity* the property known in literature as infinite-step opacity. In fact, this property is concerned with the past state trajectory of the system which can be arbitrarily long but is always finite. In the following when no confusion arises, we will omit the term 'strong'.

Ma et al. [8] first formally provided the definition of *(strong) infinite step opacity* (∞-SO) and proposed an ∞-step recognizer for verification, with an algorithm complexity of $O\left(2^{2|X|} \cdot |E_o|\right)$. Li et al. [10] verify ∞-SO for NFAs by building a verifier to check if 'safe' (non-secret-revealing) state estimates always persist for any observation sequence, achieving this with $O(2^{2|X|} \cdot |E_o|)$ complexity. Han et al. [11] verify ∞-SO by checking the absence of any state of the form $(\cdot, \emptyset)$ within their previously proposed concurrent-composition structure, the construction of which has a time complexity of $\mathcal{O}\left((|\Sigma_o||\Sigma_{uo}| + |\Sigma|)|X|^2 2^{|X|}\right)$.

Saboori and Hadjicostis [2] verify ISO using an observer with space complexity $O(2^{|X|^2})$. Wu and Lafortune [4] proposes verifying ISO by constructing an observer (called G_R-based ISE) of the reverse automaton G_R, with a worst-case complexity of $O\left(2^{|X|}\right)$. Balun and Masopust [7] establish that ISO with a single observable event is NL-complete.

Opacity and fault diagnosis share fundamental theoretical connections, as established in the literature. Lin [3] demonstrated that diagnosability can be formulated as a special case of the opacity problem through appropriate parameterization of system models. Building on this relationship, Lafortune et al. [5] characterized opacity as a strengthened variant of non-diagnosability. Our research leverages this theoretical equivalence by reformulating opacity verification within the fault diagnosis paradigm. Although our algorithm does not improve computational complexity, its practical advantages are evident in implementation. We are able to apply well-established diagnostic methodologies and analytical tools, and a notable advantage of our framework is its elegant handling of complex opacity scenarios: multiple disjoint or hierarchically structured secret sets $S_1, S_2, \ldots$ can be efficiently encoded through dedicated fault markers f_i for each classification category, with all markers processed through a unified observer architecture.

Our main contributions are summarized as follows.

1) We consider infinite-step opacity which we prefer to rename, more intuitively, *state-trajectory opacity*. Replacing each event labeling a transition inputting into a secret state with a new fault event, we can leverage on the fault-diagnosis method — monitor $\rightarrow$ recogniser $\rightarrow$ diagnoser — to verify state-trajectory opacity with time-complexity $O\left(|E_{obs}| \cdot 2^{2|X|}\right)$ and space complexity $O\left(2^{2|X|}\right)$.
2) To check initial-state opacity, we first build a new automaton that has a unique start state. From the new start state, we add unobservable transitions going to all previous initial states, marking only those going to secret states as faults. The standard diagnoser for this modified automaton efficiently decides initial-state opacity with complexity $O(2^{2(|X|+1)})$, as both the plant and its monitor now share a unique start state, making their synchronous composition simple and efficient.

We believe this approach can naturally lead to adapt

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

[1]Virginia Maria Alterio, Tianyu Liu, Carla Seatzu, and Alessandro Giua are with the Department of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy {virginiam.alterio, giua, carla.seatzu}@unica.it, t.liu@studenti.unica.it

tools developed for diagnosis to opacity enforcing for both properties. As an example, in [13] we have shown how a *joint diagnoser* can be used to determine how an attacker can modify the diagnosis state and potentially endanger the diagnosability of a plant. This approach can also be adapted to opacity enforcing, assuming the "attacker" is an operator which "edits" the observation of an intruder monitoring the plant to mislead it.

II. Background

A partially observed automaton is a non-deterministic finite automaton (NFA) composed by a deterministic finite automaton (DFA) and an observation mask M. The basic notions are here recalled.

A *deterministic finite automaton* describes a discrete-event system with the structure reported in Definition II.1.

Definition II.1 (Deterministic finite automaton)**.** *A deterministic finite automaton (DFA) is a four-tuple $G = (X, E, \delta, x_0)$ where:*

- *X is the finite set of states;*
- *E is the finite set of events called alphabet;*
- *$\delta : X \times E \to X$ is the next-state transition function;*
- *$x_0 \in X$ is the initial state.*

Note that transition function δ may be partially defined, i.e., $\delta(x, e)$ may not be defined for some state $x \in X$ and event $e \in E$. We also consider an extended transition function $\delta : X \times E^* \to X$ defined as follows: for all $x \in X$ $\delta^*(x, \varepsilon) = x$ and $\delta^*(x, se) = \delta(\delta^*(x, s), e)$ where $s \in E^*$ and $e \in E$.

Given a DFA $G = (X, E, \delta, x_0)$, the *generated language* of G is the language

$$L(G) = \{s \in E^* | \delta(x_0, s) \text{ is defined}\}.$$

Furthermore, we use the notation $L(G, x)$ to denote the set of words generated starting from an arbitrary state $x \in X$, namely

$$L(G, x) = \{s \in E^* | \delta(x, s) \text{ is defined}\}.$$

Finally, given a subset of states $Y \subseteq X$, we denote as

$$L(G, Y) = \bigcup_{x \in Y} L(G, x)$$

the language generated from Y.

We consider partially observed systems consisting in a DFA with an observation mask $M : E \to E_{obs} \cup \{\varepsilon\}$.

Definition II.2 (Observation mask)**.** *Given an alphabet E and a subset of events $E_{obs} \subseteq E$, a mask is a function $M : E \to E_{obs} \cup \{\varepsilon\}$ such that for events $e \in E_{obs}$ holds $M(e) = e$.*

The mask partitions the set of events as $E = E_{obs} \cup E_{rel} \cup E_{uo}$, where:

- events in E_{obs} are called *observable events*;
- events $e \in E_{rel}$ are such that $M(e) \in E_{obs}$ and are called *relabeled events*;

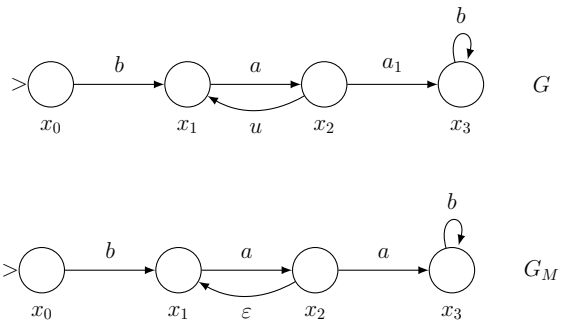

Fig. 1. A DFA G partially observed through mask $M(a) = a$, $M(b) = b$, $M(a_1) = a$ and $M(u) = \varepsilon$ (top); the corresponding DFA G_M (bottom).

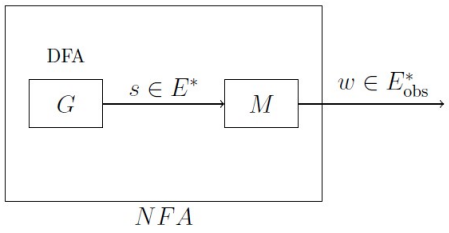

Fig. 2. A DFA G partially observed through mask M.

- events $e \in E_{uo}$ are such that $M(e) = \varepsilon$ and are called *unobservable events*.

Given a DFA G partially observed through a mask M, we denote G_M the new automaton obtained by relabeling each event $e \in E$ with $M(e)$ as in Fig. 2.

Formally, a non-deterministic automaton is defined as follows.

Definition II.3 (Non-deterministic finite automaton)**.** *A non-deterministic finite automaton (NFA) is a four-tuple $G = (X, E, \Delta, x_0)$ where:*

- *X is the finite set of states;*
- *E is the alphabet;*
- *$\Delta \subseteq X \times E \cup \{\varepsilon\} \times X$ is the transition relation;*
- *$x_0 \in X$ is the initial state.*

The *generated language* of NFA G_M corresponding to a DFA G partially observed through mask M is the set of words:

$$L(G_M) = \{w \in E^*_{obs} \mid (\exists s \in L(G))\ w = M(s)\}.$$

Example II.1. Fig. 1 (top) shows a DFA G with $E = \{a, a_1, b, u\}$ and $E_{obs} = \{a, b\}$, $E_{rel} = \{a_1\}$ and $E_{uo} = \{u\}$ with $M(a_1) = a$. The corresponding NFA G_M is shown in Fig. 1 (bottom).

According to Definition II.3, in an NFA we write $(x, e, x') \in \Delta$ to denote the transition going from state x to state x' via event e. The same notation can be used for a DFA and we write $(x, e, x') \in \delta$ to denote that $\delta(x, e) = x'$.

Definition II.4 (Concurrent composition for DFA)**.** *Let $G' = (X', E', \delta', x'_0)$ and $G'' = (X'', E'', \delta'', x''_0)$ be two DFAs.*

Their concurrent composition is the DFA $G = (X, E, \delta, x_0)$ *that generates language* $L(G) = L(G') \parallel L(G'')$ *and:*

- *event set* $E = E' \cup E''$*;*
- *state space* $X \subseteq X' \times X''$*;*
- *initial state* $x_0 = (x'_0, x''_0)$*;*
- *transition function* $\delta : X \times E \to X$ *defined as follows: for* $x = (x', x'')$

$$\delta(x,e) = \begin{cases} (\bar{x}', x'') & \text{if } e \in E' \setminus E'', \delta'(x', e) = \bar{x}' \\ (x', \bar{x}'') & \text{if } e \in E'' \setminus E', \delta''(x'', e) = \bar{x}'' \\ (\bar{x}', \bar{x}'') & \text{if } e \in E' \cap E'', \\ & \quad \delta'(x', e) = \bar{x}', \delta''(x'', e) = \bar{x}'' \\ \textit{undefined} & \textit{otherwise}. \end{cases}$$

III. State estimation and diagnosis for partially observed DFA

The evolution of a system is given by sequences of consecutive events, called *runs*.

Definition III.1 (Run)**.** *Given a DFA* $G = (X, E, \delta, x_0)$*, a run of length* k *is a sequence of states and transitions:*

$$x_{(0)} \xrightarrow{e_1} x_{(1)} \xrightarrow{e_2} \cdots \xrightarrow{e_k} x_{(k)}$$

where: for all $i = 0, ..., k$ *it holds that* $x_{(i)} \in X$ *and for all* $i = 1, ..., k$ *it holds that* $x_{(i)} = \delta(x_{(i-1)}, e_i)$*. This run produces word* $w = e_1 e_2 ... e_k$*, starting from* x_0 *and reaching* x_k*.*

Example III.1. Example of runs for the automaton G in Fig. 1 are:

$$x_0 \xrightarrow{b} x_1 \xrightarrow{a} x_2 \xrightarrow{a_1} x_3$$

generating word $s = baa_1$ which brings the system from state x_0 to state x_3 and run

$$x_0 \xrightarrow{b} x_1 \xrightarrow{a} \varepsilon \xrightarrow{u} x_1 \xrightarrow{a} x_2$$

which brings the system from state x_0 to state x_2. Note however that these runs produce the same observation $w = baa$.

This means that the observation produced by a partially observed DFA does not always allows one to reconstruct with certainty the run that generated it, and, consequently, the state reached.

Definition III.2. *Given a DFA and mask* M*, it is possible to define the set of sequences consistent with observation* $w \in E^*_{obs}$ *as:*

$$\mathcal{S}(w) = \{s \in L(G) | M(s) = w\} = M^{-1}(w) \cap L(G)$$

and the set of states consistent with observation w *as:*

$$\mathcal{X}(w) = \{\delta(x, s) | s \in \mathcal{S}(w)\}$$

Definition III.2 is better explained through Example III.2.

Example III.2. Consider again the partially observed DFA G in Fig. 1. Given observation $w = baa$, the set of consistent sequences is $\mathcal{S}(w) = \{baa_1, baua, bauau\}$ and correspondingly the set of consistent states is $\mathcal{X}(w) = \{x_1, x_2, x_3\}$.

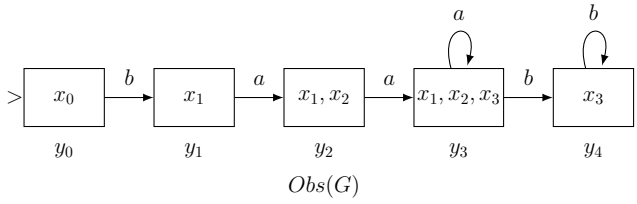

Fig. 3. Observer $Obs(G)$ for partially observed DFA G in Fig. 1.

A. State estimation with an observer

This motivates the definition of *observer* which is a DFA equivalent to a partially observed NFA.
For this purpose, it is possible to develop the *observer* of the considered automaton, as reported in Definition III.3.

Definition III.3 (Observer)**.** *Let* $G = (X, E, \delta, x_0)$ *be a DFA partially observed through a mask* $M : E \to E_{obs} \cup \{\varepsilon\}$*. The observer* $Obs(G) = (Y_{obs}, E_{obs}, \delta_{obs}, y_{0_{obs}})$ *is a DFA such that:*

- $Y_{obs} \subseteq 2^X$*: meaning that each state of the observer is a subset of states of* G*;*
- $L(Obs(G)) = M(L(G))$*: the language of the observer coincides with the set of possible observations produced by* G*;*
- *for all* $w \in M(L(G))$*:* $q = \delta_{obs}(y_{0_{obs}}, w) \equiv \mathcal{X}(w)$*, the state reached in* $Obs(G)$ *by an observation* w *coincides with the set of states in* G *consistent with* w*.*

We refer to the book by Cassandras and Lafortune [9], as well as some of the references therein, for the algorithm used to construct the observer.

The observer for the partially observed DFA G in Fig. 1 is shown in Fig. 3. As expected from Example III.2, given observation $w = baa$ it holds that $\delta_{obs}(y_{0_{obs}}, baa) = \{x_1, x_2, x_3\} \equiv \mathcal{X}(w)$.

B. Fault diagnosis with a diagnoser

Diagnosis is a process that verifies the occurrence of faults during the evolution of the system. A fault is modeled as a transition which in our approach may or may not be observable.

The *diagnoser* [1] is defined as follows.

Definition III.4 (Diagnoser)**.** *Let* $G = (X, E, \delta, x_0)$ *be a DFA partially observed through a mask* $M : E \to E_{obs} \cup \{\varepsilon\}$ *and let* $E_f \subseteq E$ *be a set of fault transitions. The diagnoser* $Diag(G) = (Y_{diag}, E_{obs}, \delta_{diag}, y_{0_{diag}})$ *is a DFA such that:*

- $Y_{diag} \subseteq 2^{X \times \{N,F\}}$*, i.e., each state of the diagnoser is a subset of pairs* (x, d) *where* $x \in X$ *is a state of* G *and* $d \in \{N, F\}$*;*
- $L(Diag(G)) = M(L(G))$*: the language of the diagnoser coincides with the set of possible observations produced by* G*;*
- *for all* $w \in M(L(G))$ *with* $q = \delta_{obs}(y_{0_{obs}}, w)$ *it holds that* $(x, N) \in q$ *(resp.,* $(x, F) \in q$*) iff there exists a*

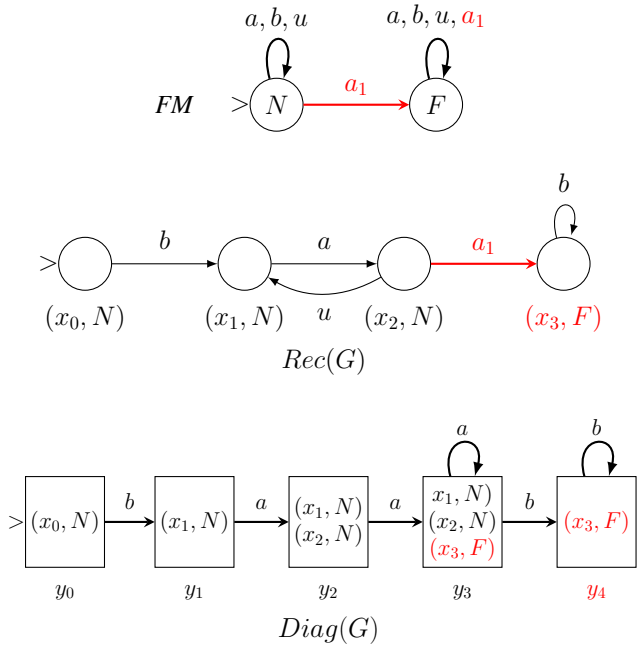

Fig. 4. Fault monitor *FM*, recognizer $Rec(G)$ and diagnoser $Diag(G)$ for DFA G in Fig. 1 with $E_f = \{a_1\}$.

string $s \in \mathcal{S}(w)$ that does not contain (resp., contains) a fault event in E_f and $\delta(x_0, s) = x$.

An algorithm for the construction of the diagnoser has been proposed in [1] and referred to in most of the literature [9]. In this paper we follow a different approach that is based on the notions of *fault monitor* and *fault recognizer*.

Definition III.5 (Fault monitor)**.** *Given an alphabet E and a set of fault events $E_f \subseteq E$, a fault monitor is the DFA $FM = (X_M, E, \delta_M, x_{M,0})$, with:*

- *set of states $X_M = \{N, F\}$;*
- *initial state $x_{M,0} = N$;*
- *transition function $\delta_M : X_M \times E \to X_M$ such that:*
  - *$\delta_M(N, e) = N$ if $e \in E \setminus E_f$,*
  - *$\delta_M(N, e) = F$ if $e \in E_f$,*
  - *$\delta_M(F, e) = F$ for all $e \in E$.*

Fig. 4 shows the fault monitor *FM* for DFA G in Fig. 1 with set of fault events $E_f = \{a_1\}$.

Definition III.6 (Fault recognizer)**.** *Given a DFA G with alphabet E and a set of fault events $E_f \subseteq E$, let FM be its fault monitor. The fault recognizer for G is $Rec(G) = G \| FM$, where $\|$ denotes the concurrent composition operator.*

This automaton has alphabet E and generates language $L(Rec(G)) = L(G)$, with structure:

- *set of states $X_R \subseteq X \times \{N, F\}$;*
- *initial state $x_{R0} = (x_0, N)$;*
- *transition function $\delta : X_R \times E \to X_R$.*

In simple words, the states of $Rec(G)$ can be of two types: (x, N), with $x \in X$, if the fault has not occurred while reaching x; (x, F), with $x \in X$, if the fault has occurred while reaching x.

Finally, the diagnoser can be computed as the observer of the partially observed system composed by $Rec(G)$ observed through a mask M. This result is formalized by Proposition III.1 whose proof follows from the definitions of monitor, recognizer, and diagnoser.

Proposition III.1. *[12] Given an automaton G with alphabet E and a set of fault events E_f, let $Rec(G)$ be its fault recognizer. Assume events in G — and hence in $Rec(G)$ — are partially observed through mask M.*

The diagnoser of G according to Definition III.4 can be computed as $Diag(G) = Obs(Rec(G))$.

Example III.3. *The fault recognizer $Rec(G)$ and the diagnoser $Diag(G)$ for the partially observed DFA G in Fig. 1 with $E_f = \{a_1\}$ are shown in Fig. 4.*

IV. State-trajectory opacity

Before introducing the notion of state-trajectory opacity, let us formalize the notion of *state-trajectory*.

Definition IV.1 (State trajectory)**.** *Let $G = (X, E, \delta, x_0)$ be a DFA. Consider a generated string $s = e_1 e_2 ... e_k \in L(G)$ and assume it is produced by a run*

$$x_{(0)} \xrightarrow{e_1} x_{(1)} \xrightarrow{e_2} \cdots \xrightarrow{e_k} x_{(k)}$$

where $x_{(0)} = x_0$. We define state trajectory of s, $traj(s) = \{x_{(i)} | i = 0, ..., k\}$, the set of states that belong to the run that produces s.

The notion of state-trajectory opacity can now be formalized.

Definition IV.2 (State-trajectory opacity)**.** *Let $G = (X, E, \delta, x_0)$ be a DFA partially observed through a mask M and let $S \subseteq X$ be a set of of secret states.*

System G is state-trajectory opaque w.r.t. S and M if for all $s \in L(G)$ such that $traj(s) \cap S \neq \emptyset$ there exists another string $s' \in L(G)$ such that $traj(s') \cap S = \emptyset$ and $M(s) = M(s')$.

In plain words, a system is state-trajectory opaque if, for any possible evolution of the system, the intruder can never infer if the system currently is, or has been in the past, in a secret state.

As mentioned in the Introduction, *state-trajectory opacity* coincides with (strong) *infinite-step opacity* introduced by Ma et al. in [8]. In this paper, we prefer to rename this property because we want to better emphasize the fact that it is trajectory-dependent, meaning that if a trajectory violates the property, all of its continuations will also violate it.

A. Verification of state-trajectory opacity

In this paper we show that the problem of state-trajectory verification of a partially observed DFA G may be converted into a reachability analysis problem in the diagnoser of G obtained assuming as fault events all the events entering a secret state.

In the following we assume, without loss of generality, that $x_0 \notin S$. In fact, if $x_0 \in S$ no further analysis is needed, since the plant is obviously not state-trajectory opaque.

Definition IV.3 (Relabeled automaton for state-trajectory opacity)**.** *Let $G = (X, E, \delta, x_0)$ be a DFA partially observed through a mask M and let $S \subseteq X$ be a set of secret states.*

The relabeled automaton for state-trajectory opacity

$$G_{sto} = (X, E_{sto}, \delta_{sto}, x_0)$$

with mask M_{sto} and set of fault transitions E_f can be constructed with the following procedure.

1) $E_f := \emptyset$, $\delta_{sto} := \delta$, $M_{sto} = M$;
2) ***for*** $(x, e, x') \in \delta$,
   if $x' \in S$ ***then***
   i. $E_f := E_f \cup \{e_f\}$;
   ii. $\delta_{sto} := (\delta_{sto} \cup \{(x, e_f, x')\}) \setminus \{(x, e, x')\}$;
   iii. $M_{sto}(e_f) := M(e)$.
   end if
   end for
3) $E_{sto} := E \cup E_f$.

To construct the relabeled DFA G_{sto} each event e labeling a transition in G inputting into a secret state is changed into a new event e_f with $M_{sto}(e_f) = M(e)$. The set of new events E_f is the set of fault events.

Theorem IV.1. *Let $G = (X, E, \delta, x_0)$ be a DFA partially observed through a mask M and let $S \subseteq X$ be a set of secret states.*

Consider the relabeled DFA $G_{sto} = (X, E \cup E_f, \delta_{sto}, x_0)$ be as in Definition IV.3 with mask M_{sto} and set of fault events E_f.

System G is state-trajectory opaque w.r.t. S and M if and only if in the diagnoser $Diag(G_{sto})$ there exists no state which only contains pairs with label F.

Proof: The result follows by the definition of diagnoser: if there exists a state of $Diag(G_{sto})$ that only contains pairs with label F, it means that all the observations leading to such a state of $Diag(G_{sto})$ correspond to runs in G_{sto} that contain events in E_f. By definition of G_{sto}, this is equivalent to saying that all such runs pass through some secret state.

On the other hand, if no state of $Diag(G_{sto})$ only contains label F, it means that there exists no state of $Diag(G_{sto})$ that can be reached only by observations produced by runs in G_{sto} involving events in E_f. This implies that there exists no observation of G that only corresponds to trajectories passing through secret states. □

The verification procedure runs in $O\left(|E_{obs}| \cdot 2^{2|X|}\right)$ time and $O\left(2^{2|X|}\right)$ space, which is exponential in the size of the original state set – as is unavoidable for observer-based opacity or diagnosability problems.

Example IV.1. Consider partially observed DFA $G = (X, E, \delta, x_0)$ in Fig. 5. The alphabet is divided into $E_{obs} = \{a\}$ and $E_{uo} = \{u\}$, and the set of secret state is $S = \{x_3\}$.

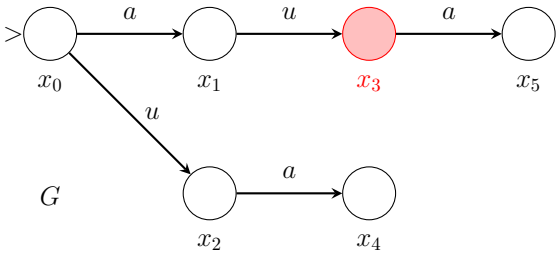

Fig. 5. DFA G in Example IV.1.

The relabeled automaton G_{sto} is obtained from G relabeling the transition (x_1, u, x_3) entering the secret state as (x_1, u_f, x_3). The relabeled automaton has set of fault events $E_f = \{u_f\}$ and set of unobservable events $E_{sto,uo} = \{u, u_f\}$. This automaton and the corresponding fault monitor *FM* are shown in Fig. 6. The fault recognizer $Rec(G_{sto}) = G_{sto} \| FM$ is also shown in the same figure.

The last step to verify state-trajectory opacity consists in computing $Diag(G_{sto}) = Obs(Rec(G_{sto}))$, which is shown at the bottom of Fig. 6. By Theorem IV.1, we conclude that that G is not state-trajectory opaque, since the diagnoser contains state $y_2 = \{(x_5, F)\}$.

V. Initial State Opacity

In this section the focus is on initial-state opacity. We first recall the initial-state opacity definition, then we show how the diagnoser may be used to verify such a property.

Definition V.1. *Let $G = (X, E, \delta, X_0)$ be a DFA*[1] *whose initial state is known to belong to a given set $X_0 \subseteq X$. Let G be observed through a mask M, and let $S \subseteq X_0$ be the set of secret initial states. System G is* initial-state opaque *(ISO) w.r.t S and M if*

$$\forall s \in L(G, S),\ \exists s' \in L(G, X_0 \setminus S) : M(s') = M(s).$$

Analogously to state-trajectory opacity verification, we convert the problem of initial-state opacity verification into a reachability analysis problem in the diagnoser of a new automaton. In this case the new automaton is called *modified automaton for initial-state opacity*, it is denoted as G_{iso} and is computed in accordance with the following definition.

Definition V.2 (Modified automaton for initial-state opacity)**.** *Let $G = (X, E, \delta, X_0)$ be a DFA partially observed through a mask M and let $S \subseteq X_0$ be a set of secret states.*

The modified automaton for initial-state opacity

$$G_{iso} = (X_{iso}, E_{iso}, \delta_{iso}, x_{start})$$

with mask M_{iso} and set of fault transitions E_f can be constructed with the following procedure.

1) $X_{iso} = X \cup \{x_{start}\}$;
2) $E_{iso} = E \cup \{u_N, u_S\}$;

[1]The notation $G = (X, E, \delta, X_0)$ where $X_0 \subseteq X$ is not a singleton, is used to denote an automaton which is structurally deterministic but whose unknown initial state may take values in the set X_0, as is the case in the context of initial-state opacity.

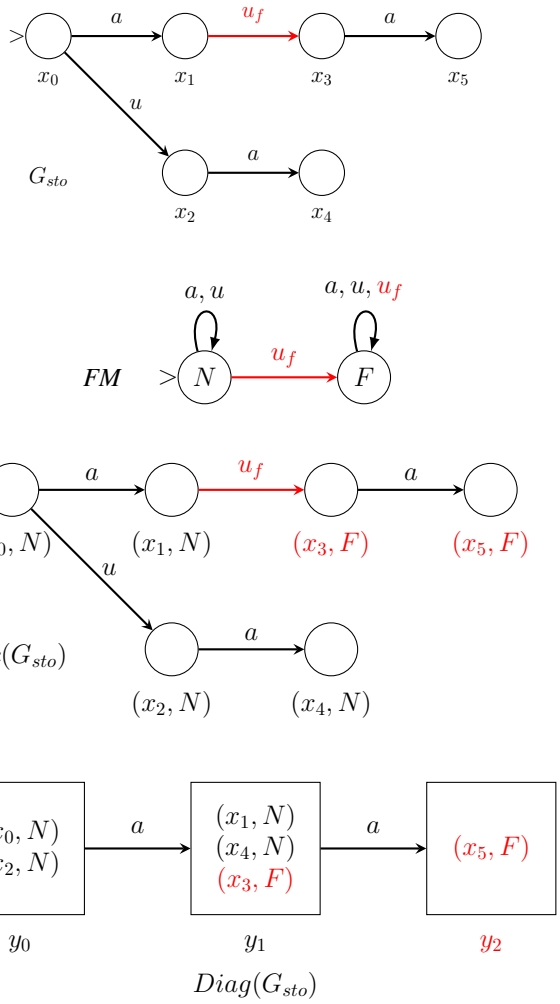

Fig. 6. Relabeled automaton G_{sto} and its fault monitor *FM*, recognizer $Rec(G_{sto})$ and diagnoser $Diag(G_{sto})$ in Example IV.1 for verifying state-trajectory opacity.

3) $E_f = \{u_S\}$;
4) $\delta_{iso} := \delta$, $M_{iso} := M$;
5) ***for*** $x \in S$,
   $\delta_{iso} := \delta_{iso} \cup \{(x_{start}, u_S, x)\}$;
   end for
6) ***for*** $x \in X_0 \setminus S$,
   $\delta_{iso} := \delta_{iso} \cup \{(x_{start}, u_N, x)\}$;
   end for
7) $M_{iso}(u_S) := \varepsilon$, $M_{iso}(u_N) := \varepsilon$.

To construct the modified DFA G_{iso} we add a new state x_{start} connected to each secret initial state by a transition labeled u_S, and to each non-secret initial state by a transition labeled u_N. The new state x_{start} becomes the unique initial state. The two new events u_S and u_N are unobservable. The set of fault events E_f contains only event u_S.

Theorem V.1. *Let $G = (X, E, \delta, X_0)$ be a DFA partially observed through a mask M and let $S \subseteq X_0$ be a set of secret states.*

Consider the modified DFA $G_{iso} = (X_{iso}, E_{iso}, \delta_{iso}, x_{start})$ as in Definition V.2 with mask M_{iso} and set of fault events E_f.

System G is initial-state opaque w.r.t. S and M if and only if in the diagnoser $Diag(G_{iso})$ there exists no state which only contains pairs with label F.

Proof: The result follows by the definition of diagnoser and the procedure used to construct G_{iso}: if there exists a state of $Diag(G_{iso})$ that only contains pairs with label F, it means that to all the observations leading to such a state of $Diag(G_{iso})$ it correspond runs in G_{iso} that contain events in E_f. By definition of G_{iso}, this is equivalent to saying that all such runs visit a secret state as soon as the initial state x_{start} has been left at the very beginning.

On the other hand, if no state of $Diag(G_{sto})$ only contains label F, it means that there exists no state of $Diag(G_{sto})$ that can be reached only by observations produced by runs involving events in E_f. This implies that there exists no observation of G produced only by runs starting from a secret state. □

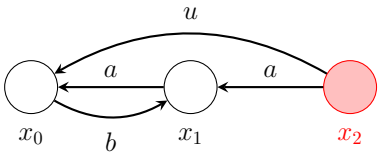

Fig. 7. DFA G in Example V.1.

Example V.1. *Consider the DFA $G = (X, E, \delta, X_0)$ in Fig. 7 where $X_0 = X$, i.e., there is total uncertainly about the initial state. The alphabet is divided in $E_{obs} = \{a, b\}$ and $E_{uo} = \{u\}$, and the secret state is $S = \{x_2\}$.*

The modified automaton for initial-state opacity G_{iso} is reported in Fig. 8. It has been obtained from G adding the new state x_{start} and the three transitions (x_{start}, u_N, x_0), (x_{start}, u_N, x_1), and (x_{start}, u_S, x_2). The set of fault events for G_{iso} is $E_f = \{u_S\}$, while $E_{iso} = \{a, b, u, u_N, u_S\}$ leading to the fault monitor FM the same figure.

The recognizer $Rec(G_{iso})$ with unobservable events $E_{iso,uo} = \{u, u_N, u_S\}$ and the diagnoser $Diag(G_{iso})$ are also shown. In the diagnoser there exist states that only contain pairs with label F, namely states y_3 and y_6. Therefore, we conclude that G is not initial state opaque w.r.t. the secret S.

VI. Conclusions and Future Work

This paper focuses on two opacity properties for discrete event systems under partial observation: state-trajectory opacity and initial-state opacity. Both properties have already been considered in the literature, and methods for their verification have been provided. In this paper, we propose alternative verification methods based on the notion of fault diagnoser.

In more detail, we introduce two new automata with their respective observation masks, called the *relabeled automaton for state-trajectory opacity* and the *modified automaton for*

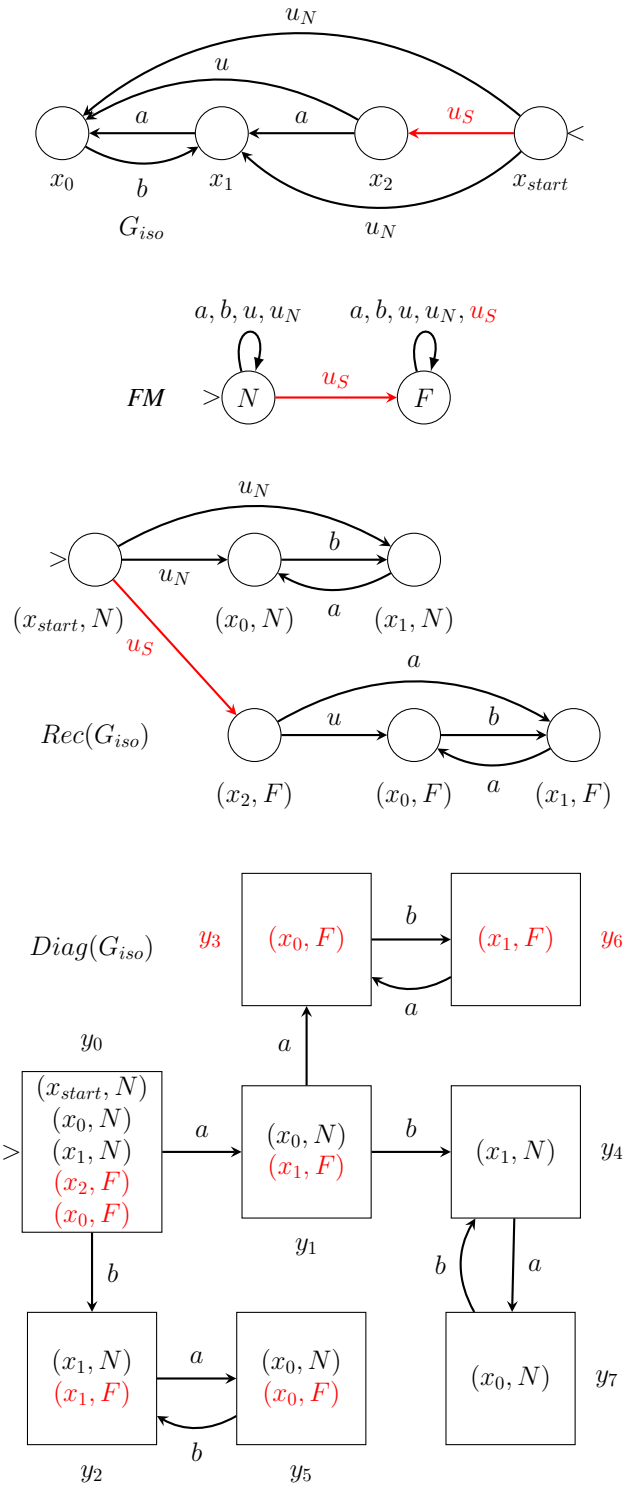

Fig. 8. Modified automaton G_{iso} and its fault monitor FM, recognizer $Rec(G_{iso})$ and diagnoser $Diag(G_{iso})$ in Example V.1 for verifying initial-state opacity.

initial-state opacity. The alphabets of these automata are partitioned into faulty and non-faulty events. Based on these partitions, fault diagnosers are constructed. We show how the verification of the two opacity properties can be carried out by analyzing the states of these diagnosers.

Although the complexity of the proposed verification method is exponential in the number of states of the original automaton —- similarly to previous approaches in the literature —- we believe that this method may be useful in addressing more complex problem formulations. In fact, the diagnoser contains additional information that is lost in other verification tools. This additional insight may enable the solution of opacity enforcement problems and opacity verification under attack, which are particularly challenging in the context of cyber-physical systems.

References

[1] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen. D. Teneketzis, Diagnosability of discrete-event systems. IEEE Transactions on Automatic Control, 40(9):1555-1575, 1995.

[2] A. Saboori, C.N. Hadjicostis, Verification of initial-state opacity in security applications of DES, in: Proc. of the 9th International Workshop on Discrete Event Systems, pp. 328–333, 2008.
Volume 42, Issue 5, Pages 46-51, 2009.

[3] F. Lin, Opacity of discrete event systems and its applications, Automatica, Volume 47, Issue 3, Pages 496-503, 2011.

[4] Y. Wu and S. Lafortune, Comparative analysis of related notions of opacity in centralized and coordinated architectures. Discrete Event Dynamic Systems 23, 3, pp. 307–339, 2013.

[5] S. Lafortune, F. Lin, C. N. Hadjicostis, On the history of diagnosability and opacity in discrete event systems. Annual Reviews in Control, Volume 45, Pages 257-266, 2018.

[6] C.N. Hadjicostis, Estimation and Inference in Discrete Event Systems. Springer, 2020.

[7] J. Balun and T. Masopust. Comparing the notions of opacity for discrete-event systems. Discrete Event Dynamic Systems, 31, 553-582, 2021.

[8] Z. Ma, X. Yin, and Z. Li, Verification and Enforcement of Strong Infinite- and K-Step Opacity Using State Recognizers, Volume 133, 109838, 2021.

[9] C. Cassandras, S. Lafortune. Introduction to Discrete Event Systems. Springer, 2021.

[10] X. Li, C. N. Hadjicostis, and Z. Li, Reduced-Complexity Verification for K-Step and Infinite-Step Opacity in Discrete-Event Systems. Automatica, 128:109690, 2023.

[11] X. Han, K. Zhang, and Z. Li, Verification and Enforcement of Strong State-Based Opacity for Discrete-Event Systems. arXiv*:2401.10363, 2024.

[12] A. Giua, Notes for the Course: Analysis and Control of Cyber-Physical Systems. https://www.alessandro-giua.it/UNICA/ACCPS/notes_ACCPS_2025-03-04.pdf, 2025.

[13] T. Kang, C. Seatzu, Z. Li, A. Giua, A joint diagnoser approach for diagnosability of discrete event systems under attack. Automatica, 2025.