Property-Based Transparency: a New Utility Definition

Patrícia C Mayer, Felipe G Cabral, Públio M M Lima, Marcos V Moreira, Audine Subias, Yannick Pencolé

Property-Based Transparency: a New Utility Definition*

Patrícia C. Mayer[1] and Felipe G. Cabral[1] and Públio M. M. Lima[1] and Marcos V. Moreira[2] and Audine Subias[3] and Yannick Pencolé[4]

***Abstract*— Cyber-physical systems (CPSs) are composed of computational and physical components that interact to achieve a desired behavior for a given system. These systems are widely used in industrial applications and have guaranteed a solid improvement in productivity. However, their high connectivity also makes them more vulnerable to cyber-attacks, and thus, strategies to increase and/or guarantee the privacy of classified information are mandatory. On the other hand, the availability of system information, known as utility, to trustworthy agents is fundamental to operating large-scale CPSs efficiently, which establishes a trade-off between the privacy of information against malicious agents and the utility of data to legitimate receivers. In this paper, we explore the utility problem for CPSs abstracted as Discrete-Event Systems (DESs), where the intended receiver must know useful information about the system to achieve a desired goal. In this context, we consider that this information is the ability to be sure when a given property of interest is satisfied before the system evolves to a state where this property is no longer valid. To this end, we propose a new utility notion called Property-Based Transparency (PT), and a method for verifying whether a system is transparent with respect to the desired property. In addition, we discuss PT's applicability in two domains and present how it relates to the notion of current-state opacity and fault diagnosis of DES.**

I. Introduction

Cyber-Physical Systems (CPSs) comprise the interaction between computational and physical systems, and are a fundamental element of Industry 4.0. The components of CPSs are intrinsically interconnected, constantly exchanging information to achieve a desired behavior efficiently. Usually, this information is also communicated to a receiver that, by precisely interpreting the transmitted data, can increase the operational safeness of the CPS. However, the network that transmits sensitive data in these systems are susceptible to cyber-attacks, which can compromise the availability of system information.

*This work has been partially supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Finance Code 001 and FAPESC.

[1]Patrícia C. Mayer, Felipe G. Cabral, and Públio M. M. Lima are with Department of Automation and Systems, Federal University of Santa Catarina, Campus Trindade, Florianópolis, 88.040-900, SC, Brazil patricia.mayer@posgrad.ufsc.br; felipe.gomes.cabral@ufsc.br; publio.lima@ufsc.br

[2]Marcos V. Moreira is with COPPE - Electrical Engineering Program, Universidade Federal do Rio de Janeiro, Cidade Universitária, Ilha do Fundão, Rio de Janeiro, 21.945-970, RJ, Brazil moreira.mv@poli.ufrj.br

[3]Audine Subias is with LAAS-CNRS, INSA, University of Toulouse, France subias@laas.fr

[4]Yannick Pencolé is with LAAS-CNRS, CNRS, University of Toulouse, France yannick.pencole@laas.fr

In this work, we assume a Discrete Event System (DES) abstraction of the CPS to model its behavior [1], [2]. In the DES community, the problem of guaranteeing that an authorized receiver is able to obtain useful information about the system has been called utility [3], [4]. The utility problem can be considered as the other side of the coin of the opacity problem. A system is opaque when the secret information is not revealed to an external unauthorized observer, which usually accesses the system through an eavesdropping attack. Several works in the literature propose different definitions of opacity, such as current-state opacity [5], k-step opacity [6], initial-state opacity [7], current-state opacity based on outputs [8], to cite a few [9], [10], to deal with distinctive viewpoints on the opacity problem [11].

In addition, different techniques have been proposed in the literature to enforce opacity when the system is not opaque [12], [13]. Although these enforcement methods increase the system's privacy to an unauthorized agent, the utility of the transmitted information can be lost for a legitimate receiver. To overcome this drawback, some works have introduced notions of utility that must be preserved when an opacity enforcement technique is in place [4], [14], [15], [16].

In [4], the notion of utility was first introduced in the context of DES, where the utility property is satisfied when the number of transitions between the current system state and the state inferred by the receiver is smaller than a given number. However, secret states cannot be considered useful since they must never be reported. The same utility approach has been considered in [14] and [15].

Recently, the utility-ensured property has been proposed in [16]. A system is utility-ensured when the receiver can precisely know when the system reaches each one of the useful states. Therefore, if the receiver reaches an estimate composed only of useful states, the system is not utility-ensured since the receiver is not able to uniquely differentiate all useful states.

In this paper, we propose a new notion of utility called Property-Based Transparency (PT), where the useful information is whether a given property of interest is valid. The property of interest is the useful information the receiver needs to know. This problem arises from scenarios in which a remote supervisor must know when a given property of interest is satisfied before the system reaches a state where it is no longer true. In addition, this concept has applications in several fields related to cyber-security, such as in agent's geolocation [17], prevention of sensitive information leakage [5], or identifying a non-expected behavior in CPSs [18]. For instance, in the coordinated multi-robot control for urban

search and rescue scenario considered in [17], the supervisor must know if a given robot is in a geographic region to issue a control command, such as searching for human victims. In the opacity context, it can be required that the information transmitted through a communication channel be considered secret to an unauthorized agent but still available to a legitimate receiver. Additionally, to ensure the safety of a CPS process, the supervisor might be interested in identifying the occurrence of a fault event based on its observations.

Note that these problems cannot be addressed with the utility notion presented in [4] since we are concerned with detecting if the system has reached a given region and not if a minimum distance from a set of states to the current one is preserved. This work also differentiates from the utility-ensured notion proposed in [16], since we assume that there is no need for the receiver to precisely distinguish a useful state from another one, *i.e.*, we are only concerned with verifying if the receiver can always detect if a property of interest has been satisfied, which provides a broader solution to several practical applications.

In this paper, we introduce the notion of Property-Based Transparency as a utility definition and provide a method for verifying it. With the view to illustrate the applicability of PT, we apply it to CPS abstracted as DES in the contexts of current-state opacity based on events and fault diagnosis. It is important to remark that, as a general definition, PT can be used in several domains and engineering contexts. The application contexts of opacity and fault diagnosis are explored due to the proximity of these subjects to the utility problem in DES. Examples are provided throughout the text to illustrate the results and comparisons.

This paper is organized as follows. In section II, preliminary concepts of DESs and utility are presented. In Section III, we formulate the Property-Based Transparency problem. The verification method for PT is proposed in Section IV. In Section V, two applications of PT are presented. Finally, in Section VI, the conclusions are drawn, and some future works are presented.

II. Preliminaries

A DES is modeled as an automaton $G = (Q, \Sigma, f, q_0)$, where Q is the finite set of states, Σ is the finite set of events, $f : Q \times \Sigma \to Q$ is the deterministic partial transition function, and q_0 is the initial state. We denote by $\Gamma_G : Q \to 2^{\Sigma}$ the active event function of G, where $\Gamma_G(q) = \{\sigma \in \Sigma : f(q,\sigma)!\}$, where ! denotes that $f(q_0,\sigma)$ is defined. The domain of the transition function can be extended to $Q \times \Sigma^*$, where Σ^* denotes the Kleene-closure of Σ, as $f(q,\varepsilon) = q$ and $f(q,s\sigma) = f(f(q,s),\sigma)$, for all $s \in \Sigma^*$ and $\sigma \in \Sigma$, where ε denotes the empty trace. The language generated by G is defined as $L = \{s \in \Sigma^* : f(q_0,s)!\}$. A path p of length k of G is defined as $p = (q_1, \sigma_1, q_2, \ldots, \sigma_{k-1}, q_k)$, where $q_i \in Q$, for $i = 1, \ldots, k$, $\sigma_i \in \Sigma$, for $i = 1, \ldots, k-1$, and $f(q_i, \sigma_i) = q_{i+1}$, for $i = 1, \ldots, k-1$. Thus, associated with each path p of length k there is a sequence of events $s = \sigma_1\sigma_2 \ldots \sigma_{k-1}$, such that $f(q_1, s)$ is defined. The length of trace $s \in \Sigma^*$ is defined as $\|s\|$. The prefix closure of a language L is defined as $\bar{L} = \{s \in \Sigma^* : (\exists t \in \Sigma^*)[st \in L]\}$. The post-language of L after s is defined as $L/s = \{t \in \Sigma^* : st \in L\}$. Let G_1 and G_2 be automata models, then, the parallel composition between G_1 and G_2, $G_1 \| G_2$, is defined as usual [19].

Let $\Sigma = \Sigma_o \cup \Sigma_{uo}$, where Σ_o and Σ_{uo} are the sets of observable and unobservable events, respectively. Let $P_o : \Sigma^* \to \Sigma_o^*$ be a projection operation, where $P_o(\varepsilon) = \varepsilon$, $P_o(\sigma) = \sigma$, if $\sigma \in \Sigma_o$, and $P_o(\sigma) = \varepsilon$, if $\sigma \in \Sigma \setminus \Sigma_o$, and $P_o(s\sigma) = P_o(s)P_o(\sigma)$, for all $s \in \Sigma^*$ and $\sigma \in \Sigma$. Thus, $P_o(s)$ represents the observation of s. The automaton that generates language $P_o(L)$ is called an observer of G, denoted as $Obs(G, q_0)$ [19]. Each state of $Obs(G, q_0)$ corresponds to a state estimate of G considering the set of observable events Σ_o.

The notion of utility has been widely discussed in the literature since the seminal paper [4], and different notions have been proposed. In the sequel, the utility definition introduced in [16] is presented.

Definition 1 (Utility-Ensured Systems [16]): Given a system $G = (Q, \Sigma, f, q_0)$, a projection $P_o : \Sigma^* \to \Sigma_o^*$, and the set of useful states $Q_U \subseteq Q$, system G is utility-ensured with respect to P_o and Q_U, if: $\forall q_U \in Q_U$ and $\forall s \in L$ such that $f(q_0, s) = q_U$, $\forall q' \in Q \setminus \{q_U\}$ and $\forall s' \in L$ such that $f(q_0, s') = q'$, we have $P_o(s) \neq P_o(s')$. □

It is important to remark that, according to Definition 1 from [16], G is said to be "utility-ensured" if a useful state is always uniquely estimated under projection P_o. Note that some states cannot always be uniquely determined for a real application due to the occurrence of unobservable events. For example, for a given system where a transition from one state q to another state q' is labeled with an unobservable event, *i.e.*, $f(q, \sigma_{uo}) = q'$, where $\sigma_{uo} \in \Sigma_{uo}$, and $q, q' \in Q_U$, it is impossible to have an estimation with only state q, and G is trivially not utility-ensured.

III. Problem Formulation

In this paper, we consider that the plant and its supervisor or monitoring device (also called the receiver) can exchange information based on observable event occurrences. The communicated information is used by the receiver to estimate the plant's behavior. For example, to compute the next control event so that the system follows the designed specifications or to detect a non-desired behavior that must trigger an alarm for the operator. Therefore, the communication architecture is designed to guarantee that the receiver can distinguish if and when the plant is in one of the states of a given set, called useful states.

In many practical applications, the receiver must know when a given property $\mathscr{P}$, associated with a set of states, is satisfied before the system evolves to a state where $\mathscr{P}$ is false. In the geolocation problem from [17], the property $\mathscr{P}$ is true while the robot is in the region of interest, and the supervisor does not need to know precisely the robot's location during its mission. The opacity's enforcement problem, as discussed in [12], [13], is another context where the supervisor can be interested in detecting when a given property $\mathscr{P}$ is true instead of trying to estimate a

Fig. 1: The PT notion in a Cyber-Physical System.

set of system states. This is because the information being communicated is manipulated to guarantee that, for example, a set of secret states remain hidden for an attacker, which is not necessarily the same data the receiver is interested in. The notion of utility proposed in this paper is based on a property of interest related to the necessarily known observed network information for the supervisor to achieve a desired goal. We call this notion Property-Based Transparency (PT), formally introduced in the sequel and illustrated for an Industrial Cyber-Physical System in Figure 1.

A. *The notion of Property-Based Transparency*

Before introducing the definition of PT, it is necessary to define the following notation. The set of useful states Q_U, composed of the states that satisfy property $\mathscr{P}$, is defined as $Q_U = \{q \in Q : q \vDash \mathscr{P}\}$. The set of non-useful states is $Q_{\neg U} = Q \setminus Q_U$. We also define language $L_{\neg U}$ that contains all sequences of events of language L that reaches a non-useful state, *i.e.*, a state $q \in Q_{\neg U}$, as $L_{\neg U} = \{s \in L : f(q_0,s) \notin Q_U\}$. The transparency property is defined in the sequel.

Definition 2 (Property-Based Transparency): Given a system $G = (Q,\Sigma,f,q_0)$, a projection $P_o : \Sigma^* \to \Sigma_o^*$, and set Q_U, G is said to be transparent with respect to P_o and Q_U, if $\forall s = s'\sigma \in L$, $\sigma \in \Sigma$, such that $f(q_0,s) \in Q_U$ and $f(q_0,s') \in Q \setminus Q_U$, or $s = \varepsilon$ and $q_0 \in Q_U$, then:

$$(\exists n \in \mathbb{N})(\forall t \in L/s : \|t\| \geq n)$$
$$(\exists v \in \overline{\{t\}} : \forall v' \in \overline{\{v\}}, f(q_0,sv') \in Q_U)$$
$$(P_o(sv) \neq P_o(\omega), \forall \omega \in L_{\neg U})$$

□

In Definition 2, we consider that the system G is property transparent if for all sequences $s \in L$ that reach a state in Q_U, such that $s = s'\sigma$ and $f(q_0,s') \in Q_{\neg U}$, s has at least one suffix v that does not lead the system to a state in $Q_{\neg U}$ such that $P_o(sv) \neq P_o(\omega)$, for all $\omega \in L_{\neg U}$, as it is graphically illustrated in Figure 2. In other words, according to Definition 2, a system G is said to be transparent if, for all sequences $s = s'\sigma$ that reach a state $q \in Q_U$, where $f(q_0,s') \in Q_{\neg U}$, the receiver, based on the natural projection, can certainly know that property $\mathscr{P}$ is satisfied before G reaches a state that does not belong to Q_U. Note that all sequences s that reach a state in Q_U are in the form of $s = s'\sigma$, $\sigma \in \Sigma$, except when the initial state $\{q_0\} \in Q_U$. In this case, it is necessary also to consider sequence $s = \varepsilon$. In the sequel, we present an example of a property transparent system according to Definition 2.

Example 1: Consider automaton G shown in Figure 3, where $\Sigma_o = \{a,b\}$ and $\Sigma_{uo} = \{\sigma_u\}$. Let us consider that this system transmits events a and b to the receiver that must identify when property $\mathscr{P}$ is satisfied. In this system,

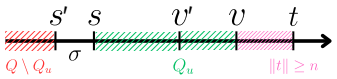

Fig. 2: Graphical interpretation of Definition 2.

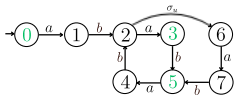

Fig. 3: Automaton G of Example 1.

we consider that states 0, 3, and 5 satisfy $\mathscr{P}$ and thus $Q_U = \{0,3,5\}$. We say that G is transparent according to Definition 2 if, after reaching a state $q_U \in Q_U$, the legitimate receiver estimates a subset of states of Q_U before G reaches a state $q \in Q_{\neg U}$. By analyzing the observer of G, $Obs(G,q_0)$, depicted in Figure 4, it is possible to see that when the state estimate of the system is $\{0\}$ or $\{5\}$, the receiver is certain that $\mathscr{P}$ is satisfied. Note that when the system reaches state 3, the receiver is not able to know if the system is in state $3 \in Q_U$ or $7 \in Q_{\neg U}$, according to $Obs(G,q_0)$. However, when the receiver state estimate corresponds to $\{3,7\}$, the only feasible event is b that takes $Obs(G,q_0)$ to state $\{5\}$, a state estimate where $\mathscr{P}$ is satisfied. Thus, in this example, when the state estimate is $\{3,7\}$, there are only two possibilities after event b is observed: (i) the system was in state 3 and reached state 5, which guarantees that the receiver can be assured that $\mathscr{P}$ was satisfied in state 3 and continued true after reaching state 5; or (ii) the system was in state 7 and reached state 5, that satisfies $\mathscr{P}$. Therefore, G is transparent with respect to P_o and Q_U. It is important to remark that this system is not "utility-ensured" according to Definition 1 [16], since when the system reaches state 3, the receiver cannot know if G is in state 3 or 7. □

Remark 1: Note, as illustrated in Example 1, that Definition 2 is more permissive than Definition 1, introduced in [16]. This is due to the fact that, in this paper, the receiver is concerned with knowing when a given property $\mathscr{P}$ is satisfied before the system reaches a state that does not satisfy $\mathscr{P}$. In addition, Definition 2 does not establish any difference between useful states belonging to Q_U. □

It is also important to remark that Definition 2 can be used for several practical problems where the receiver must only know if property $\mathscr{P}$ is satisfied. For example, considering the problem of fault diagnosis, the receiver is only interested in knowing if a fault event has occurred. Thus, it suffices to determine if the system has reached a faulty state. We also explore an application of Definition 2 to the fault diagnosis case in Section V.

In the following section, we present an algorithm to verify whether a system G is transparent according to Definition 2.

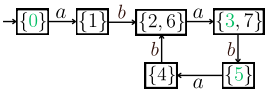

Fig. 4: $Obs(G,q_0)$ of Example 1.

IV. Verification of Property-Based Transparency

Algorithm 1: Property-Based Transparency verification

Input: $G = (Q, \Sigma, f, q_0)$, Σ_o, Q_U
Output: $T \in \{True, False\}$

1 Compute $G_{obs} = Obs(G, q_0) = (Q_{obs}, \Sigma_o, f_{obs}, q_{0,obs})$
2 Compute $V_T = (Q_V, \Sigma, f_V, q_{0,V}) = G \parallel G_{obs}$
3 Define the empty lists $TL = [\,]$ and $VL = [\,]$
4 **if** $q_0 \in Q_U$ **then**
5 add $q_{0,V}$ to TL and VL
6 **for** $(q, q_{obs}) \in Q_V$ **do**
7 **if** $q \notin Q_U$ **then**
8 **for** $\sigma \in \Gamma_G(q)$ **do**
9 **if** $f(q, \sigma) \in Q_U$ **then**
10 **if** $f_V((q, q_{obs}), \sigma) \notin VL$ **then**
11 add $f_V((q, q_{obs}), \sigma)$ to TL and VL
12 **while** $TL \neq [\,]$ **do**
13 Remove the first state, $(q_t, q_{t,obs})$, from TL
14 **if** $q_{t,obs} \cap Q_{\neg U} \neq \emptyset$ **then**
15 Verify the existence of a cyclic path $p = (q_{V_1}, \sigma_1, q_{V_2}, \sigma_2, \ldots, q_{V_k}, \sigma_k, q_{V_1})$ in V_T, where $q_{V_i} = (q_i, q_{obs,i})$, such that $q_{V_1} = (q_t, q_{t,obs})$, satisfying the conditions: $\forall i = 1, \ldots, k$, $(q_i \in Q_U) \wedge (q_{obs,i} \cap Q_{\neg U} \neq \emptyset)$
16 **if** $\exists p$ *in* V_T **then**
17 $T \leftarrow False$
18 Stop the Algorithm
19 **for** $\sigma \in \Gamma_G(q_t)$ **do**
20 **if** $f(q_t, \sigma) \in Q_{\neg U}$ **then**
21 $T \leftarrow False$
22 Stop the Algorithm
23 **else**
24 **if** $f_V((q_t, q_{t,obs}), \sigma) \notin VL$ **then**
25 add $f_V((q_t, q_{t,obs}), \sigma)$ to TL and VL
26 $T \leftarrow True$

The Property-Based Transparency of a system G can be verified using Algorithm 1, where the idea is to compare the observation of sequences of L that reach states of Q_U with the ones that reach states in $Q_{\neg U}$. To do so, in Line 1 of Algorithm 1, the observer automaton of G, G_{obs}, is computed. In Line 2, the PT verifier automaton $V_T = G \| G_{obs}$ is computed. Note that since $V_T = (Q_V, \Sigma, f_V, q_{0,V})$ is obtained by the parallel composition between the plant G and its observer G_{obs}, V_T maps all sequences of events that have the same projection while recording the exact state reached by each sequence of L in its first coordinate. In addition, each state $q_V \in Q_V$ has the form $q_V = (q, q_{obs})$, where $q \in Q$ and $q_{obs} \in Q_{obs}$. Therefore, each state $q_V \in Q_V$ of V_T has two components, where, for all $s \in L$, the first one represents the state q of G reached after s, $q = f(q_0, s)$, and the second component represents the state estimate q_{obs} after the observation $P_o(s)$. In Line 3, two lists are initialized: (i) TL stores all states of the verifier V_T that need to be tested to verify the transparency of the system and (ii) VL stores already tested states to avoid retesting.

The first states $q_V = (q, q_{obs})$ of V_T that need to be tested are added to lists TL and VL in lines 4-11. In lines 4-5, the initial state of V_T, $q_{0,V} = (q_0, q_{0,obs})$, is added to TL and VL if $q_0 \in Q_U$. In lines 6-11, all states $q'_V = (q', q'_{obs}) \in Q_V$ such that $f_V((q, q_{obs}), \sigma) = q'_V$, where $q \in Q_{\neg U}$ and $q' \in Q_U$ are added to TL and VL. In other words, we are interested in testing all states of V_T whose first coordinate belongs to Q_U that is reached by a state of V_T whose first coordinate does not belong to Q_U.

After filling lists TL and VL, we test these states using a loop in lines 12-25. In Line 14, we test if the first element of list TL is a state q_V whose second coordinate contains only states that belong to Q_U. If this is the case, this state is removed from the list TL; otherwise, lines 15-25 are executed. In lines 15-18, we verify if a cyclic path $p = (q_{V_1}, \sigma_1, q_{V_2}, \sigma_2, \ldots, q_{V_k}, \sigma_k, q_{V_1})$ exists in V_T with the following characteristics: (i) p starts from the testing state of Line 13, (ii) the first coordinate of all states of p belongs to Q_U, and (iii) the second coordinate of all states of p, which corresponds to the state estimates of the receiver, have states belonging to both Q_U and $Q_{\neg U}$. If p exists in V_T, Algorithm 1 declares that system G is not transparent in Line 17 since the existence of p means that there is a cycle in G starting from state q_t composed only of states belonging to Q_U with the same projection as sequences of events that reach states in $Q_{\neg U}$. If p exists, there is at least one arbitrarily long sequence where the receiver cannot distinguish if property $\mathscr{P}$ is satisfied. In lines 19-22, we verify if, from the current testing verifier state q_V, corresponding to a state in Q_U, the system evolves to a state in $Q_{\neg U}$. If this is the case, then the receiver cannot attest that property $\mathscr{P}$ is satisfied before the system reaches a state where $\mathscr{P}$ is false and T is declared to be False. Otherwise, in lines 23-25, the states that can be reached immediately after q_V are added to the list TL to be tested.

Example 2: Consider again automaton G presented in Figure 3, where $\Sigma_o = \{a, b\}$ and $Q_U = \{0, 3, 5\}$. The TP verifier automaton for G, V_T, computed according to Algorithm 1, is depicted in Figure 5. Since property $\mathscr{P}$ is valid in the initial state of G, the initial state of V_T is added to lists $TL = VL = [0, \{0\}]$ in Line 5 of Algorithm 1. Then, in lines 6-11, all states of V_T whose first coordinate is an element of Q_U that are reached by states with the first coordinate in $Q_{\neg U}$ are added to lists TL and VL. Thus, after Line 11 of Algorithm 1, lists TL and VL are updated to $TL = VL = [0, \{0\}; 3, \{3, 7\}; 5, \{5\}]$, since states $3, \{3, 7\}$ and $5, \{5\}$ are

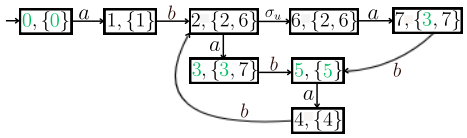

Fig. 5: Verifier automaton V_T of Example 2.

reached by states $2,\{2,6\}$ and $7,\{3,7\}$, respectively, and $2 \in Q_{\neg U}$ and $7 \in Q_{\neg U}$. Now, Algorithm 1 enters the loop of lines 12-25, where all states of list TL are tested. Note that the verification made in Line 14 is only carried out for $3,\{3,7\}$ of list TL, since $\{3,7\} \cap Q_{\neg U} = \{7\} \neq \emptyset$. In this case, there is no cyclic path p with the characteristics presented in Line 15 of Algorithm 1 starting from state $3,\{3,7\}$ of V_T. Therefore, the loop of lines 19-25 is executed since $\Gamma_G(3) = \{b\}$, and $f(3,b) = 5 \in Q_U$. In this case, lines 21-22 are not executed. In Line 24, note that $f_V((3,\{3,7\}),b) = 5,\{5\}$, which belongs to $VL = [0,\{0\}; 3,\{3,7\}; 5,\{5\}]$. Thus, state $5,\{5\}$ is not added to list TL, which is now empty, and the system is declared transparent in Line 26. □

Theorem 1: Let G be the automaton that models the plant, Σ_o the set of observable events, and Q_U the set of states in which property $\mathscr{P}$ is satisfied. Then, G is property-based transparent with respect to P_o and Q_U if, and only if, $T = True$ according to Algorithm 1.

Proof: Since the verifier V_T is computed from the parallel composition of the plant and its observer, each state $(q_t, q_{t,obs}) \in V_T$ stores, in the first coordinate q_t, the state of the plant reached after sequence $s \in L$ and, in the second coordinate $q_{t,obs}$, the state estimate after the observation of s, $P_o(s)$. If, for a given transition $f(q_t,\sigma) = q'_t$ of the plant, such that $q_t \in Q_{\neg U}$, $\sigma \in \Sigma$, and $q'_t \in Q_U$, then exists a sequence $s = s'\sigma$ that reaches a useful state, and the state of the verifier whose first coordinate is q'_t will be added to the test list, TL, at Line 11 of Algorithm 1. In addition, if q_0 is useful, then the first state of the verifier V_T is also added to TL at Line 5 of the algorithm. According to Definition 2, for each sequence s that reaches a state added to list TL, two conditions must hold: (i) the post-language L/s cannot have a sequence that leaves the set of useful states before being certain that the estimation is a subset of the useful states, and (ii) after a finite number of event occurrences, the receiver must estimate that the system is in a subset of the useful states. In this regard, TL, before Line 17 of the algorithm, is composed of all states of the verifier reached by sequences $s = s'\sigma$ such that $f(q_0,s') \in Q_{\neg U}$ and $f(q_0,s) \in Q_U$. In Line 25 of Algorithm 1, states $(q'_t, q'_{t,obs})$, where $q'_t \in Q_U$ and $q'_{t,obs} \cap Q_{\neg U} \neq \emptyset$, such that $f_V((q_t,q_{t,obs}),\sigma) = (q'_t, q'_{t,obs})$, $q_t \in Q_U$ are added to TL. The rest of the proof is divided into two proofs, representing the necessary and sufficient conditions.

($\Rightarrow$) This proof is done by contrapositive. Let us suppose that $T = False$ according to Algorithm 1. Therefore, Algorithm 1 executes either (i) Line 17 or (ii) Line 21. In case (i) Line 17 occurs, there exists a path $p = (q_1,\sigma_1,q_2,\sigma_2,\ldots,\sigma_n,q_1)$ in V_T starting in a state in TL, such that all first coordinates of the states of p belong to Q_U and the second coordinate have states in Q_U and $Q_{\neg U}$. Therefore, there exists a sequence $t = \sigma_1\sigma_2\ldots\sigma_k$ that can be repeated indefinitely after sequence s whose observation does not lead to an estimate with only states of Q_U, *i.e.*, $\exists \omega v \in L : f(q_0,\omega v) \in Q_{\neg U}$ and $P_o(\omega v) = P_o(st^*)$. Since p is cyclic, there does not exist a $n \in \mathbb{N}$ that satisfies the conditions in Definition 2, and therefore, the system is not transparent. On the other hand, in case (ii), Line 21 occurs only if there exists an event $\sigma \in \Sigma$ such that $f(q_0,st\sigma) \in Q_{\neg U}$ and there exists a sequence ω such that $f(\omega) \in Q_{\neg U}$ and $P_o(st) = P_o(\omega)$, and therefore, according to Definition 2 the system is not transparent.

($\Leftarrow$) If Algorithm 1 returns True, then neither of the conditions in Line 17 nor Line 21 have occurred, and therefore, $\forall st$ such that $s = s'\sigma$, $f(q_0,s') \in Q_{\neg U}$, $f(q_0,s) \in Q_U$, $t \in \Sigma^*$ where, $\|t\| > n$, where n is, in the worst case, equals to the length of the list of visited states VL. Then, some state in the visited states has an estimate of only states in Q_U for all ramifications of the sequence s. Otherwise, this ramification would be included in VL and TL in Line 31 of Algorithm 1. Therefore, there exists a prefix of t such that all prefixes v' of v are such that $f(q_0,sv') \in Q_U$ and the estimate of the system after observing sv is a subset of Q_U, *i.e.* $\forall \omega \in L_{\neg U}$, $P_o(sv) \neq P_o(\omega)$. Thus, according to Definition 2, the system is transparent. ■

A. Computational complexity

Note that Algorithm 1 computes the observer G_{obs} of the system G, and composes G with G_{obs} to obtain the verifier automaton V_T for Property-Based Transparency. Therefore, the number of states and transitions of V_T is, in the worst case, equal to $2^{|Q|} \times |Q|$ and $2^{|Q|} \times |Q| \times |\Sigma|$, respectively. After the computation of V_T, all its states can be visited in Algorithm 1 using operations that are linear in the number of states and transitions of V_T. Thus, Algorithm 1 has computational complexity $O(2^{|Q|} \times |Q| \times |\Sigma|)$.

V. Applications of Property-Based Transparency

This section shows two possible application domains for which Property-Based Transparency can be used: (i) current-state opacity and (ii) fault diagnosis.

A. PT in the context of current-state opacity

Let us consider that the communication channel between the plant and the supervisor can be eavesdropped, as illustrated in Figure 6. In this scenario, we assume that the attacker has full knowledge of the system model and can observe all observable event occurrences after starting the attack. We also assume that the attack can be initiated any time after the system has been initialized. Thus, when the attack begins, the attacker must estimate the system's current state based on his/her observations. At the same time, the receiver starts to observe the system from its beginning of functioning, following its behavior from the initial state. In this context, we want to verify if the system is transparent

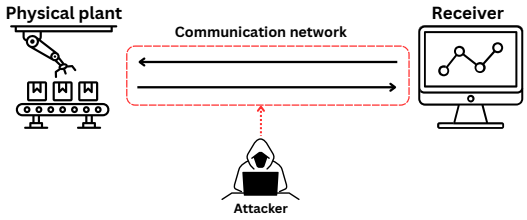

Fig. 6: Eavesdropping attack.

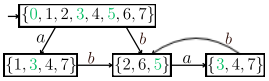

Fig. 7: Estimator automaton $\mathscr{E} = Obs(G,Q)$ of G.

and Current-State Opaque (CSO), according to the following definition [6], [9], [5].

Definition 3 (Current-State Opacity): Given a system $G = (Q, \Sigma, f, Q_0)$, a projection P_o, and a set of secret states $Q_S \subseteq Q$, then G is current-state opaque if

$$\begin{aligned} &\forall q_i \in Q_0 \quad \text{and } \forall s \in L(G,q_i) \text{ such that } f(q_i,s) \in Q_S, \\ &\exists q_j \in Q_0 \quad \text{and } \exists s' \in L(G,q_j) \text{ such that} \\ &\qquad f(q_j,s') \in Q \setminus Q_S \text{ and } P_o(s) = P_o(s'). \end{aligned}$$

□

Note that, in Definition 3, the initial state of G can be a set of states and even equal to $Q_0 = Q$. In [20], a method to verify CSO is presented. The method is based on the state estimator automaton $\mathscr{E} = Obs(G,Q)$, obtained considering Q as the set of initial states. A system G is CSO if no state of $\mathscr{E}$ is a subset of Q_S.

Let us consider again automaton G depicted in Figure 3 and let $Q_S = Q_U = \{0,3,5\}$. The estimator $\mathscr{E} = Obs(G,Q)$ for this system is presented in Figure 7. Note that no state of $\mathscr{E}$ is a subset of set Q_S; thus, the system is CSO if the attacker does not know the system's current state when he/she starts the attack. It is important to remark that G is transparent considering the same set of secret states as useful (states where property $\mathscr{P}$ is valid), which means that, for this system, the same information required for the legitimate receiver is opaque to an attacker. Note that in almost all works presented in the literature, the set of secret states and useful states are disjoint, except when some encryption is used [21]. This restriction is no longer needed in the approach considered in this paper.

It is also important to remark that the notion of PT is not the opposite of CSO even when considering the same initial state for both the receiver and the attacker. To show this fact, let us consider the plant automaton presented in Figure 8, where $\Sigma_o = \{a,b\}$ and $\Sigma_{uo} = \{\sigma_u\}$, and let us consider that the secret and useful states is $\{1\}$, *i.e.*, $Q_S = Q_U = \{1\}$. Note that, after observing event a, the external observer is certain that the system's current state is $\{1\}$, and thus, according to Definition 3, G is not CSO. In addition, note that after the occurrence of sequence $s' = ab$, the system is in state 2, and after sequence $s = s'\sigma = aba$, the system

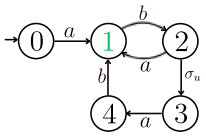

Fig. 8: Automaton G.

is in state 1. Therefore, according to Definition 2, if there exists a sequence $t \in L/s$ such that $f(q_0,st) \notin Q_U$, and for all prefixes v of t such that $f(q_0,sv) \in Q_U$, there exists $\omega \in L_{\neg U}$, $P_o(sv) = P_o(\omega)$, then G is not transparent. Note that, in this case, if $t = b$, then $st = abab$ leads the system to state $\{2\} \notin Q_U$, and the receiver is not able to infer with certainty that state $\{1\} \in Q_U$ was reached. Therefore, G is not transparent.

B. PT in the context of fault diagnosis

In the fault diagnosis classical approach [22], the receiver is interested in identifying the occurrence of an unobservable fault event based on its observations of the system behavior. Thus, we can consider that the property of interest $\mathscr{P}$ is that the receiver must know whether a fault event has occurred in the system. Therefore, $\mathscr{P}$ is true for all states reached after the fault event. Since there may exist states in G that are reached after the occurrence of a fault event and also by a sequence of non-faulty events, it is necessary to distinguish the states according to which sequences have been executed by the system. In order to do so, let us consider automaton A_ℓ presented in Figure 9. Note that the states of $G_\ell = (Q_\ell, \Sigma, f_\ell, q_{0,\ell}) = G \| A_\ell$ [19], are labeled with N if the state is reached after the occurrence of a non-faulty sequence of events, and Y if the state has been reached after the occurrence of a fault event. Thus, the set of useful states can be defined as $Q_U = \{q_\ell \in Q_\ell : q_\ell = (q,Y)\}$, *i.e.*, the set of faulty states.

To compare Property-Based Transparency with the diagnosability of a system language, we recall the definition of fault diagnosability presented in [23], where L_N is the fault-free language of the system, and $\Sigma_f \subset \Sigma$ is the set of fault events, as follows.

Definition 4 (Diagnosability): The language generated by G, L, is said to be diagnosable with respect to projection $P_o : \Sigma^\star \to \Sigma_o^\star$ and Σ_f if

$$\begin{aligned} &(\exists z \in \mathbb{N})(\forall s \in L \setminus L_N)(\forall st \in L \setminus L_N) \\ &(\|t\| \geq z \Rightarrow P_o(st) \notin P_o(L_N)) \end{aligned}$$

□

Comparing Definitions 2 and 4, we can make the following considerations:

1) Language $L_{\neg U}$ is formed of all sequences of L that do not contain a fault event, *i.e.*, $L_{\neg U} = L_N$;
2) Sequence st can be written as $s = s'\sigma_f t$, where $\sigma_f \in \Sigma_f$ is a fault event;
3) All states reached after $\sigma_f \in \Sigma_f$ in G_ℓ belong to Q_U, thus, once a state of Q_U is reached, the system cannot reach a state in $Q_{\neg U}$.

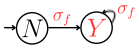

Fig. 9: Automaton A_ℓ.

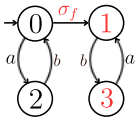

Fig. 10: Automaton G with a fault event.

Assuming these considerations, Definition 4 can be interpreted as a particular case of Definition 2. In other words, if the property of interest $\mathscr{P}$ is the fault event occurrency, diagnosability can be written as a Property-Based Transparency with respect to the system language. Then, if the system G is PT, its generated language L is diagnosable.

Let us consider the plant automaton presented in Figure 10. By making the parallel composition between G and A_ℓ, we obtain automaton G_ℓ depicted in Figure 11. In this case, set $Q_U = \{1Y;3Y\}$, and the observer of G_ℓ, $Obs(G_\ell)$, is presented in Figure 12. Note that, according to $Obs(G_\ell)$, the receiver has no doubt the fault has occurred after the observation of event b, *i.e.*, L is diagnosable with respect to Σ_f and P_o. Therefore, G is transparent with respect to Q_U and P_o.

VI. Conclusions

In this paper, we introduce a new definition of utility called Property-Based Transparency (PT). It is based on the receiver's ability to identify when a given property is satisfied before the system reaches a state where it is no longer valid. We compare the notion of PT with the "utility-ensured" property proposed in [16] and show that PT is more general and can be applied to a wider variety of systems. A verification method is presented to test if a given system with a set of states that satisfy a property of interest is transparent. We also present two scenarios where PT is relevant to the intended receiver: a fault diagnosis and an opacity case. We are currently investigating obfuscation techniques to satisfy the Property-Based Transparency while keeping useful information private for an unauthorized agent.

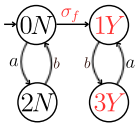

Fig. 11: Automaton G_ℓ.

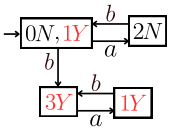

Fig. 12: Obs(G_ℓ).

References

[1] C. Gao, C. Seatzu, Z. Li, and A. Giua, "Multiple attacks detection on discrete event systems," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 2352–2357.

[2] R. Meira-Góes, E. Kang, R. H. Kwong, and S. Lafortune, "Synthesis of sensor deception attacks at the supervisory layer of cyber–physical systems," *Automatica*, vol. 121, p. 109172, 2020.

[3] C. Dwork, "Differential privacy," in *International Conference on Automata, Languages and Programming*, 2006.

[4] Y. C. Wu, V. Raman, B. C. Rawlings, S. Lafortune, and S. A. Seshia, "Synthesis of Obfuscation Policies to Ensure Privacy and Utility," *Journal of Automated Reasoning*, vol. 60, no. 1, pp. 107–131, 2018.

[5] S. Lafortune, F. Lin, and C. N. Hadjicostis, "On the history of diagnosability and opacity in discrete event systems," *Annual Reviews in Control*, vol. 45, pp. 257–266, 2018.

[6] A. Saboori and C. N. Hadjicostis, "Notions of security and opacity in discrete event systems," *Proceedings of the IEEE Conference on Decision and Control*, pp. 5056–5061, 2007.

[7] A. Saboori and C. N. Hadjicostis, "Verification of initial-state opacity in security applications of discrete event systems," *Information Sciences*, vol. 246, 2013.

[8] P. C. Mayer, F. G. Cabral, P. M. Lima, and M. V. Moreira, "Current-state opacity based on state outputs," *IFAC-PapersOnLine*, 2024, 17th IFAC Workshop on Discrete Event Systems.

[9] R. Jacob, J.-J. Lesage, and J.-M. Faure, "Overview of discrete event systems opacity: models, validation, and quantification," *Annual Reviews in Control*, vol. 41, pp. 135–146, 2016.

[10] S. Oliveira, A. B. Leal, M. Teixeira, and Y. K. Lopes, "A classification of cybersecurity strategies in the context of discrete event systems," *Annual reviews in Control*, 2023.

[11] F. Lin, "Opacity of discrete event systems and its applications," *Automatica*, vol. 47, no. 3, pp. 496–503, 2011.

[12] Y. Ji, Y.-C. Wu, and S. Lafortune, "Enforcement of opacity by public and private insertion functions," *Automatica*, vol. 93, pp. 369–378, 2018.

[13] X. Li, C. N. Hadjicostis, and Z. Li, "Opacity enforcement in discrete event systems using extended insertion functions under inserted language constraints," *IEEE Transactions on Automatic Control*, 2023.

[14] A. Winterberg, S. Lafortune, M. Blischke, and O. Necmiye, "A dynamic obfuscation framework for security and utility," *13th International Conference on Cyper-Physical Systems*, 2022.

[15] J. M. C. Cardoso, M. V. Moreira, and L. K. Carvalho, "Synthesis of an obfuscation policy that guarantees utility satisfying a new privacy criterion," in *IFAC-PapersOnLine*. Elsevier, 2023.

[16] R. J. Barcelos and J. C. Basilio, "Ensuring utility while enforcing current-state opacity," in *IFAC World Congress*, vol. 56, no. 2, 2023, pp. 4595–4600.

[17] M. E. Simon, F. L. Baldissera, M. H. de Queiroz, and F. G. Cabral, "Multi-robots coordination system for urban search and rescue assistance based on supervisory control theory," *Journal of Control Automation Electrical Systems*, vol. 34, p. 484–495, 2023.

[18] P. C. Mayer, F. G. Cabral, and M. V. Moreira, "A protocol for decentralized synchronous diagnosis with coordination," *Control Engineering Practice*, vol. 141, p. 105732, 2023.

[19] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event System*. Secaucus, NJ: Springer-Verlag New York, Inc., 2008.

[20] A. Saboori and C. Hadjicostis, "Verification of k-step opacity and analysis of its complexity," *IEEE Transactions on Automation Science and Engineering*, vol. 8, pp. 549–559, 2011.

[21] P. M. Lima, L. K. Carvalho, and M. V. Moreira, "Ensuring confidentiality of cyber-physical systems using event-based cryptography," *Information Sciences*, vol. 621, pp. 119–135, 2023.

[22] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Transactions on automatic control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[23] ——, "Failure diagnosis using discrete-event models," *IEEE Transactions on Control Systems Technology*, vol. 4, no. 2, pp. 105–124, 1996.