Opacity of Discrete Event Systems: models, validation and quantification

Romain Jacob, Jean-Jacques Lesage, Jean-Marc Faure

Opacity of Discrete Event Systems: models, validation and quantification

Romain Jacob *, **Jean-Jacques Lesage** *, **Jean-Marc Faure** *

* *LURPA, ENS Cachan, Univ Paris-Sud, F-94235 Cachan, France (e-mail: romain.jacob@ens-cachan.fr).*

Abstract: Over the last decade, opacity of discrete event systems (DES) has become a very fertile field of research. Driven by safety and privacy concerns in network communications and online services, much theoretical work has been conducted in order to design opaque systems. A system is opaque if an external observer in unable to infer a "secret" about the system behavior. This paper aims to review the most commonly used techniques of opacity validation for deterministic models and opacity quantification for probabilistic ones. Available complexity results are also provided. Finally, we review existing tools for opacity validation and current applications.

Keywords: Opacity, Discrete event systems, Validation, Verification, Enforcement Quantification, Secrecy, Privacy, Security

1. INTRODUCTION

Online services and network communications have become ubiquitous over the past 30 years. This evolution in our everyday life brought along new preoccupations regarding security and privacy. Despite continuously releasing tons of information about everything we do and think, we still want some information to remain secret. Thus, a new problem has arisen in computer science, called *Information Flow*. It characterizes the (possibly illegal and indirect) transmission of secret data from a high level user to a low level one. Various information flow properties have been defined in the literature: anonymity, non-interference, secrecy, privacy, security and opacity.

In this paper, we focus on *opacity*, which characterizes whether a given "secret" about a system behavior is hidden or not from an external observer, further called the intruder. It is assumed the intruder has full knowledge of the system's structure but only partial observability. Based on its observations, the intruder constructs an estimate of the system's behavior. The secret is said to be opaque if the intruder's estimate never reveals the system's secret. Specifically, the system is opaque if for any secret behavior, there exists at least one other non-secret behavior that looks the same to the intruder.

Opacity is a rather recent field of research. It was first introduced in 2004 in computer science to analyze cryptographic protocols (Mazaré (2004)). It reached the discrete event systems (DES) community with the work of Bryans et al. (2005) which investigated opacity in systems modeled as Petri nets. Secrets were set as predicates over Petri net markings (i.e., states). In Bryans et al. (2008), previous work was extended by investigating opacity in labeled transition systems (LTS), in which secrets were defined as predicates over runs. More recently, Saboori and Hadjicostis investigated state-based opacity properties using finite-state automata (FSA) models (Saboori and Hadjicostis (2007), Saboori (2011)). Many researchers have considered the validation of opacity properties which spans from system's opacity verification (Hadjicostis and Keroglou (2014)) to the synthesis of a controller/scheduler which assures opacity, either through supervision (Dubreil (2010)) or enforcement (Falcone et al. (2014)). Recent work has been conducted on quantifying a system's opacity (Bérard et al. (2010)). Indeed, opacity validation w.r.t. a given secret and intruder is a yes/no question for deterministic models. However, many systems are probabilistic. In that case, one might be interested in quantifying the possible information leakage from the system.

This paper aims to provide a comprehensive and general review of opacity related work considering DES models. After introducing relevant notations in Section 2, we synthesize different notions of opacity used in the literature in Section 3. Section 4 reviews validation methods of various opacity properties. Extensions to probabilistic models are presented in Section 5. Section 6 summarizes decidability and complexity of most approaches surveyed in this paper. Finally, applications of opacity in DES are presented in Section 7 and Section 8 concludes the paper.

2. PRELIMINARIES

Let E be an alphabet of labels. E^* is the set of all finite strings composed of elements of E, including the empty string ε. A language $L \subseteq E^*$ is a set of finite-length strings of labels in E. For any string t, $|t|$ denotes the length of t. For a string ω, $\overline{\omega}$ denotes the prefix-closure of ω and is defined as $\overline{\omega} = \{t \in E^* | \exists s \in E^*, ts = \omega\}$. The post-string ω/s of ω after s is defined as $\omega/s = \{t \in E^*, st = \omega\}$.

A finite-state automaton $G = (X, E, f, X_0)$ is a 4-tuple composed of a finite set of states $X = \{0, 1, ..., N-1\}$, a finite set of events E, a partial state transition function $f : X \times E \to X$ and a set of initial states X_0. The function f is extended to domain $X \times E^*$ in the usual manner. The language generated by the system G describes the system's behavior and is defined by

$\mathcal{L}(G, X_0) := \{s \in E^* | \exists i \in X_0, f(i, s) \text{ is defined}\}$; it is prefix-closed by definition.

Note that in opacity problems, the initial state need not to be known *a priori*, therefore the *set* of initial states instead of a single initial state. We also consider partially observable systems. The event set is partitioned into an observable set E_o and an unobservable set E_{uo}. Given a string $t \in E^*$, its observation is the output of the natural projection function $P : E^* \to E_o^*$, which is recursively defined as $P(te) = P(t)P(e)$ where $t \in E^*$ and $e \in E$. The projection of an event $P(e) = e$ if $e \in E_o$, while $P(e) = \varepsilon$ if $e \in E_{uo} \cup \{\varepsilon\}$. Finally, for a language $J \subseteq E^*$, the inverse projection is defined as $P^{-1}(J) = \{t \in E^* : P(t) \in J\}$.

3. OPACITY OF DISCRETE-EVENT SYSTEMS

In this section, we formalize different opacity properties of DES. In the general case, the intruder is assumed to have full knowledge of the system structure (plus eventually the system's controller) but has only partial observability over it. Opacity is parameterized by a secret predicate S and by the intruder's observation mapping P over the system's executions. A system is opaque w.r.t. S and P if, for any secret run in S, there is another run not in S which is observably equivalent.

In cases of DES models, the secret predicate S can be of two classes: a subset of executions (or parts of executions) or a subset of states. This classifies opacity properties into two families: *language-based opacity* and *state-based opacity*.

3.1 Language-based opacity – LBO

LBO has been formalized in different ways in the literature. It was first introduced in Badouel et al. (2007) and Dubreil et al. (2008). LBO (also referred to as trace-based opacity) is defined over a secret behavior described by a language $L_S \subseteq E^*$. The system is opaque w.r.t. L_S and the projection map P if no execution leads to an estimate that is completely inside the secret behavior. Alternatively, in Lin (2011), LBO is defined over two sublanguages of the system, $(L_1, L_2) \subseteq (\mathcal{L}(G, X_0))^2$. Sublanguage L_1 is opaque w.r.t. L_2 and any observation mapping θ if the intruder confuses every string in L_1 with some strings in L_2 under θ. In most recent papers considering LBO, the latter definition is used except for the observation mapping θ which is generally the natural projection mapping P.

Definition 1 (LBO – Strong Opacity)**:** *Given a system $G = (X, E, f, X_0)$, a projection P, a secret language $L_S \subseteq \mathcal{L}(G, X_0)$, and a non-secret language $L_{NS} \subseteq \mathcal{L}(G, X_0)$, G is language-based opaque if for every string $t \in L_S$, there exists another string $t' \in L_{NS}$ such that $P(t) = P(t')$.*
Equivalently, $L_S \in P^{-1}[P(L_{NS})]$.

The system is language-based opaque if for any string t in the secret language L_S, there exists at least one other string t' in the non-secret language L_{NS} with the same projection. Therefore, given the observation $s = P(t) = P(t')$, intruders cannot conclude whether the secret string t or the non-secret string t' has occurred. Note that L_S and L_{NS} do not need to be prefix-closed in general, nor even regular.

Part of the literature refers to Definition 1 as *strong opacity*. In Lin (2011), a smoother opacity property is also defined.

Definition 2 (LBO – Weak Opacity)**:** *Given a system $G = (X, E, f, X_0)$, a projection P, a secret language $L_S \subseteq \mathcal{L}(G, X_0)$, and a non-secret language $L_{NS} \subseteq \mathcal{L}(G, X_0)$, G is weakly opaque if for some string $t \in L_S$, there exists another string $t' \in L_{NS}$ such that $P(t) = P(t')$.*
Equivalently, $L_S \cap P^{-1}[P(L_{NS})] \neq \emptyset$.

The system is weakly opaque if some strings in L_S are confused with some strings in L_{NS}.

As a consequence, the property of *no opacity* can be easily defined.

Definition 3 (LBO – No opacity)**:** *L_S is no opaque w.r.t. L_{NS} and P if L_S is not weakly opaque w.r.t. L_{NS} and P.*
Equivalently, $L_S \cap P^{-1}[P(L_{NS})] = \emptyset$.

Remark 1. It is shown in Ben-Kalefa and Lin (2009) that LBO properties are closed under union, but may not be closed under intersection. They further discuss how to modify languages to satisfy the strong, weak, and no opacity by investigating sublanguages and superlanguages.

Example 1. From Wu and Lafortune (2013) – Consider the system G in Fig. 1(a) with $E_o = \{a, b, c\}$. It is language-based opaque when $L_S = \{abd\}$ and $L_{NS} = \{abc^*d, adb\}$ because whenever the intruder sees $P(L_S) = \{ab\}$, it is not sure whether string *abd* or string *adb* has occurred. However, this system is not language-based opaque if $L_S = \{abcd\}$ and $L_{NS} = \{adb\}$; no string in L_{NS} has the same projection as the secret string *abcd*.

3.2 State-based opacity – SBO

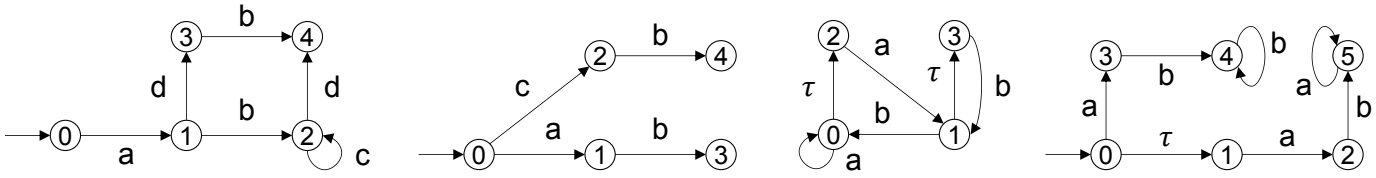
The state-based approach for opacity of DES was introduced in Bryans et al. (2005) (for Petri nets models) then extended to FSA in Saboori and Hadjicostis (2007). The state-based approach relates to the intruder ability to infer that the system is or has been in a given "secret" state or set of states. Depending on the nature of the secret set, different opacity properties have been defined.

Current-State Opacity – CSO CSO was first introduced in Bryans et al. (2005) and called *final opacity*, in the context of Petri nets. The definition was then adapted to LTS in Bryans et al. (2008), and further developed in finite state automata models in Saboori and Hadjicostis (2007). A system is CSO if the observer can never infer, from its observations, whether the current state of the system is a secret state or not.

Definition 4 (CSO)**:** *Given a system $G = (X, E, f, X_0)$, a projection P, a set of secret states $X_S \subseteq X$, and a set of non-secret states $X_{NS} \subseteq X$, G is current-state opaque if $\forall i \in X_0$ and $\forall t \in \mathcal{L}(G, i)$ such that $f(i, t) \in X_S$, $\exists j \in X_0$, $\exists t' \in \mathcal{L}(G, j)$, such that $f(j, t') \in X_{NS}$ and $P(t) = P(t')$.*

The system is CSO if for every string t that leads to a secret state, there exists another string t' leading to a non-secret state whose projection is the same. As a result, the intruder can never assert with certainty that the system's current state is in X_S.

Remark 2. In Bryans et al. (2005), the property of *always-opacity* is also introduced. A system is always-opaque (or *total-opaque* in Bryans et al. (2008)) over a set of runs if it is CSO for any state visited during these runs. This is equivalent to consider a set of secret states which lies on a prefix-closed language.



(a) Example 1 (LBO) (b) Example 2 (CSO) (c) Example 3 (ISO) (d) Example 4 (KStep)

Fig. 1. From Wu and Lafortune (2013) and Falcone and Marchand (2013) – Systems discussed in Examples 1 to 4.

Example 2. From Wu and Lafortune (2013) – Consider G in Fig. 1(b) and the sets of secret and non-secret states $X_S = \{3\}$ and $X_{NS} = X \backslash X_S$. If $E_o = \{b\}$, then G is current-state opaque because the intruder is always confused between ab and cb when observing b; that is, the intruder cannot tell if the system is in state 3 or 4.
However, if $E_o = \{a, b\}$, CSO does not hold because the intruder is sure that the system is in state 3 when observing ab.

Initial-State Opacity – ISO ISO property relates to the membership of the system's initial state within a set of secret states. The system is *initial-state opaque* if the observer is never sure whether the system's initial state was a secret state or not.

Definition 5 (ISO)**:** *Given a system $G = (X, E, f, X_0)$, a projection P, a set of secret initial states $X_S \subseteq X_0$, and a set of non-secret initial states $X_{NS} \subseteq X_0$, G is initial-state opaque if $\forall i \in X_S$ and $\forall t \in \mathcal{L}(G, i)$, $\exists j \in X_{NS}$, $\exists t' \in \mathcal{L}(G, j)$, such that $P(t) = P(t')$.*

The system is ISO if for every string t that originates from a secret state i, there exists another string t' originating from a non-secret state j such that t and t' are observationally equivalent. Therefore, the intruder can never determine whether the system started from a secret state i or from a non-secret state j.

Example 3. From Wu and Lafortune (2013) – Consider G in Fig. 1(c) with $E_o = \{a, b\}$, $X_S = \{2\}$, and $X_{NS} = X \backslash X_S$. G is initial-state opaque because for every string t starting from state 2, there is another string $(\tau)t$ starting from state 0 that looks the same.
However, ISO does not hold if $X_S = \{0\}$. Whenever the intruder sees string aa, it is sure that the system originated from state 0; no other initial states can generate strings that look the same as aa.

Remark 3. Hadjicostis (2012) defines resolution of initial state w.r.t. a secret set of states S. It requires that when the system starts from a secret state, the observer will be able to eventually (i.e., after a finite sequence of events/observations) determine with certainty that the system's initial state lied within the set of secret states S. It is worth pointing out at this point that absence of resolution of initial state is necessary but not sufficient for ISO.

K-step opacity Except for ISO, these opacity properties do not consider the system behavior once it has exited a secret state. A more general problem would be to keep secret the fact the system *was* in a secret state a few steps ago. This property is called *K-step opacity* and was first introduced in Saboori and Hadjicostis (2007).

Definition 6 (K-step opacity)**:** *Given a system $G = (X, E, f, X_0)$, a projection P, and a set of secret states $X_S \in X$, G is K-step opaque w.r.t. X_S and P, or (X_S, P, K)-opaque (for $K \geq 0$), if:*

$\forall i \in X_0$, $\forall t \in \mathcal{L}(G, i)$ and $\forall t' \in \bar{t}$ such that $f(i, t') \in X_S$ and $|P(t)/P(t')| \leq K$,

$\exists j \in X_0$, $\exists s \in \mathcal{L}(G, j)$ and $\exists s' \in \bar{s}$, such that $f(j, s') \in X_{NS}$, $P(s) = P(t)$ and $P(s') = P(t')$.

This definition can be reformulate as in Falcone et al. (2014). The system is (X_S, P, K)-opaque if for every execution t of G and for every secret execution t' prefix of t with an observable difference inferior to K, there exist two executions s and s' observationally equivalent respectively to t and t' such that s' is not a secret execution (i.e., which does not bring the system in a secret state).

Example 4. From Falcone and Marchand (2013) – Consider G in Fig. 1(d) with $E_o = \{a, b\}$, $X_S = \{2\}$, and $X_{NS} = X \backslash X_S$. G is $(X_S, P, 1)$-opaque.
However, it is not $(G, P, 2)$-opaque as only $(\tau)aba$ is a compatible execution with the observation aba. After the last a has occurred, the intruder can deduce that the system was in state 2 two steps ago.

Remark 4. K-step opacity is a direct extension of CSO. CSO is equivalent to 0-step opacity (Saboori and Hadjicostis (2007)).

Remark 5. In Falcone et al. (2014), Definition 6 is referred to as *K-step weak opacity*. The property of *K-step strong opacity* holds if the system is K-step weakly opaque and there exists a trace of the system (observably equivalent to the actual execution) which does not cross any secret state over the last K steps.

Infinite-step opacity In Saboori (2011), K-step opacity has been further extended to infinite-step opacity.

Definition 7 (Infinite-step opacity)**:** *Given a system $G = (X, E, f, X_0)$, a projection P, and a set of secret states $X_S \in X$, G is infinite-step opaque w.r.t. X_S and P, or (X_S, P, ∞)-opaque, if:*

$\forall i \in X_0$, $\forall t \in \mathcal{L}(G, i)$ and $\forall t' \in \bar{t}$ such that $f(i, t') \in X_S$,

$\exists j \in X_0$, $\exists s \in \mathcal{L}(G, j)$ and $\exists s' \in \bar{s}$, such that $f(j, s') \in X_{NS}$, $P(s) = P(t)$ and $P(s') = P(t')$.

A system is infinite-step opaque if, for every execution of the system, after having observed an arbitrary long sequence of events, the intruder cannot infer the system was in a secret state at some point (at any step back in the execution).

3.3 Transformations between different opacity properties

The aforementioned opacity properties have strong connections between each another. Several works have addressed the translation between opacity properties.

Saboori (2011) adapts the language-based definition to ISO in order to apply supervisory control (refer to Section 4.2). On the contrary, Cassez et al. (2012) describes transformations from LBO to CSO. In Wu and Lafortune (2013), the authors extend these works and provide a full transformation mapping between LBO, CSO, ISO and IFO (initial-and-final state opacity – defined therein).

In addition, we already mentioned that K-step opacity is an extension of CSO. CSO is equivalent to 0-step opacity.

Finally, in Saboori (2011), a language-based translation of K-step opacity is suggested: *trace-based K-step opacity*. It is a special case of K-step opacity which, to the best of our knowledge, has never been used or considered in any other work. It is mentioned here for the sake of completeness.

3.4 Distributed opacity

Most opacity-related studies account for a single intruder. However, a few of them consider distributed notions of opacity. Hence, Badouel et al. (2007) considers multiple intruders, each of them having its own observation mapping and secret of interest. The system is said to be *concurrently opaque* if all secrets are safe. A different notion, called *joint opacity* is presented in Wu and Lafortune (2013) and Wu (2014). In this setting, several intruders collaborate through a coordinator in order to discover the same secret. Finally, Paoli and Lin (2012) considers decentralized framework with and without coordination among agents and formalizes definitions of decentralized opacity. It is shown to be an extension of co-observability, used in traditional supervisory control.

3.5 Infinite DES models

Up to a few years ago, opacity-related studies only considered finite-state DES models have been considered in . There are recent work addressing extensions to infinite-state DES. CSO and diagnosability verification are investigated for infinite-state DES modeled by pushdown automata in Kobayashi and Hiraishi (2013) (therein called pushdown systems), as well as in Chédor et al. (2014), in the more general setting of recursive tile systems.

3.6 Relation with other DES and information flow properties

We mentioned in Section 1 that opacity is an information flow property. Relations between several of these properties can be easily drawn. First of all, *secrecy* is a special case of opacity. A system predicate is *secret* if the predicate and its complement are simultaneously opaque (Badouel et al. (2007)).

It was shown in Bryans et al. (2008) that anonymity and non-interference problems may be reduced to opacity, using suitable observation functions. Lin (2011) also establishes links between opacity, anonymity and secrecy and shows that observability, diagnosability, and detectability, can be reformulated as opacity as well. The equivalence between opacity and intransitive non-interference is proven in Mullins and Yeddes (2013).

Moreover, opacity can be considered in some sense as the dual of diagnosability (Zaytoon and Lafortune (2013)): for opacity to hold, the secret should not be diagnosable from the viewpoint of the intruder. As a result, several opacity related works use techniques from diagnosis of DES; e.g., Dubreil (2010), Kobayashi and Hiraishi (2013), Chédor et al. (2014).

4. ENSURING OPACITY

Traditional opacity formulations from the literature were presented in Section 3. The following question is how does one know that a given system G is opaque w.r.t. a secret and the information available to intruders? Furthermore, if it is not, what can be done to make it opaque? These questions have been continuously addressed and this section aims to synthesize the main approaches available in the literature.

There are three main approaches to ensure opacity properties of DES:

- Verification, which roughly consists in *model-checking* opacity properties;
- Supervisory control theory (SCT), which restricts the system's behavior in order to preserve the secret;
- Enforcement, which inputs observable events of the systems and outputs (possibly) modified information to the observer, such that the secret is preserved.

The main difference between SCT and enforcement is that SCT constrains the system behavior (by restraining its output) by means of a controller while enforcement allows the system free-behavior but post-processes all its output.

4.1 Verification of opacity properties

Diagnosis approaches We mentioned in Section 3.6 that opacity can be considered as the dual property of diagnosability. Dubreil et al. (2009) investigates the use of techniques from diagnosis of DES (Zaytoon and Lafortune (2013)) to detect and predict the flow of secret information and construct a monitor that allows an administrator to detect it.

In the sequel of the paper, only control and enforcement approached will be further developed. For more details about verification of: (i) LBO, refers to Lin (2011); (ii) SBO, refers to Falcone et al. (2014) and Hadjicostis and Keroglou (2014).

4.2 Supervisory control theory – SCT

The potential use of SCT for opacity validation of DES is rather obvious. Several works present construction of minimally restrictive opacity-enforcing supervisory controllers; e.g., Takai and Kumar (2009), Saboori (2011), Ben-Kalefa and Lin (2011). It is shown that optimal control always exists for strong-opacity (Dubreil (2010)).

In these approaches, the intruder is generally assumed to have full knowledge of the controller's structure in addition to the system's. Moreover, the set of events the intruder can observe is fixed.

The applicability of SCT depends on the hypothesis made on the system's model. Given E_I, E_O and E_C being respectively the set of events observable by the intruder, of events observable by the controller and of controllable events respectively, SCT can be directly applied in the following cases (Dubreil (2010)):

(1) $E_C \subseteq E_O \subseteq E_I$;
(2) $E_I \subseteq E_C \subseteq E_O$.

Furthermore, to deal with the following two cases, slight extensions of SCT have been suggested in Dubreil (2010):

(3) $E_C \subseteq E_I \subseteq E_O$;
(4) $E_I \subseteq E_O$ and $E_C \subseteq E_O$
but without E_C and E_I being comparable.

In Badouel et al. (2007), the authors solved the problem of concurrent secrecy (Section 3.4) using SCT. Sufficient conditions to compute an optimal controller preserving all secrets are provided, assuming that the controller has complete knowledge of the system and full control over it.

The work of Ben-Kalefa and Lin (2011) considers the verification of both strong and weak opacity. It shows that the solution to the Strong-Opacity Control Problem (SOCP) exists and is unique if all controllable events are observable. However solutions for the Weak-Opacity Control Problem (WOCP) does not exist. This means that if a system is not weakly opaque w.r.t. a given secret language, there exists no controllable and observable sublanguage which can assure weak opacity.

In Darondeau et al. (2014), the authors lift the opacity enforcing control problem using SCT from a single finite transition systems to families of finite transition systems specified by modal transition systems (Larsen (1990)). The objective is to ensure opacity of a secret predicate on all models of a LTS derived from a given modal transition system.

Using SCT is naturally more suited to language-based notions of opacity. However, the verification of initial state opacity has been addressed in Saboori (2011) by means of reformulation of ISO to LBO, under regular SCT hypothesis (cases (1) and (2)). Similar work was performed for infinite-step opacity even though it cannot be so easily translated to LBO. It is shown that the approach for ISO can be extended by using a finite bank of supervisors and ensure infinite-step opacity in a minimally restrictive way.

Superlanguages and Sublanguages In a nutshell, supervisory control resumes to find the supremal sublanguage that ensures opacity. In Ben-Kalefa and Lin (2009), the authors further investigate language composition and show that opacity properties (with secrets being languages) are closed under union, but may not be closed under intersection. They also demonstrate the following results:
(i) the supremal strongly opaque sublanguage exists and is unique;
(ii) the minimal strongly opaque superlanguage exists but may not be unique;
(iii) the minimal weakly opaque superlanguage exists but may not be unique; and
(iv) the supremal not opaque sublanguage exists and is unique.

4.3 Enforcement of opacity properties

Opacity enforcement at run-time was recently surveyed in Falcone et al. (2014). Enforcement does not restrict the system behavior anymore. Instead, it "hides" some of the system's output events whenever it is necessary. It is a non-intrusive approach, opposite to supervision. There are three main approaches used for opacity enforcement: 1) Deleting occurrences of events from the output; 2) Adding events to the output; 3) Delaying the output.

Deletion of events Considering a trace observed by the intruder, it may happen that the observation of the next event discloses the secret. A simple idea is to hide the occurrence of this event from observation at run-time (and possibly only this single occurrence) to avoid information flow.

Main work achieving this is synthesized in Cassez et al. (2012). In this approach, the enforcer is a device called a mask. This mask restricts the observable outputs of the system either in a static or dynamic fashion. The latter case allows the mask to adapt to the intruder observation mapping (assumed to be dynamic) at each execution step.

Addition of events Deleting events from the output was still considered as intrusive by some researchers. Even if the internal behavior of the system is no longer restricted (as it is with SCT), its actual output is.

To cope with this problem, Wu and Lafortune derived a method which artificially adds output to the set of observable events. This approach is called *insertion functions* (Wu (2014)). An insertion function is a monitoring interface at the system's output that changes it by inserting additional ("fake") observable events.

Remark 6. Both these approaches were suggested in Ligatti et al. (2005), which proposed an enforcement mechanism called edit-automata. This mechanism featured the idea of "suppressing" and "inserting" actions in the current execution of a system but without direct application to information flow and opacity.

Delay of events The last approach to enforce of opacity properties is to delay emissions of one or several events which would have disclosed the secret, up to the point the disclosure is of no interest anymore, or the system reaches a state in which opacity holds again. This method allows the full system behavior as well, but can only apply to secrets for which time duration is of concern.

This approach has been presented in Saboori and Hadjicostis (2007) and applied to K-step (weak) opacity. It was later extended in Falcone et al. (2014) to K-step strong opacity.

5. QUANTIFYING OPACITY

We presented in Section 3 main formulations of opacity properties which have been considered in the literature. With these definitions, even decidable problems (refer to Section 6) only provide a yes/no answer to the system's opacity. Control (Section 4.2) and enforcement (Section 4.3) can manage to turn a non-opaque system into an opaque one.

However, this only accounts for deterministic models, which is known to be a strong limitation in practice. Thus, researchers extended some notions of opacity and tried to quantify it in a probabilistic setting. That is, how can one evaluate the possible information leakage of a system w.r.t. a given secret? Hence, for a given system's execution, we do not ask if there exists an observably equivalent execution, but how many there are, with a probabilistic measure taking into account the likelihood of such executions.

5.1 Quantification of language-based opacity

Initial work on quantification of opacity properties was reviewed in Bryans et al. (2013). It provides quantitative measures of LBO in a probalistic setting but it is limited to purely probalistic models, based on labeled Markov chains.

In Bérard et al. (2010) two dual notions of probabilistic opacity are introduced:
(i) *Liberal probabilistic opacity (LPO)* measures the probability for an intruder observing a random execution of the system to be able to gain information he can be sure about. This definition provides a measure of how insecure the system is. $LPO = 0 \Leftrightarrow LBO$.
(ii) *Restrictive probabilistic opacity (RPO)* measures the level of certitude in the information acquired by an intruder observing the system. $RPO = 0$ means the system is never opaque, whichever the running execution.

This work was extended very recently in Bérard et al. (2015) to Markov decision processes with infinite executions. Quantification is performed through the computation of a *probabilistic disclosure (PD)*. It is the probabilistic measure that a run disclosing the secret has been executed. Several problems are addressed, among which:
(i) General disclosure: Is PD bigger than a threshold?
(ii) Limit disclosure problem: Is $PD = 1$?
(iii) Almost-sure disclosure: does there exists a scheduler such that $PD = 1$?
(iv) Almost-sure opacity: Is $PD = 0$?

Future extensions to this work would include the investigation of disclosure before some given delay, either as a number of steps in the spirit of Saboori (2011), or for probabilistic timed systems with an explicit time bound.

5.2 Quantification of state-based opacity

Saboori first investigated the extension of state-based opacity properties to probabilistic models. Three probabilistic properties are presented in Saboori and Hadjicostis (2014):
(i) *Step-based almost current-state opacity* considers the *a priori* probability of violating current state opacity following any sequence of events of length K. It requires this probability to lie below a threshold for all possible lengths $k = (0, 1, \dots K)$. It is the extension of K-step opacity.
(ii) *Almost current-state opacity* is step-based almost current-state opacity when there is no consideration regarding the length of the sequence of events, i.e., it considers the *a priori* probability of violating CSO following any sequence of events. It requires this probability to lie below a threshold. It is the extension of infinite-step opacity.
(iii) *Probabilistic current-state opacity* holds if the maximum increase in the conditional probability that the system's current state lies in the set of secret states (conditioned on a sequence of observations) compared to the case when no observation is available (prior probability) is bounded.

These definitions were extended to ISO in Keroglou and Hadjicostis (2013) for systems modeled as probabilistic finite automata:
(i) *Step-based almost initial state opacity* captures the *a priori* probability that the system will generate behavior that violates initial state opacity after a certain number of events.
(ii) *Almost initial-state opacity* captures the *a priori* probability that the system will eventually generate behavior that violates initial state opacity.

Finally, Ibrahim et al. (2014) extended step-based almost current-state opacity from Saboori and Hadjicostis (2010). Instead of the disclosure probability being below a threshold at each time step, it considers the probability of revealing the secret over the set of all behaviors. Two properties are introduced :
(i) *S_τ-Secrecy* (stochastic-secrecy) holds if the probability of secret disclosure is always below τ.
Secrecy $\Leftrightarrow$ S_0-secrecy.
(ii) *I-S-Secrecy* (increasing stochastic-secrecy) hold if, whatever the threshold, there exists a size n of execution length beyond which every trace has a disclosure probability below the threshold.

6. DECIDABILITY AND COMPLEXITY OF OPACITY PROPERTIES

Opacity is a very general property. As a result, many opacity problems are undecidable. This was demonstrated in Bryans et al. (2008). It remains undecidable for general finite labeled transition systems if you do not restrict the class of observation function. Even for decidable opacity problems, they are computationally complex to solve in general. This sections synthesizes decidability and complexity results demonstrated in the literature.

Note that LBO, ISO and CSO – referred to as general opacity problems – have been proven to be reducible into one another in polynomial time (Wu and Lafortune (2013), Chédor et al. (2014)). Therefore, these problems have same decidability and complexity (since their own complexity is at least polynomial).

Table 1 synthesizes decidability and complexity results of general opacity problems w.r.t the system's model and the observation mapping. Table 2 gathers results from opacity quantification approaches. Finally, more specific complexity results are presented in Table 3 .

7. APPLICATIONS AND RELATED ISSUES

Over the literature, some opacity properties and validation strategies have been applied and evaluated. The reference case of study is known as the *Dinning cryptographers* problem, introduced by Chaum (1988); see e.g., Bérard et al. (2010), Wu and Lafortune (2013). It illustrates properties of ISO and CSO. Another ISO application is presented in Saboori (2011), related to *encryption using pseudo random generators*. The same work also presents the problem of *sensor network coverage* for vehicle tracking. *Opacity Issues in Games with Imperfect Information* is another application considered in Maubert et al. (2011). It exhibits relevant opacity verification problems, which noticeably generalizes approaches considered in the literature for opacity analysis in DES. Finally, the problem of *Ensuring Privacy in Location-Based Services* was studied in Wu (2014), using opacity techniques.

Much of the work in the DES community is essentially theoretical. This applies to opacity-related papers as well, therefore the shortness of this section. However, there have been a few implementations, which are presented in the following section.

Tools and implementation

Saboori used the UMDES Library to implement the verification method for infinite-step opacity, as described in Saboori (2011). UMDES is a library of C routines developed at the University of Michigan (UMDES (2009)) for studying DES modeled by finite automata.

Table 1. Decidability and complexity results for general opacity problems

System model	Observation mapping	Decidability	Complexity	Reference
Petri Nets	–	Undecidable	–	Bryans et al. (2008)
Finite labeled transition system	Static	Undecidable	–	Bryans et al. (2008)
	m-orwellian	Decidable	PSPACE-complete	Cassez et al. (2012)
Pushdown automata (PDA)	–	Undecidable / Decidable if $X \backslash X_S$ is a visible PDA	– / 2(1)-EXPTIME (if X_S is a visible PDA)	Kobayashi and Hiraishi (2013)
Recursive tile systems (RTS)	–	Undecidable	–	Chédor et al. (2014)
Weighted RTS ($\mathcal{C}$wRTS)	–	Decidable	2-EXPTIME	Chédor et al. (2014)

Table 2. Decidability and complexity results for quantified opacity problems

Problem	Decidability	Complexity	Reference
General disclosure Limit disclosure opacity Almost-sure disclosure	Decidable / Undecidable with partial observation of the controller	Polynomial / –	Bérard et al. (2015)
Almost-sure opacity (perfect observation)	Decidable	Polynomial	
Almost-sure opacity (partial observation)	Decidable for ω-regular secrets	EXPTIME	
Step-based / almost current-state opacity	Decidable	PSPACE-hard	Saboori and Hadjicostis (2014)
Probabilistic current-state opacity opacity	Undecidable	–	
Step-based / almost initial-state opacity	Decidable	–	Keroglou and Hadjicostis (2013)
S_τ-Secrecy/I-S-Secrecy	Decidable	–	Ibrahim et al. (2014)

Table 3. Other complexity results

Problem	Complexity	Order	Reference
Current-state opacity	PSPACE-complete	$O(2^N)$	Saboori (2011)
Initial-state opacity	PSPACE-complete	$O(4^N)$	
K-Step opacity	NP-hard	$O((\lvert E_{obs}\rvert + 1)^K \times 2^N)$	
Infinite-Step opacity	PSPACE-hard	–	
Resolution of initial-state	Polynomial	–	Hadjicostis and Keroglou (2014)
LBO Strong-opacity	PSPACE-complete	–	Lin (2011)
LBO Weak-opacity	Polynomial	–	Zhang et al. (2012)
Static mask synthesis	PSPACE-complete	–	Cassez et al. (2012)
Dynamic mask systhesis	EXPTIME lower bound	–	

Falcone developed a specific toolbox named Takos: a Java Toolbox for the Analysis of K-Opacity of Systems (Takos (2010)) to implement the K-step opacity enforcement method presented in Falcone et al. (2014) (using delays).

Finally, in Klai et al. (2014), a symbolic observation graph-based opacity checker has been implemented in C++ using a binary decision diagram package called BuDDy (BuDDy (1998)). Results are compared with the Takos toolbox on the other well known *Dinning philosophers* problem.

8. CONCLUSIONS AND OPEN PROBLEMS

Over the past ten years, opacity applied to DES has been broadly studied. Almost all opacity problems proven decidable have a known complexity. Future trends are oriented toward infinite-state discrete-event models, eventually coupled with probabilistic transition functions.

Moreover, in order to broaden the fields of applications, one could consider opacity validation from another perspective. Starting from a fully observable system and a given secret, which events one should "hide" in order to assure opacity. This approach could provide a pragmatic methodology for people interested in designing opaque systems. To the best of our knowledge, this problem has never been addressed so far and is an interesting perspective.

REFERENCES

Badouel, E., Bednarczyk, M., Borzyszkowski, A., Caillaud, B., and Darondeau, P. (2007). Concurrent secrets. *Discrete Event Dynamic Systems*, 17(4), 425–446. doi:10.1007/s10626-007-0020-5.

Ben-Kalefa, M. and Lin, F. (2009). Opaque superlanguages and sublanguages in discrete event systems. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, 199–204. IEEE. doi:10.1109/CDC.2009.5400704.

Ben-Kalefa, M. and Lin, F. (2011). Supervisory control for opacity of discrete event systems. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, 1113–1119. IEEE. doi:10.1109/Allerton.2011.6120292.

Bérard, B., Chatterjee, K., and Sznajder, N. (2015). Probabilistic opacity for markov decision processes. *Information Processing Letters*, 115(1), 52–59. doi:10.1016/j.ipl.2014.09.001.

Bérard, B., Mullins, J., and Sassolas, M. (2010). Quantifying opacity. In *Quantitative Evaluation of Systems (QEST), 2010 Seventh International Conference on the*, 263–272. IEEE. URL `http://arxiv.org/pdf/1301.`

6799.pdf.

Bryans, J.W., Koutny, M., Mazaré, L., and Ryan, P.Y. (2008). Opacity generalised to transition systems. *International Journal of Information Security*, 7(6), 421–435. doi:10.1007/s10207-008-0058-x.

Bryans, J.W., Koutny, M., and Mu, C. (2013). *Towards quantitative analysis of opacity*. Springer. doi:10.1007/978-3-642-41157-1_10.

Bryans, J.W., Koutny, M., and Ryan, P.Y. (2005). Modelling opacity using petri nets. *Electronic Notes in Theoretical Computer Science*, 121, 101–115. doi:doi:10.1016/j.entcs.2004.10.010.

BuDDy (1998). Buddy. online. URL http://vlsicad.eecs.umich.edu/BK/Slots/cache/www.itu.dk/research/buddy/.

Cassez, F., Dubreil, J., and Marchand, H. (2012). Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, 40(1), 88–115. doi:10.1007/s10703-012-0141-9.

Chaum, D. (1988). The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1), 65–75. doi:10.1007/BF00206326.

Chédor, S., Morvan, C., Pinchinat, S., Marchand, H., et al. (2014). Diagnosis and opacity problems for infinite state systems modeled by recursive tile systems. *Discrete Event Dynamic Systems*. doi:10.1007/s10626-014-0197-3.

Darondeau, P., Marchand, H., and Ricker, L. (2014). Enforcing opacity of regular predicates on modal transition systems. *Discrete Event Dynamic Systems*, 1–20. doi:10.1007/s10626-014-0193-7.

Dubreil, J. (2010). *Monitoring and Supervisory Control for Opacity Properties*. English. tel-00461306, Ph.D. dissertation, Software Engineering, Universit Rennes 1. URL https://tel.archives-ouvertes.fr/tel-00461306/.

Dubreil, J., Darondeau, P., and Marchand, H. (2008). Opacity enforcing control synthesis. In *Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on*, 28–35. IEEE. doi:10.1109/WODES.2008.4605918.

Dubreil, J., Jéron, T., Marchand, H., et al. (2009). Monitoring confidentiality by diagnosis techniques. In *European Control Conference*, 2584–2589.

Falcone, Y. and Marchand, H. (2013). Runtime enforcement of k-step opacity. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, 7271–7278. IEEE. doi:10.1109/CDC.2013.6761043.

Falcone, Y., Marchand, H., et al. (2014). Enforcement and validation (at runtime) of various notions of opacity. *Discrete Event Dynamic Systems*, pp.42. doi:10.1007/s10626-014-0196-4.

Hadjicostis, C.N. (2012). Resolution of initial-state in security applications of des. In *Control & Automation (MED), 2012 20th Mediterranean Conference on*, 794–799. IEEE. doi:10.1109/MED.2012.6265735.

Hadjicostis, C. and Keroglou, C. (2014). Opacity formulations and verification in discrete event systems. In *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*, 1–12. IEEE. doi:10.1109/ETFA.2014.7005032.

Ibrahim, M., Chen, J., and Kumar, R. (2014). Secrecy in stochastic discrete event systems. In *Networking, Sensing and Control (ICNSC), 2014 IEEE 11th International Conference on*, 48–53. IEEE. doi:10.1109/ICNSC.2014.6819598.

Keroglou, C. and Hadjicostis, C.N. (2013). Initial state opacity in stochastic des. In *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*, 1–8. IEEE. doi:10.1109/ETFA.2013.6648005.

Klai, K., Hamdi, N., and Hadj-Alouane, N.B. (2014). An on-the-fly approach for the verification of opacity in critical systems. In *WETICE Conference (WETICE), 2014 IEEE 23rd International*, 345–350. IEEE. doi:10.1109/WETICE.2014.84.

Kobayashi, K. and Hiraishi, K. (2013). Verification of opacity and diagnosability for pushdown systems. *Journal of Applied Mathematics*, 2013. doi:10.1155/2013/654059.

Larsen, K.G. (1990). Modal specifications. In *Automatic Verification Methods for Finite State Systems*, 232–246. Springer. doi:10.1007/3-540-52148-8_19.

Ligatti, J., Bauer, L., and Walker, D. (2005). Edit automata: Enforcement mechanisms for run-time security policies. *International Journal of Information Security*, 4(1-2), 2–16. doi:10.1007/s10207-004-0046-8.

Lin, F. (2011). Opacity of discrete event systems and its applications. *Automatica*, 47(3), 496–503. doi:10.1016/j.automatica.2011.01.002.

Maubert, B., Pinchinat, S., and Bozzelli, L. (2011). Opacity issues in games with imperfect information. *arXiv preprint arXiv:1106.1233*. doi:10.4204/EPTCS.54.7.

Mazaré, L. (2004). Using unification for opacity properties. In *Proceedings of WITS*, volume 4, 165–176. URL http://www-verimag.imag.fr/TR/TR-2004-24.pdf.

Mullins, J. and Yeddes, M. (2013). Opacity with orwellian observers and intransitive non-interference. *arXiv preprint arXiv:1312.6426*. URL http://arxiv.org/abs/1312.6426.

Paoli, A. and Lin, F. (2012). Decentralized opacity of discrete event systems. In *American Control Conference (ACC), 2012*, 6083–6088. IEEE. doi:10.1109/ACC.2012.6315028.

Saboori, A. (2011). *Verification and enforcement of state-based notions of opacity in discrete event systems*. Ph.D. thesis, University of Illinois at Urbana-Champaign. URL http://hdl.handle.net/2142/18226.

Saboori, A. and Hadjicostis, C.N. (2007). Notions of security and opacity in discrete event systems. In *Decision and Control, 2007 46th IEEE Conference on*, 5056–5061. IEEE. doi:10.1109/CDC.2007.4434515.

Saboori, A. and Hadjicostis, C.N. (2010). Opacity verification in stochastic discrete event systems. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, 6759–6764. IEEE. doi:10.1109/CDC.2010.5717580.

Saboori, A. and Hadjicostis, C.N. (2014). Current-state opacity formulations in probabilistic finite automata. *Automatic Control, IEEE Transactions on*, 59(1), 120–133. doi:10.1109/TAC.2013.2279914.

Takai, S. and Kumar, R. (2009). Verification and synthesis for secrecy in discrete-event systems. In *American Control Conference, 2009. ACC'09.*, 4741–4746. IEEE. doi:10.1109/ACC.2009.5160162.

Takos (2010). Takos: A java toolbox for analyzing the k-opacity of systems. online. URL http://toolboxopacity.gforge.inria.fr/.

Umdes (2009). Umdes-lib. online. URL http://www.eecs.umich.edu/umdes/toolboxes.html. Software library.

Wu, Y.C. (2014). *Verification and Enforcement of Opacity Security Properties in Discrete Event Systems*. Ph.D. thesis, University of Michigan. URL http://hdl.handle.net/2027.42/108905.

Wu, Y.C. and Lafortune, S. (2013). Comparative analysis of related notions of opacity in centralized and coordinated architectures. *Discrete Event Dynamic Systems*, 23(3), 307–339. doi:10.1007/s10626-012-0145-z.

Zaytoon, J. and Lafortune, S. (2013). Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2), 308–320. doi:10.1016/j.arcontrol.2013.09.009.

Zhang, B., Shu, S., and Lin, F. (2012). Polynomial algorithms to check opacity in discrete event systems. In *Control and Decision Conference (CCDC), 2012 24th Chinese*, 763 – 769. IEEE. doi:10.1109/CCDC.2012.6244117.