Opacity and Abstractions

Jérémy Dubreil

INRIA Rennes - Bretagne Atlantique, jeremy.dubreil@irisa.fr

Abstract. The opacity property characterises the absence of confidential information flow towards inquisitive attackers. Verifying opacity is well established for finite automata but is known to be not decidable for more expressive models like Turing machines or Petri nets. As a consequence, for a system dealing with confidential information, formally certifying its confidentiality may be impossible, but attackers can still infer confidential information by approximating systems' behaviours. Taking such attackers into account, we investigate the verification of opacity using abstraction techniques to compute executable counterexamples (attack scenarios). Considering a system and a predicate over its executions, attackers are modelled as semi-conservative decision process determining from observed traces the truth of that predicate. Moreover, we will show that the most precise the abstraction is, the most accurate(and then dangerous) the corresponding class of attackers will be. Consequently, when no attack scenario is detected on an approximate analysis, we know that this system is safe against all "less precise" attackers. Therefore, this can be used to provide a level of certification relative to the precision of abstractions.

Introduction

Ensuring the confidentiality of critical information stored on or transferred through computer systems has become one of the most challenging objectives of modern hardware and software design. Interconnected networks like Internet or mobile phones are open by nature and then vulnerable to malicious attacker. But in practise, the level of security is often determined by the quantity of known and public vulnerabilities. This approach, possibly forgetting vulnerabilities that are only known by a malicious group of users, is not satisfying regarding the significant place of information technologies in some critical sectors like medicine, e-government or finance. For example, large scale stealing of medical records or massive modification of votes on e-voting systems can be very dangerous. Then, there should be no security breaches on such infrastructures. Furthermore, security of systems alone is not sufficient, the confidence users have in their security is also imperative. Indeed, even if a system happens to be secure, it is crucial for users to know that it is secure. This situation implies the development of reliable methods for certifying the absence of security breaches on such critical services. When a service needs to keep information secret, formal certification is essentially the only way to gain confidence since information flow attacks may be hard to detect and the damages intricate to recover. Unfortunately, manual analysis is very expensive and requires a high level of expertise. Moreover, it such analysis is often impossible to achieve in practise for large infrastructures, especially when updates are regularly performed.

In order to automate the certification process, security requirements have to be formalised. For this purpose security properties are generally classified into three categories [3]:

- *integrity* (users cannot perform some illegal actions),
- *availability* (some user process can always be performed), and
- *confidentiality* (unauthorised users cannot acquire secret information).

For example, it is expected for an e-voting system that votes cannot be modified by a third party: this is a concern of integrity. Also, requiring that every elector can vote is a concern of availability. Finally, requiring that no third party can know the vote of an elector is a concern of confidentiality. A *security policy* is a set of security properties that have to be simultaneously satisfied.

In this article, we will only focus on confidentiality properties. We consider a critical system, modelled by a possibly infinite labelled transition system, required to keep secret some confidential information against inquisitive attackers. For security analysis, it is generally safer to consider that potential attackers have a complete knowledge of the system structure and we will make similar assumptions throughout the article. As an example this would be the case when analysing standardised protocols or software with sources publicly available. We also assume that attackers are partially observing the system. Attackers typically observe the interactions, i.e. the inputs and outputs, but we can also consider timing information, power consumption, or electromagnetic leaks to be observable as long as they can be modelled in an event based fashion. The partial observation is given by a function mapping the runs to a set of observed traces.

The notion of *opacity*, introduced in [22], is a formalisation of the ability of a system to keep secret some critical information. Considering a predicate over the runs, the confidential information consists in the occurrence of a run executed by the system satisfying the predicate. In [4], the authors proved that opacity is general enough to model a significant set of confidentiality properties like anonymity and non-deductibility. The notion of trace-based non-interference, defined in [14,15] and generalising the non-interference introduced in [18], can also be considered using opacity. A predicate ϕ is said to be opaque on a system M if whenever a run r satisfying ϕ is executed, there exists another run r' which is observationally equivalent to r from the attacker point of view and does not satisfy ϕ. In that case, if ν is the observed trace of r, the attacker cannot discern if r or r' has been executed and then cannot infer from ν whether the run executed satisfies ϕ.

For example, consider a program P to be an implementation of a cryptographic component that needs to keep secret the value of some key k. This value is randomly generated when P is launched and is not modified during execution. Let the predicate ϕ be defined as the set of executions of P that are only possible when the value of k is smaller than a given value k_0. If ϕ is not opaque on P, then for some observation ν, the attacker knows that the current execution of P belongs to ϕ and then that $0 \leq k \leq k_0$. In this example, the program P also leaks information if the attacker can infer that $k_0 \leq k$. This idea is formalised in [2] with the notion of *secrecy* where a system leaks information about a predicate ϕ when for a run r, the attacker knows that $r \models \phi$ or that $r \models \neg\phi$. Secrecy can be handled considering the opacity of ϕ and $\neg\phi$ but the information $\neg\phi$ may not always be relevant as the following example will point out.

Let M be a basic authorisation service. Users send their logins and passwords to M which read from a database file `pwd` if the values match and decide whether to let users proceed to the next step. The file `pwd` is normally protected, nobody can read its content, but in order to make the comparison, M enables read access on `pwd`, compares the values and then disables read access. The attacker wants to infer that `pwd` can be read and then launch a `print pwd` command to steal all login information. We can see on this example, inspired from a security issue on early UNIX systems, that the attacker is not interested in the information "pwd is not read accessible" which is likely to be the case most of the time during execution. This case of attacks also suggests that attackers want to know "the current run executed by M satisfies ϕ" rather than "a run satisfying ϕ has been executed by M". In the previous example, when the login is successful, the attacker knows that `pwd` has been, at some time, read accessible but this is not a useful information. This makes a difference with works on failures diagnosis of discrete events systems [23,20].

We reformulate the notion of opacity of [4], showing that the property finds roots in the possible world model of epistemic logic [19,13]. Such formulation will help us to investigate what information is lost and what is preserved when reasoning with overapproximations and underapproximations of the semantic of M. Then, given an approximation framework based on Galois connections, we provide a construction of monitors allowing an attacker to infer confidential information from observed traces. Finally, using together an overapproximation and an underapproximation of M, we give a method to statically detect cases of information flow.

In section 1, we present the notation for the considered models of system. In section 2 we detail our model of attacker from which we derive the notion of property inference from imperfect information through the K ("knows") modality of epistemic logic. Opacity is then defined as the impossibility to acquire from observations that systems behaviours belongs to a secret. We study in section 3 the opacity verification problem for finite automata and proves its PSPACE-complete complexity. We use these results in section 4 to apply abstraction techniques for the verification of opacity on infinite systems.

1 Notations and Definitions

Relations Let S be a set. A relation r over S is a subset of $S \times S$. We say that this relation r is

- *reflexive*: $\forall x \in S, (x, x) \in r$
- *symmetric*: $\forall x, y \in S, (x, y) \in r \Rightarrow (y, x) \in r$
- *antisymmetric*: $\forall x, y \in S, (x, y) \in r \wedge (y, x) \in r \Rightarrow x = y$
- *transitive*: $\forall x, y, z \in S, (x, y) \in r \wedge (y, z) \in r \Rightarrow (x, z) \in r$

A relation θ over S is an *equivalence relation* if θ is reflexive, symmetric and transitive. We use the notation $x \sim_\theta y$ for $(x, y) \in \theta$ and $[x]_\theta = \{y \in s : y \sim_\theta x\}$.

Lattice Theory Let $\sqsubseteq$ be a relation over S. We say that $\sqsubseteq$ is an *order relation* if $\sqsubseteq$ is reflexive, antisymmetric and transitive. In that case, $(S, \sqsubseteq)$ is called a partially ordered

set. Let $(A, \sqsubseteq_a)$ and $(B, \sqsubseteq_b)$ be two partially ordered set and $f : A \to B$. The function f is monotone if for all $x, y \in A$, $f(x) \sqsubseteq_b f(y)$ whenever $x \sqsubseteq_a y$. Two functions $\alpha : A \to B$ and $\gamma : B \to A$ establish a Galois connection between A and B, denoted $(A, \sqsubseteq_a) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (B, \sqsubseteq_b)$, when for all $x \in A$ and $y \in B$, $\alpha(x) \sqsubseteq_b y$ if and only if $x \sqsubseteq_a \gamma(y)$. In that case, both α and γ are monotone.

For $X \subseteq A$ and $y \in A$ we say that y is an upper bound (resp. lower bound) if for all $x \in X$, $x \sqsubseteq_a y$ (resp. $y \sqsubseteq_a x$). Also, y is a least upper bound, if for every other upper bound z, we have $y \sqsubseteq_a z$. When such upper bound exists, it is unique and denoted $lub(X)$. Similarly, $glb(X)$ denotes the greatest lower bound. A partially ordered set $(A, \sqsubseteq_a)$ is a lattice whenever $lub(\{x, y\})$ and $glb(\{x, y\})$ exists for every $x, y \in A$ and is a complete lattice whenever $lub(X)$ and $glb(X)$ exists for every $X \subseteq A$. We also note $x \sqcap_a y$, $x \sqcup_a y$, $\sqcap_a X$, $\sqcup_a X$, $\bot_a$ and $\top_a$ for $glb(\{x, y\})$, $lub(\{x, y\})$, $glb(X)$, $lub(X)$, $glb(A)$ and $lub(A)$ respectively. For example, if S is a set, then $(\mathcal{P}(S), \subseteq)$, the subsets of S forms a complete lattice for the inclusion order relation, with $\sqcap = \cap$ and $\sqcup = \cup$.

We note $x \sqsubset_a y$ when $x \sqsubseteq_a y$ and $x \neq y$. A set $\{x_n\}_{n \in \mathbb{N}} \subseteq A$ is an increasing chain if for all $i \in \mathbb{N}$, $x_i \sqsubseteq_a x_{i+1}$. We say that the chain is stricly increasing if for all $i \in \mathbb{N}$, $x_i \sqsubset_a x_{i+1}$. The height of a lattice A is the maximum number of pairwise disjoint element in a chain on A. It is denoted $height(A)$. Typically, $height(A) < \infty$ if $|A| < \infty$.

Let $(A, \sqsubseteq_a)$ and $(B, \sqsubseteq_b)$ be complete lattices and $g : A \to B$. We say that g is *ω-continuous* if for every increasing chain $\{x_n\}_{n \in \mathbb{N}} \subseteq A$, $g(\sqcup_a \{x_n : n \in \mathbb{N}\}) = \sqcup_b \{g(x_n) : n \in \mathbb{N}\}$. Let $id_A : A \to A$ denote the identity function: for every $x \in A$, $id_A(x) = x$, and let $f : A \to A$ be a function. The iterations of f are defined by $f^0 = id_A$ and for $n \in \mathbb{N}$, $f^n = f^{n-1} \circ f$. An element $x \in A$ is a *fixed point* of f if $f(x) = x$. If f is monotone and ω-continuous, then there exists a *least fixed point* of f, denoted $lfp(f) \in A$ which can be computed by $lfp(f) = \sqcup_a(\{f^n(\bot)\})$. If $height(A) < \infty$, then there exists $N \in \mathbb{N}$ such that $lfp(f) = f^N(\bot)$. Similarly for the *greatest fixed point* $gfp(f) = \sqcap(\{f^n(\top)\})$.

Labelled Transition Systems Let Λ and S be possibly infinite sets representing respectively the labels for possible actions and the labels for possible states, or configurations, of a system. If a finite set of variables are used within a program P then S can represent all the possible valuations of the variables in the semantics of P. If A and B are two subsets of Λ, we note $AB = \{\lambda\lambda' : \lambda \in A, \lambda' \in B\}$. We denote $\Lambda^n = \Lambda^{n-1}\Lambda$, assuming that $\Lambda^0 = \{\varepsilon\}$, where ε denote the empty word, and $\Lambda^* = \cup_{n \in \mathbb{N}} \Lambda^n$ is the set of finite words over Λ. For $w \in \Lambda^*$, $|w|$ denotes the number of letters occuring in w. A set L of finite words over Λ, $L \subseteq \Lambda^*$, is called a language.

Definition 1. *A LTS is a tuple $M = (\Lambda, S, \delta, S_0)$ where:*

- *Λ is an alphabet of events;*
- *S is a set of states;*
- *$\delta : \Lambda \times S \to \mathcal{P}(S)$ is the transition function;*
- *$S_0 \subseteq S$ are the initial states.*

The LTS M is deterministic if for all $\lambda \in \Lambda$ and $s \in S$, $|\delta(\lambda, s)| \leq 1$. A finite *run* (or execution) of M is a sequence of the form

$$r = s_0 \xrightarrow{\lambda_1} s_1 \xrightarrow{\lambda_2} s_2 \ \cdots \ s_{n-1} \xrightarrow{\lambda_n} s_n \tag{1}$$

constructed by alternating labels of Λ and states of S such that $s_{i+1} \in \delta(\lambda_{i+1}, s_i), 0 \leq i < n$. The trace $tr(r)$ of the run in (1) is the word $\lambda_0\lambda_1\lambda_2 \ ... \ \lambda_n$. The operators $fst(\cdot)$ and $lst(\cdot)$ denote respectively the first and the last state of a run. For r in (1), $fst(r) = s_0$ and $lst(r) = s_n$. The length of a run is defined by $length(r) = |tr(r)|$. The set of all finite runs which can be constructed from S and Λ is denoted $E(\Lambda, S) = S(\Lambda S)^*$. The set S is identified to $\{s\varepsilon : s \in S\} \subset E(\Lambda, S)$. The set of runs of M, denoted $\mathcal{R}(M)$ is defined by $\mathcal{R}(M) = lfp(f)$, where the function $f : R \mapsto \{S_0\} \cup \{r \xrightarrow{\lambda} s : r \in R, \ s \in \delta(\lambda, lst(r))\}$ is monotone on the complete lattice $\mathcal{P}(E(\Lambda, S))$. For X and Y subsets of S, we note $\mathcal{L}(M, X, Y) = \{tr(r) : r \in \mathcal{R}(M), fst(r) \in X, lst(r) \in Y\}$. The language of M is $\mathcal{L}(M) = \mathcal{L}(M, S_0, S)$. Finally, we define the concatenation of two runs by :

$$\begin{array}{rcl} cat : E(\Lambda, S) \times E(\Lambda, S) & \to & E(\Lambda, S) \\ r, \ s & \mapsto & \begin{cases} r \text{ if } lst(r) = s \\ \text{undefined otherwise} \end{cases} \\ r, \ s \xrightarrow{\lambda} r' & \mapsto & \begin{cases} cat(r \xrightarrow{\lambda} fst(r'), \ r') \text{ if } lst(r) = s \\ \text{undefined otherwise} \end{cases} \end{array}$$

In the sequel, we will simply denote $r \cdot r'$ for $cat(r, r')$. We say that r is a prefix of r', denoted $r \leq r'$, when there exists $r'' \in E(\Lambda, S)$ such that $r' = r \cdot r''$. Then, $\leq$ is an order relation over $E(\Lambda, S)$.

2 Model of attackers and Notion of Opacity

Considering a LTS $M = (\Lambda, S, \delta, S_0)$ as above and a predicate ϕ over the runs of M modelling the secret. We do not put restriction on Λ and S which can be possibly infinite sets. This allow the following definitions to be general enough to apply our results to more specific models like process algebras or Petri nets. The occurrence of a run in M satisfying ϕ is the information that should remain confidential. Let $\mathcal{A}$ denotes an attacker whose aim is to infer that the current run executed by M satisfies ϕ. We understand $\mathcal{A}$ to be a machine, or a class of machines, and not a real person. In that case we can define precisely its capabilities and knowledge. We suppose that $\mathcal{A}$ has an imperfect information about what really happens when a run is executed by M. This is modeled by a equivalence relation θ over the runs of M whose semantic is: when a run r is executed by M, if there exists an other $r' \in E(\Lambda, S)$ such that $r \sim_\theta r'$, then $\mathcal{A}$ cannot discern if r or r' has been executed but knows that at least one run of $[r]_\theta$ has been executed. So in this context of imperfect information, when an unknown run $x \in \mathcal{R}(M)$ is executed, $\mathcal{A}$ has to infer whether x satisfies ϕ, denoted $x \models \phi$, from the known fact $[x]_\theta$. In particular, $\mathcal{A}$ will be sure that $x \models \phi$ when every run of $[x]_\theta$ satisfies ϕ. The notion of opacity defined in [4] is based on this idea. This concept is

also known in the context of modal logic with the formalisation of knowledge using the possible world model [19]. In this setting, agents are considered to know a fact ϕ if this fact is true in every world that are possible from their point of view. Then, for analysing confidentiality for a system M, knowledge about the set of runs $\mathcal{R}(M)$ that are possible according to the transition relation plays a central role in the ability of attackers to acquire information during system's executions. But, computing the exact semantic $\mathcal{R}(M)$ may be impossible when Λ and S are infinite. So, we have to consider attackers reasoning according to an imprecise semantic R of M. To this end, we reformulate the notion of opacity using epistemic modalities to establish links between an attacker reasoning with R and a, possibly only theoretical, ideal attacker reasoning with $\mathcal{R}(M)$. This considerations will allow us to model the more realistic case of attackers reasoning from an overapproximation or an underapproximation of $\mathcal{R}(M)$.

2.1 Possible Worlds Model

We model the knowledge of the attacker $\mathcal{A}$ using the possible worlds model of epistemic logic [19]. More details can be found in [26,13,17]. We also consider the order relation $\leq$ over $E(\Lambda, S)$ to express the behavioural aspects of the computations of M. Let $Atm = \{t_1, t_2, ..., t_k\}$ be a finite set of atomic propositions and $\pi : E(\Lambda, S) \rightarrow \mathcal{P}(Atm)$ a valuation function. Let the language $Fml(Atm)$ to be the smallest set of formulae such that:

- The predicates of Atm, $true$ and $false$ belong to $Fml(Atm)$;
- if ϕ and ψ are in $Fml(Atm)$, then so are $\neg\phi$ and $\phi \wedge \psi$;
- if ϕ is in $Fml(Atm)$, then so is $\Box\phi$;
- if ϕ is in $Fml(Atm)$, then so is $K\phi$.

Consider an attacker $\mathcal{A}$ reasoning from a set of runs $R \subseteq E(\Lambda, S)$ with the indistinguishability relation θ over R. The formulae of $Fml(Atm)$ are interpreted on the Kripke structure $(R, \leq, \theta, \pi)$ as follows, for $r \in R$:

- $(R, r) \models true$;
- $(R, r) \models t$ when $t \in \pi(r)$;
- $(R, r) \models \neg\phi$ when $(R, r) \not\models \phi$;
- $(R, r) \models \phi \wedge \psi$ when $(R, r) \models \phi$ and $(R, r) \models \psi$;
- $(R, r) \models \Box\phi$ when for all $r' \in R$ such that $r \leq r'$, $(R, r') \models \phi$;
- $(R, r) \models K\phi$ when for all $r' \in R$ such that $r' \sim_\theta r$, $(R, r') \models \phi$.

The connectives $\vee$, $\Rightarrow$ and $\Leftrightarrow$ can be derived from $\neg$ and $\wedge$ in the usual way. The modality $\Box$ in $(R, r) \models \Box\phi$ expresses that the predicate ϕ holds for every run of R extending r. Following classical notations of modal logic, we denote by $\Diamond$ the possibility modality, defined by $\Diamond\phi = \neg\Box\neg\phi$. In other words, $(R, r) \models \Diamond\phi$ if and only if there exists a run $r' \in R$ extending r such that $(R, r') \models \phi$.

The modality K, for knowledge, is also a necessity modality of modal logic but depends on θ instead of $\leq$. If $(R, r) \models K\phi$, then $\mathcal{A}$ knows that ϕ is true for every run that $\mathcal{A}$ thinks possible when r is executed. Similarly to $\Diamond$, we can define the possibility modality P by $(R, r) \models P\phi$ whenever $(R, r) \models \neg K\neg\phi$, expressing that for $\mathcal{A}$, ϕ is

possibly true when r is executed. Note that we can identify $\mathcal{A}$ to θ since considering an other attacker $\mathcal{A}'$ with the same indistinguishably relation θ will lead to the same analysis. We will say that a formula ϕ is satisfiable on $(R, \leq, \theta, \pi)$ if there exists $r \in R$ such that $(R, r) \models \phi$.

This simple logic allows us to express information flow properties through the general notion of opacity. Considering temporal modalities allows us to express situations where the confidentiality concerns behavioural aspects of the system. This approach is followed by [25,12,10,11] where the set of confidential behaviour is given as a regular language over the labels of the system. In this paper, we consider a finite set of predicates Ω that are expressed by the accessibility of certain states: there exists $F : \Omega \to \mathcal{P}(S)$ such that for $\phi \in \Omega$, $(R, r) \models \phi$ whenever $lst(r) \in F(\phi)$.

2.2 Definition of Opacity

Definition 2 (Opacity). *We say that the predicates Ω are θ-opaque on M if for all $\phi \in \Omega$, $\mathcal{A}$ cannot acquire whether the run executed by M satisfies ϕ. Formally, using the logic above, Ω is θ-opaque on M when no formula of $\{K\phi, \phi \in \Omega\}$ is satisfiable on $(\mathcal{R}(M), \leq, \theta, \pi)$.*

Remark 1. An equivalent definition of θ-opacity would be to require that the formulae $P\neg\phi$, $\phi \in \Omega$, are always satisfied on $(\mathcal{R}(M), \leq, \theta, \pi)$. In other words, there always exists at least one run, that the attacker $\mathcal{A}$ thinks possible, not ending in one of the $F(\phi)$. This prevents $\mathcal{A}$ to be sure that the current execution of M ends in one the set of states $F(\phi)$.

Remark 2. We can use temporal modalities to express the evolution of the attacker's knowledge during the execution of M. Opacity can then be interpreted on the initial states of M: Ω is θ-opaque on M if and only if for all $s_0 \in S_0$ and for all $\phi \in \Omega$, $(\mathcal{R}(M), s_0) \models \Box\neg K\phi$.

Example 1. Let $Var = \{x_1, x_2, \ldots, x_m\}$ be a finite set of variables used in a program implementing a cryptographic primitive. Let $Dom(x)$ denote the domain of the variable x. Suppose now that x_0 is used to store the value of a cryptographic key. For such system, it is required that an attacker cannot acquire more precise information about the value of x_0 that its domain $Dom(x_0)$. Let Λ denote the set of internal and external transition labels, $S = Dom(x_0) \times Dom(x_1) \times \cdots \times Dom(x_m)$ and M denotes the execution graph of the program. Let $\eta \in Dom(x_0)$ and consider $F_\eta^{\leq} = \{s \in S : x_0 \leq \eta\}$, $F_\eta^{>} = S \setminus F_\eta^{\leq}$ and the corresponding set of predicates $\Omega = \{t_\eta^{\leq}, t_\eta^{>}\}$ where $F(t_\eta^{\leq}) = F_\eta^{\leq}$ and $F(t_\eta^{>}) = F_\eta^{>}$. Consider an attacker observing the inputs and the outputs, with the corresponding indistinguishability relation denoted θ. This attacker will be able to acquire that the value of x_0 is less than (resp. greater than) η when a run r such that $(\mathcal{R}(M), r) \models Kt_\eta^{\leq}$ (resp. $(\mathcal{R}(M), r) \models Kt_\eta^{>}$) is executed by M.

As suggested by the previous example, the notion of opacity can be used to define the notion of secrecy like in [2], where we consider the system to be unsafe when $\mathcal{A}$ either knows that a confidential predicate is satisfied or knows that is is not satisfied. This notion of secrecy is especially suitable to reason about the confidentiality of variables.

Definition 3 (Secrecy). *A predicate ϕ is θ-secret on M when the predicates $\{\phi, \neg\phi\}$ are θ-opaque on M.*

We consider now that $\mathcal{A}$ observes a subset Λ_o of the events of Λ and we note $\Lambda_{uo} = \Lambda \setminus \Lambda_o$ the set of unobservable events. The observed trace of a run is given by the observation function p defined by:

$$\begin{array}{rcl} p : E(\Lambda, S) & \to & \Lambda_o^* \\ s & \mapsto & \varepsilon \\ r \xrightarrow{\lambda} s & \mapsto & \begin{cases} p(r)\lambda \text{ if } \lambda \in \Lambda_o \\ p(r) \text{ otherwise} \end{cases} \end{array}$$

We suppose that $\mathcal{A}$ knows $\mathcal{R}(M)$. The corresponding equivalence relation over $\mathcal{R}(M)$ is: $r \sim_\theta r'$ if $p(r) = p(r')$. In that case, if a run r is executed by M, $\mathcal{A}$ thinks that all the runs of $[r]_\theta = p^{-1}(p(r)) \cap \mathcal{R}(M)$ are possible.

We note that θ is such that for all $\phi \in \Omega$, if $p(r) = p(r')$, then $(\mathcal{R}(M), r) \models K\phi$ if and only if $(\mathcal{R}(M), r') \models K\phi$. So, the attacker can decide the truth of the predicates of Ω on the basis of the observed traces.

Definition 4. *The set of observed traces such that the secret is disclosed is denoted* $Discloser(\Omega, \mathcal{R}(M), p) = p(\{r \in \mathcal{R}(M) : \exists \phi \in \Omega, (\mathcal{R}(M), r) \models K\phi\})$.

The predicates Ω are θ-opaque on M if and only if $Discloser(\Omega, \mathcal{R}(M), p) = \emptyset$ and we will see now how to verify θ-opacity. From the attacker's point of view, the objective is to compute the set $Discloser(\Omega, \mathcal{R}(M), p)$ and, when non empty, observe (or execute) at least one of it traces. We can write this set of traces in a form that will be easier to manipulate for verification purpose: $Discloser(\Omega, \mathcal{R}(M), p) = \{\nu \in \Lambda_o^* : p^{-1}(\nu) \cap \mathcal{R}(M) \neq \emptyset \wedge \exists \phi \in \Omega, \forall r \in p^{-1}(\nu) \cap \mathcal{R}(M), lst(r) \in F(\phi)\}$.

Definition 5. *The set of observed traces such that the secret is preserved is denoted* $Safe(\Omega, \mathcal{R}(M), p) = p(\{r \in \mathcal{R}(M) : \forall \phi \in \Omega, (\mathcal{R}(M), r) \models P\neg\phi\})$.

We can also write: $Safe(\Omega, \mathcal{R}(M), p) = \{\nu \in \Lambda_o^* : \forall \phi \in \Omega, \exists r \in p^{-1}(\nu) \cap \mathcal{R}(M), lst(r) \notin F(\phi)\}$. Then, the set of states Ω are θ-opaque on M if and only if $p(\mathcal{R}(M)) = Safe(\Omega, \mathcal{R}(M), p)$.

3 Opacity Verification Problem

Given a LTS $M = (\Lambda, S, \delta, S_0)$ and an inquisitive attacker $\mathcal{A}$, observing the events of $\Lambda_o \subseteq \Lambda$, trying to infer the truth of secret state-based predicates Ω. We will investigates in this section a method to exhibit attack scenarios. This method can be used to verify whether ϕ opaque on M for an attacker with full knowledge of the semantic $\mathcal{R}(M)$.

This method is based on the operators $post$ and $reach : \mathcal{P}(\Lambda) \to (\mathcal{P}(S) \to \mathcal{P}(S))$ defined as follows, for $B \in \mathcal{P}(\Lambda)$ and $X \in \mathcal{P}(S)$:

- $post(B)(X) = \cup\{\delta(\lambda, s) : \lambda \in B, s \in X\}$;
- $reach(B)(X) = lfp(Y \mapsto X \cup post(B)(Y))$;

Note that $reach$ is always defined but its computation may not terminate. In section 4, we will circumvent this problem by approximating $reach$. In the sequel, we will use the notations $reach_{uo}(\cdot)$ and $coreach_{uo}(\cdot)$ instead of $reach(\Lambda_{uo})(\cdot)$ and $coreach(\Lambda_{uo})(\cdot)$. The first step consists in defining the behaviours of the system as observed by the attacker. This is done by the determinisation operation defined as follows:

Definition 6. *The determinization of M, with respect to Λ_o, gives the deterministic LTS $det_o(M) = (\Lambda_o, \mathcal{P}(S), \Delta, X_0)$ where $X_0 = reach_{uo}(S_0)$[1] and*

$$\begin{aligned} \Delta : \Lambda_o \times \mathcal{P}(S) &\to \mathcal{P}(S) \\ \lambda, X &\mapsto reach_{uo} \circ post(\{\lambda\})(X) \end{aligned}$$

The transition relation Δ is extended to words for $X \in \mathcal{P}(S)$ by $\Delta(\varepsilon, X) = X$ and for $\nu \in \Lambda^*$, $\lambda \in \Lambda$, $\Delta(\nu\lambda, X) = \Delta(\lambda, \Delta(\nu, X))$.

Proposition 1. *For all $\nu \in \Lambda_o^*$, $\Delta(\nu, X_0) = \{s \in S : \exists r \in \mathcal{R}(M),\ p(r) = \nu \wedge lst(r) = s\}$*

Proof. For $\nu \in \Lambda_o^*$, let $Z(\nu) = \{s \in S : \exists r \in \mathcal{R}(M),\ p(r) = \nu \wedge lst(r) = s\}$. It is clear from the definition of Δ that for all $\nu \in \Lambda_o^*$, $\Delta(\nu, X_0) \subseteq Z(\nu)$. Suppose that for $n \in \mathbb{N}$ and for all $\nu \in \Lambda_o^n$, $\Delta(\nu, X_0) = Z(\nu)$. This holds for $n = 0$. Now, let $\nu\lambda \in \Lambda_o^{n+1}$ and $s \in Z(\nu\lambda)$. There exists $r \in R(M)$ such that $p(r) = \nu\lambda$ and $lst(r) = s$. Let r_1 be the longest prefix of r such that $p(r_1) = \nu$. Then we can write $r = r_1 \cdot r_2$ with $r_2 = s_1 \xrightarrow{\lambda} s_2 \xrightarrow{u} s$, $u \in \Lambda_{uo}^*$ and $s_1 \in Z(\nu)$. So $s \in \Delta(\lambda, \{s_1\}) \subseteq \Delta(\lambda, Z(\nu))$. Since $|\nu| = n$, $\Delta(\nu, X_0) = Z(\nu)$ so $s \in \Delta(\lambda, \Delta(\nu, X_0)) = \Delta(\nu\lambda, X_0)$. Finally $\Delta(\nu\lambda, X_0) = Z(\nu\lambda)$ which proves the proposition by induction.

Proposition 2. *If $r \in \mathcal{R}(M)$, then $lst(r) \in \Delta(p(r), X_0)$.*

Proof. We will prove this proposition by recurrence on the length of the run r. If $length(r) = 0$, then $p(r) = \varepsilon$ and $r = s_0 \in S_0 \subseteq X_0 = \Delta(\varepsilon, X_0)$. Suppose now that the proposition is true for runs of length n and let $r \in \mathcal{R}(M)$ such that $length(r) = n+1$. Then, we can write $r = r' \xrightarrow{\lambda} s$ with $length(r') = n$. Let $\nu = p(r')$ and $s' = lst(r')$. There are two possibilities. First, suppose that $\lambda \in \Lambda_{uo}$. Then, $p(r) = \nu$. Also, $s \in reach_{uo}(\{s'\})$ and $s' \in \Delta(\nu, X_0)$. Since $reach_{uo}(\Delta(\nu, X_0)) = \Delta(\nu, X_0)$, we get $s \in \Delta(\nu, X_0)$. Suppose now that $\lambda \in \Lambda_o$. Then, $p(\nu) = \nu\lambda$ and $s \in post(\{\lambda\})(\{s'\}) \subseteq post(\{\lambda\})(\Delta(\nu, X_0)) \subseteq reach_{uo} \circ post(\{\lambda\})(\Delta(\nu, X_0)) = \Delta(\nu\lambda, X_0)$.

Proposition 3. $Discloser(\Omega, \mathcal{R}(M), p) = \bigcup_{\phi \in \Omega} \{\mathcal{L}(det_o(M), X_0, \mathcal{P}(F(\phi)))\}$

Proof. Let $\phi \in \Omega$ such that $\nu \in \mathcal{L}(det_o(M), X_0, \mathcal{P}(F(\phi)))$. According to proposition 2, for all $r \in p^{-1}(\nu) \cap \mathcal{R}(M)$, $lst(r) \in \Delta(\nu, X_0) \subseteq F(\phi)$. This implies that $\nu \in Discloser(\Omega, \mathcal{R}(M), p)$. For the other inclusion, let $\nu \in Discloser(\Omega, \mathcal{R}(M), p)$. There exists $r \in \mathcal{R}(M)$ such that $p(r) = \nu$, so $\nu \in \mathcal{L}(det_o(M))$. Let $s \in \Delta(\nu, X_0)$, there exists $r' \in \mathcal{R}(M)$ such that $p(r') = \nu$ and $s = lst(r')$. According to proposition 1 and the definition of $Discloser$, $s = lst(r') \in F(\phi)$ for some $\phi \in \Omega$. We conclude then $\Delta(\nu, X_0) \subseteq F(\phi)$ and $\nu \in \mathcal{L}(det_o(M), X_0, \mathcal{P}(F(\phi)))$.

[1] X_0 is the set of states that are reachable when the attacker observes nothing (i.e. the trace ε)

Those two propositions implies the following theorem which provides a methodology for verifying opacity based on reachability analysis.

Theorem 1. *The predicates Ω are opaque on (M, θ) if and only if the states $\mathcal{P}(F(\phi))$, $\phi \in \Omega$ are not reachable in $det_o(M)$.*

3.1 Verification Opacity on Finite Models

If Λ and S are finite sets, $det_o(M)$ can always be computed so the verification problem is decidable for finite LTS. We will see that verifying opacity is PSPACE-complete. To prove this, we will see that the universality problem for NFA can be encoded as an opacity problem.

Let $G = (Q, \Sigma, \delta, q_0)$ be a finite and possibly non-deterministic LTS. Let $F \subseteq Q$ a set of final states. We say that G is language universal when $\mathcal{L}(G, q_0, F) = \Sigma^*$. Deciding language universality on an automaton G is known to be complete for PSPACE [24].

Suppose now that an attacker $\mathcal{A}$ knows the model G and observes all Σ. The indistinguishably relation θ is then defined for all $\rho, \rho' \in \mathcal{R}(G)$ by $\rho \sim_\theta \rho'$ when $tr(\rho) = tr(\rho')$. Consider the predicate ϕ defined by the accessibility of the states $Q \setminus F$.

Proposition 4. *G is language universal if and only if $\{\phi\}$ is θ-opaque on G.*

Theorem 2. *If M is a finite LTS and ϕ a state-based property, the opacity verification problem is PSPACE-complete.*

Let M be a finite LTS with a set $\Lambda_o \subseteq \Lambda$. In the case of a finite LTS we can always assume that the attacker knows $\mathcal{R}(M)$. Opacity can be encoded as k language universality problem, one for each $\phi \in \Omega$, thanks to an ε-closure operation which can be performed with a complexity polynomial in the size of M. In this context, the opacity verification problem is equivalent to the language universality problem and is then PSPACE-complete. We proposed an algorithm based on LTS determinization. But a complete determinization procedure can be avoided, for example by applying antichains techniques developed in [8,9] for solving universality problem and therefore obtaining a more efficient algorithms for opacity verification on finite models.

3.2 Verifying Opacity on Infinite Models

Consider now that the sets Λ and S are not necessarily finite. In that case, the fixed point computation required for the operators $reach_{uo}$ and $coreach_{uo}$ may not terminate. Then, the determinization procedure used for verifying opacity on finite models cannot be applied. Furthermore, in [4], the authors proved that verifying opacity is not decidable for Turing machines and Petri nets. But for security purpose, we also have to take into account attacker reasoning on approximations. In [4], the authors introduced the notion of *uo-opacity* (for under/over-opacity) to handle approximations and apply this approach for verifying uo-opacity (and then opacity) on Petri net using *coverability graphs*. We borrow some ideas from uo-opacity but we follow a slightly different presentation in order to connect approximations with the epistemic logic formalism and

abstract interpretation approaches based on Galois connections [5,6]. Following [16], we place approximations as part the attacker's model.

The complete operational semantic $\mathcal{R}(M)$ may not be computable. Nevertheless, we consider $\mathcal{A}$ knowing $\mathcal{R}(M)$ as above. This attacker is only theoretical but we will use it as a reference of the most accurate attacker. Let $\mathcal{A}^\sharp$ and $\mathcal{A}^\flat$ be two attackers reasoning respectively on an overapproximation $R^\sharp$, i.e. $\mathcal{R}(M) \subseteq R^\sharp \subseteq E(\Lambda, S)$ and an underapproximation $R^\flat \subseteq \mathcal{R}(M)$. We suppose that $R^\sharp$ and $R^\flat$ can be effectively computed. $\mathcal{A}$, $\mathcal{A}^\sharp$ and $\mathcal{A}^\flat$ still observes the same set of events $\Lambda_o \subseteq \Lambda$ with the same projection p as above. We denote by θ, $\theta^\sharp$ and $\theta^\flat$ the corresponding equivalence relations, defined through their equivalence classes by: $[r]_{\theta^\flat} = p^{-1}(p(r)) \cap R^\flat$, for $r \in R^\flat$; $[r]_{\theta^\sharp} = p^{-1}(p(r)) \cap R^\sharp$, for $r \in R^\sharp$. The sets $Discloser(\Omega, R^\flat, p)$, $Discloser(\Omega, R^\sharp, p)$, $Safe(\Omega, R^\flat, p)$ and $Safe(\Omega, R^\sharp, p)$ are defined in the same way that $Discloser(\Omega, \mathcal{R}(M), p)$ and $Safe(\Omega, \mathcal{R}(M), p)$.

As it is classically the case when reasoning with an overapproximation, we cannot conclude from $Discloser(\Omega, R^\sharp, p) \neq \emptyset$ that $Discloser(\Omega, \mathcal{R}(M), p) \neq \emptyset$, i.e. that the system is unsafe. It can happen that there exists $\nu \in Discloser(\Omega, R^\sharp, p)$ but it is not always possible to decide whether $\nu \in p(\mathcal{R}(M))$ and then to exhibit ν as a counterexample. But unlike the use of overapproximations for verifying safety properties for example, $Discloser(\Omega, R^\sharp, p) = \emptyset$ does not imply neither that $Discloser(\Omega, \mathcal{R}(M), p) = \emptyset$, i.e. that the system is safe. Indeed, the equivalence classes of a given run being larger in $R^\sharp$, it is less likely that all the runs of a class satisfies the same atomic proposition. But $Discloser(\Omega, \mathcal{R}(M), p) = \emptyset$ also means that for all $r \in R^\sharp$, $(R^\sharp, r) \models P\neg\phi$. Then we know that no attacker reasoning from a superset of $R^\sharp$ will be able to information about Ω since the equivalence classes will be greater that the ones of $\theta^\sharp$ in that case. Nevertheless, there are situations where an attacker reasoning with $R^\sharp$ may be able to acquire sound information about Ω. In the sequel, we consider the case of one secret $\phi \in \Omega$.

Proposition 5. *Let $r \in \mathcal{R}(M)$. If $(R^\sharp, r) \models K\phi$ then $(\mathcal{R}(M), r) \models K\phi$.*

Proof. Let $r \in \mathcal{R}(M)$, such that $(R^\sharp, r) \models K\phi$. Then, for all $r' \in [r]_{\theta^\sharp}$, $lst(r') \in F(\phi)$. In particular, since $[r]_\theta \subseteq [r]_{\theta^\sharp}$, if $r' \in [r]_\theta$ then $lst(r') \in F(\phi)$. We conclude that $(\mathcal{R}(M), r) \models K\phi$.

This implies that $\mathcal{A}^\sharp$ can acquire sound information from a run r such that $(R^\sharp, r) \models K\phi$ when $\mathcal{A}^\sharp$ knows that this run belongs to $\mathcal{R}(M)$. It is especially the case when $\mathcal{A}^\sharp$ reasons on the basis of observed traces of M. The following proposition suggest how the proposition 5 can be applied for the synthesis of monitor: computing an automaton accepting the language $Discloser(\phi, R^\sharp, p)$ can be used to monitor information flow on the observed traces of M, exhibiting sound counterexamples:

Proposition 6. *For all $\nu \in p(\mathcal{R}(M))$, such that $\nu \in Discloser(\phi, M, \theta^\sharp)$ we can conclude that $\nu \in Discloser(\phi, M, \theta)$.*

Proposition 7. *Let $r \in R^\flat$. If $(R^\flat, r) \models P\neg\phi$ then $(\mathcal{R}(M), r) \models P\neg\phi$.*

The main implication of this result is that $Safe(\phi, R^\flat, p) \subseteq Safe(\phi, \mathcal{R}(M), p)$. But, $\mathcal{A}^\flat$ cannot say anything about an observation $\nu \in p(\mathcal{R}(M))$ as long as we do

not know whether $\nu \notin Safe(\phi, R^\flat, p)$. Nevertheless, proving by some method that the other inclusion $Safe(\phi, \mathcal{R}(M), p) \subseteq Safe(\phi, R^\flat, p)$ holds provides a sufficient condition for ϕ to be θ-opaque on M.

Finally, considering together $R^\flat$ and $R^\sharp$, we can statically exhibit a counterexample:

Proposition 8. *If there exists $r \in R^\flat$ such that $(R^\sharp, r) \models K\phi$, then $(\mathcal{R}(M), r) \models K\phi$.*

We will see now how this considerations can be used using an abstract interpretation framework based on Galois connection.

4 Construction of monitors using Galois Connections

In this section, we will investigate the computation of the monitors suggested at the end of the previous section. We still consider the case LTS as before and one secret $\phi \in \Omega$.

Let $(Q^\sharp, \sqsubseteq^\sharp, \sqcap^\sharp, \sqcup^\sharp, \top^\sharp, \bot^\sharp)$ be a finite lattice that will used to represent overapproximations of the sets of states of S. Let $(\Sigma^\sharp, \preceq^\sharp)$ be another finite lattice representing overapproximations of the sets of events of Λ. Let $\mathcal{P}(S) \underset{\alpha^\sharp}{\overset{\gamma^\sharp}{\leftrightarrows}} Q^\sharp$ and $\mathcal{P}(\Lambda) \underset{a^\sharp}{\overset{c^\sharp}{\leftrightarrows}} \Sigma^\sharp$ be two Galois connections. We assume that the second connection is consistent with the observation, in the sense that the observable events of Λ_o are not assimilated by abstraction with unobservable events. Formally, we suppose that for all $X \subseteq \Lambda_0$, $c^\sharp \circ a^\sharp(X) \subseteq \Lambda_0$. This is equivalent to impose that $c^\sharp \circ a^\sharp(\Lambda_o) = \Lambda_o$, and also equivalent to $c^\sharp \circ a^\sharp(\Lambda_{uo}) = \Lambda_{uo}$. We will use the notation $a^\sharp(\lambda)$ for $a^\sharp(\{\lambda\})$ to denote the abstract representation of the event λ, and we extend this notation to words by $a^\sharp(\nu) = a^\sharp(\lambda_1)a^\sharp(\lambda_2)\dots a^\sharp(\lambda_n)$ if $\nu = \lambda_1\lambda_2\dots\lambda_n$. We note $\Sigma_o^\sharp = a^\sharp(\Lambda_o)$ and $\Sigma_{uo}^\sharp = a^\sharp(\Lambda_{uo})$.

We suppose also that $\alpha^\sharp \circ \gamma^\sharp = id_{Q^\sharp}$ as it usually the case in abstract interpretation. Let $post^\sharp : \Sigma^\sharp \to (Q^\sharp \to Q^\sharp)$ be a sound approximation of the operator $post$ defined in section 3. This abstraction is sound in the sense that for all $B \subseteq \Lambda$ and $X \subseteq S$:

$$post(B)(X) \subseteq \gamma^\sharp \circ post^\sharp(a^\sharp(B)) \circ \alpha^\sharp(X)$$

Which means that the set of states that are reachable according to $reach$ are also reachable according to $reach^\sharp$. We define also the operator:

$$\begin{aligned} reach^\sharp : \Sigma^\sharp &\to (Q^\sharp \to Q^\sharp) \\ \sigma &\mapsto \quad q \quad \mapsto lfp(q' \mapsto q \sqcup^\sharp post^\sharp(\sigma)(q')) \end{aligned}$$

Similarly to $reach_{uo}$, we define $reach_{uo}^\sharp : q \mapsto reach^\sharp(\Sigma_{uo}^\sharp)(q)$. We define the finite automaton $G^\sharp = (\Sigma_o^\sharp, Q^\sharp, \Delta^\sharp, q_0^\sharp, F^\sharp)$ by $q_0^\sharp = reach_{uo}^\sharp \circ \alpha^\sharp(S_0)$, $F^\sharp = \{q \in Q^\sharp : \gamma^\sharp(q) \subseteq F(\phi)\}$ and

$$\begin{aligned} \Delta^\sharp : \Sigma^\sharp \times Q^\sharp &\to Q^\sharp \\ \sigma, q \quad &\mapsto reach_{uo}^\sharp \circ post^\sharp(\sigma)(q) \end{aligned}$$

With this definition, the automaton $G^\sharp$ is an abstraction of the automaton $det_o(M)$ defined in 6 and used to verify opacity. The main benefit of this approach is that the

computation of $reach^\sharp$ always terminates whereas it may not be the case for $reach$. As in the definition of 6, $\Delta^\sharp$ is extended to words on $\Sigma_o^\sharp$.

Proposition 9. *For all $\nu \in \Lambda_o^*$ and $X \subseteq S$, $\Delta(\nu, X) \subseteq \gamma^\sharp \circ \Delta^\sharp(a^\sharp(\nu), \alpha^\sharp(X))$.*

Proof. The operators $post$, $reach_{uo}$, $post^\sharp$ and $reach_{uo}^\sharp$ are monotone. Let $\lambda \in \Lambda_o$ and $X \subseteq S$.

$$\begin{aligned}
\Delta(\lambda, X) &= reach_{uo} \circ post(\lambda)(X) \\
&\subseteq reach_{uo} \circ \gamma^\sharp \circ post^\sharp(a^\sharp(\lambda)) \circ \alpha^\sharp(X) \\
&\subseteq \gamma^\sharp \circ reach_{uo}^\sharp \circ \alpha^\sharp \circ \gamma^\sharp \circ post^\sharp(a^\sharp(\lambda)) \circ \alpha^\sharp(X) \\
&\subseteq \gamma^\sharp \circ reach_{uo}^\sharp \circ post^\sharp(a^\sharp(\lambda)) \circ \alpha^\sharp(X) \\
&\subseteq \gamma^\sharp \circ \Delta^\sharp(a^\sharp(\lambda), \alpha^\sharp(X))
\end{aligned}$$

Now, let $\nu = \lambda_1 \lambda_2 \ldots \lambda_n \in \Lambda_o^*$.

$$\begin{aligned}
\Delta(\nu, X) &= \Delta(\lambda_n, \cdot) \circ \ \ldots \ \circ \Delta(\lambda_2, \cdot) \circ \Delta(\lambda_1, \cdot)(X) \\
&\subseteq \Delta(\lambda_n, \cdot) \circ \ \ldots \ \circ \Delta(\lambda_2, \cdot) \circ \gamma^\sharp \circ \Delta^\sharp(a^\sharp(\lambda_1), \cdot) \circ \alpha^\sharp(X) \\
&\subseteq \gamma^\sharp \circ \Delta^\sharp(a^\sharp(\lambda_n), \cdot) \circ \ \ldots \ \circ \alpha^\sharp \circ \gamma^\sharp \circ \Delta^\sharp(a^\sharp(\lambda_1), \cdot) \circ \alpha^\sharp(X) \\
&\subseteq \gamma^\sharp \circ \Delta^\sharp(a^\sharp(\lambda_n), \cdot) \circ \ \ldots \ \circ \Delta^\sharp(a^\sharp(\lambda_2), \cdot) \circ \Delta^\sharp(a^\sharp(\lambda_1), \cdot) \circ \alpha^\sharp(X) \\
&\subseteq \gamma^\sharp \circ \Delta^\sharp(a^\sharp(\nu), \alpha^\sharp(X))
\end{aligned}$$

Theorem 3. *For all $\nu \in p(\mathcal{R}(M))$ such that $a^\sharp(\nu) \in \mathcal{L}(G^\sharp, q_0^\sharp, F^\sharp)$, we can deduce that $\nu \in Discloser(\phi, \mathcal{R}(M), p)$.*

Proof. Let $r \in \mathcal{R}(M)$ such that $p(r) = \nu$. Since $a^\sharp(\nu) \in \mathcal{L}(G^\sharp, q_0^\sharp, F^\sharp)$, $\Delta^\sharp(a^\sharp(\nu), q_0^\sharp) \in F^\sharp$, which means that $\gamma^\sharp(\Delta^\sharp(a^\sharp(\nu), q_0^\sharp)) \subseteq F(\phi)$. According to proposition 9, $\Delta(\nu, X_0) \subseteq \gamma^\sharp \circ \Delta^\sharp(a(\nu), \alpha^\sharp(X_0))$ and then $\Delta(\nu, X_0) \subseteq \gamma^\sharp \circ \Delta^\sharp(a(\nu), q_0^\sharp)$. Also $\alpha^\sharp(X_0) = \alpha^\sharp(reach_{uo}(S_0))$ so $\alpha^\sharp(X_0) \subseteq reach_{uo}^\sharp \circ \alpha^\sharp(S_0) = q_0^\sharp$. Finally $\Delta(\nu, s_0) \subseteq F(\phi)$ and then, according to theorem 1, $r \models \phi$.

This result provides an effective methodology to monitor information flow as soon as the two Galois connections are given. Practically, $\mathcal{A}^\sharp$ observes a trace ν from M and when $a^\sharp(\nu)$ is accepted by $G^\sharp$, then $\mathcal{A}^\sharp$ knows that ϕ is true on the current run executed in M.

Conclusion

The main contribution of this article is to provide analysis methods for the opacity property when considering an attacker reasoning on a approximate model, greater or smaller than the real one. The reformulation of the definition of opacity using epistemic logic modalities allowed us to integrate the notion of approximation within the model of attacker, through the parametrisability of the underlying equivalence relation

over the runs. Then, we could derive a monitoring methodology in the case of abstraction given as Galois connections. Other approaches with similar objectives have been proposed in [21,7] and it would be interesting to investigate the connections between them. In [11,10], the authors presented an application of the supervisory control theory to ensure opacity on finite models. It would be useful to investigate how this synthesis problem can be defined for infinite models and attackers reasoning on sound approximations. Also, we introduced temporal modalities in order to differentiate their formulation from the less classical epistemic ones. But, it would be interesting to consider more dynamical aspect, especially for the evolution of attackers' knowledge trough time. The purpose would be then to formulate other security properties like information integrity for example. Since system with security requirements inherently involves a multi-agent environment, adding epistemic modalities to the logic ATL [1] seems to be a promising approach. Thus, we should be able to express in a unified way properties like the existence of an attack scenario (i.e. non-opacity) by $\ll \mathcal{A} \gg \Diamond K\phi$ or the supervisory control problem by $\ll \mathcal{C} \gg \Box\neg K\phi$ where $\mathcal{C}$ denotes a controller.

References

1. Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, Florida, October 1997.
2. Rajeev Alur, Pavol Černý, and Steve Zdancewic. Preserving secrecy under refinement. In *ICALP '06: Proceedings (Part II) of the 33rd International Colloquium on Automata, Languages and Programming*, page 107–118. Springer, 2006.
3. Matt Bishop. *Introduction to computer security*. Addison-Wesley Professional, 2004.
4. Jeremy Bryans, Maciej Koutny, Laurent Mazaré, and Peter Y. A. Ryan. Opacity generalised to transition systems. *International Journal of Information Security*, 7(6):421–435, 2008.
5. Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, page 238–252, Los Angeles, California, 1977. ACM Press, New York, NY.
6. Patrick Cousot and Radhia Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, August 1992.
7. Mila Dalla Preda, Roberto Giacobazzi, Matias Madou, and Koen De Bosschere. Opaque predicates detection by abstract interpretation. In *Proc. of the 11th International Conference on Algebraic Methodology and Software Technology* (*AMAST '06*), volume 4019 of *Lecture Notes in Computer Science*, page 81–95. Springer-Verlag, 2006.
8. Martin De Wulf, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Antichains: A new algorithm for checking universality of finite automata. In *Proceedings of CAV 2006: Computer-Aided Verification*, Lecture Notes in Computer Science 4144, page 17–30. Springer-Verlag, 2006.
9. Martin De Wulf, Laurent Doyen, and Jean-François Raskin. A lattice theory for solving games of imperfect information. In *Proceedings of HSCC 2006: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 3927, page 153–168. Springer-Verlag, 2006.
10. Jérémy Dubreil, Philippe Darondeau, and Hervé Marchand. Opacity enforcing control synthesis. In B. Lennartson, M. Fabian, K. Akesson, A. Giua, and R. Kumar, editors, *Proceedings of the 9th International Workshop on Discrete Event Systems (WODES'08)*, page 28–35, Göteborg, Sweden, May 2008. IEEE.

11. Jérémy Dubreil, Philippe Darondeau, and Hervé Marchand. Supervisory control for opacity. Technical Report 1921, IRISA, February 2009.
12. Jérémy Dubreil, Thierry Jéron, and Hervé Marchand. Monitoring confidentiality by diagnosis techniques. In *European Control Conference*, Budapest, Hungary, August 2009.
13. Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. The MIT Press, 1995.
14. Riccardo Focardi and Roberto Gorrieri. An information flow security property for ccs. In *Proceedings of Second North American Process Algebra Workshop*, 1993.
15. Riccardo Focardi and Roberto Gorrieri. Classification of security properties: Information flow. In *Foundations of Security Analysis and Design, LNCS No 2171*, page 331–396, 2000.
16. Roberto Giacobazzi and Isabella Mastroeni. Abstract non-interference: parameterizing non-interference by abstract interpretation. In *POPL '04: Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, page 186–197. ACM, 2004.
17. Janice I. Glasgow, Glenn H. MacEwen, and Prakash Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10(3):226–264, 1992.
18. Joseph A. Goguen and José Meseguer. Security policies and security models. In *1982 Berkeley Conference on Computer Security*, page 11–20. IEEE Computer Society, April 1982.
19. Jaakko Hintikka. *Knowledge and Belief*. Cornell university Press, 1962.
20. Thierry Jéron, Hervé Marchand, Sophie Pinchinat, and Marie-Odile Cordier. Supervision patterns in discrete event systems diagnosis. In *Workshop on Discrete Event Systems, WODES'06*, Ann-Arbor (MI, USA), July 2006.
21. Isabella Mastroeni. Deriving bisimulations by simplifying partitions. In *Nineth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'08)*, volume 4905 of *Lecture Notes in Computer Science*, page 157–171. Springer-Verlag, 2008.
22. Laurent Mazaré. Using unification for opacity properties. In *Proceedings of the 4th IFIP WG1.7 Workshop on Issues in the Theory of Security (WITS'04)*, page 165–176, Barcelona (Spain), 2004.
23. Mera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinaamohideen, and Demosthenis Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 1995.
24. Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In *STOC*, page 1–9, 1973.
25. Shigemasa Takai and Yusuke Oka. A formula for the supremal controllable and opaque sublanguage arising in supervisory control. *SICE Journal of Control, Measurement, and System Integration*, 1(4):307–312, March 2008.
26. Edward N. Zalta. *Basic Concepts in Modal Logic*. Stanford University Lecture Notes, 1995. http://mally.stanford.edu/notes.pdf.