School of Computing Science,
University of Newcastle upon Tyne

Opacity Generalised to Transition Systems

Jeremy W. Bryans, Maciej Koutny, Laurent Mazare and
Peter Y. A. Ryan

Technical Report Series

CS-TR-868

November 2004

Published by the University of Newcastle upon Tyne,
School of Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, UK.

Opacity Generalised to Transition Systems

Jeremy W. Bryans[1], Maciej Koutny[1], Laurent Mazaré[2] and Peter Y.A. Ryan[1]

[1] School of Computing Science, University of Newcastle,
Newcastle upon Tyne, NE1 7RU, United Kingdom
{Jeremy.Bryans,Maciej.Koutny,Peter.Ryan}@ncl.ac.uk
[2] Laboratoire VERIMAG
2, av. de Vignates, Gières, France
laurent.mazare@imag.fr

Abstract. Recently, opacity has proved to be a promising technique for describing security properties. Much of the work has been couched in terms of Petri nets. Here, we extend the notion of opacity to the model of labelled transition systems and generalise opacity in order to better represent concepts from the work on information flow. In particular, we establish links between opacity and the information flow concepts of anonymity and non-interference such as non-inference. We also investigate ways of verifying opacity when working with Petri nets. Our work is illustrated by two examples, one describing anonymity in a commercial context, and the other modelling requirements upon a simple voting system.
Keywords: opacity, non-deducibility, anonymity, non-interference, Petri nets, observable behaviour, labelled transition systems, abstract interpretation.

Introduction

The notion of secrecy has been formulated in various ways in the computer security literature. However, two views of security have been developed over the years by two separate communities. The first one starts from the notion of information flow, describing the knowledge an intruder could gain in terms of properties such as non-deducibility or non-interference. The second view was initiated by Dolev and Yao's work and focussed initially on security properties. The idea here is to describe properly the capability of the intruder. Some variants of secrecy appeared, such as strong secrecy, giving more expressivity than the security property but still lacking the expressivity of information flow concepts.

Recently, opacity has proved to be a promising technique for describing security properties. Much of the work has been couched in terms of Petri nets. In this paper, we extend the notion of opacity to the more general framework of labelled transition systems. When using opacity we have fine-grained control over the observation capabilities of the players, and we show one way that these capabilities may be encoded. The essential idea is that a predicate is opaque if an observer of the system will never be able to establish the truth of that predicate.

In the first section, after recalling some basic definitions, we present a generalisation of opacity, and show how this specialises into the three previously defined variants: initial opacity, final opacity and total opacity. In Section 2, we show how opacity is related to previous work in security. We consider how opacity may describe anonymity and non-interference, and discuss it in the context of security protocols. In Section 3, we consider the question of opacity checking, and state a general undecidability result for opacity. After restricting ourselves to Petri nets, we give some decidability and undecidability properties. As opacity is undecidable as soon as we consider systems with infinite number of states, we present an approximation technique which may provide a way of model checking even in such cases. Finally, in Section 4, we conclude with two examples. The first, drawn from the commercial world, illustrates how anonymity may be expressed using opacity. The second considers a voting scheme, and shows how the approximation technique might be used.

The contributions of this paper are therefore: we present a general theory of opacity in the context of labelled transition systems, which allows us to compare our work with other work in the security community, and also unify the work already done with opacity. We also prove a number of (un)decidability results, and present a technique which may allow model checking even though the problem at hand is in general undecidable.

1 Basic Definitions

The set of finite sequences over a set A will be denoted by A^*, and the empty sequence by ϵ. The length of a finite sequence λ will be denoted by $len(\lambda)$, and its projection onto a set $B \subseteq A$ by $\lambda|_B$.

Definition 1 *A labelled transition system (LTS) is a tuple $\Pi = (S, L, \Delta, S_0)$, where S is the (potentially infinite) set of states, L is the (potentially infinite) set of labels, $\Delta \subseteq S \times L \times S$ is the transition relation, and S_0 is the nonempty (finite) set of initial states. We consider only deterministic LTSs, and so for any transitions $(s, l, s'), (s, l, s'') \in \Delta$, it is the case that $s' = s''$.*
A run of Π is a pair (s_0, λ), where $s_0 \in S_0$ and $\lambda = l_1 \ldots l_n$ is a finite sequence of labels such that there are states $s_1, \ldots, s_n$ satisfying (s_{i-1}, l_i, s_i), for $i = 1, \ldots, n$. We will denote the state s_n by $s_0 \oplus \lambda$, and call it reachable from s.
The set of all runs is denoted by $run(\Pi)$, and the language generated by Π is defined as $\mathcal{L}(\Pi) = \{\lambda \mid \exists\, s_0 \in S_0 : (s_0, \lambda) \in run(\Pi)\}$.

Let $\Pi = (S, L, \Delta, S_0)$ be an LTS fixed for the rest of this section, and Θ be a set of elements called *observables*. We will now aim at modelling the different capabilities for observing the system modelled by Π. First, we introduce a general observation function and then, specialise it to reflect limited information about runs available to an observer.

Definition 2 *Any function $obs : run(\Pi) \to \Theta^*$ is an observation function. It is called label-based and: static / dynamic / orwellian / m-orwellian ($m \geq 1$) if respectively the following hold (below $\lambda = l_1 \ldots l_n$):*

- *static: there is a mapping* $obs' : L \to \Theta \cup \{\epsilon\}$ *such that for every run* (s, λ) *of* Π, $obs(s, \lambda) = obs'(l_1) \dots obs'(l_n)$.
- *dynamic: there is a mapping* $obs' : L \times L^* \to \Theta \cup \{\epsilon\}$ *such that for every run* (s, λ) *of* Π, $obs(s, \lambda) = obs'(l_1, \epsilon) obs'(l_2, l_1) \dots obs'(l_n, l_1 \dots l_{n-1})$.
- *orwellian: there is a mapping* $obs' : L \times L^* \to \Theta \cup \{\epsilon\}$ *such that for every run* (s, λ) *of* Π, $obs(s, \lambda) = obs'(l_1, \lambda) \dots obs'(l_n, \lambda)$.
- *m-orwellian: there is a mapping* $obs' : L \times L^* \to \Theta \cup \{\epsilon\}$ *such that for every run* (s, λ) *of* Π, $obs(s, \lambda) = obs'(l_1, \kappa_1) \dots obs'(l_n, \kappa_n)$, *where for* $i = 1, \dots, n$, $\kappa_i = l_{max\{1,i-m+1\}} l_{\max\{1,i-m+1\}+1} \cdots l_{\min\{n,i+m-1\}}$.

In each of the above four cases, we will often use $obs(\lambda)$ *to denote* $obs(s, \lambda)$.

Note that allowing obs' to return ϵ allows one to model invisible actions. The different kinds of observable functions reflect different computational power of the observers. Static functions correspond to an observer which always interprets the same executed label in the same way. Dynamic functions correspond to an observer which has potentially infinite memory to store labels, but can only use knowledge of previous labels to interpret a label. Orwellian functions correspond to an observer which has potentially infinite memory to store labels, and can use knowledge (either subsequent or previous) of other labels to (re-)interpret a label. m-orwellian functions are a restricted version of the last class where the observer can store only a bounded number of labels. Static functions are nothing but 1-orwellian ones; static functions are also a special case of dynamic functions; and both dynamic and m-orwellian are a special case of orwellian functions.

It is possible to define state-based observation functions. For example, a state-based static observation function obs is one for which there is $obs' : S \to \Theta \cup \{\epsilon\}$ such that for every run $(s, l_1 \dots l_1)$, we have $obs(s, l_1 \dots l_n) = obs'(s) obs'(s \oplus l_1) \dots obs'(s \oplus l_1 \dots l_n)$.

Let us consider an observation function obs. We are interested in whether an observer can establish a property ϕ (a predicate over system states and traces) for some run having only access to the result of the observation function. We will identify ϕ with its characteristic set: the set of runs for which it holds.

Now, given an observed execution of the system, we would want to find out whether the fact that the underlying run belongs to ϕ can be deduced by the observer (note that we are not interested in establishing whether the underlying run does not belong to ϕ; to do this, we would rather consider the property $\overline{\phi} = run(\Pi) \setminus \phi$).

What it means to deduce a property can mean different things depending on what is relevant or important from the point of view of real application. Below, we give a general formalisation of opacity and then specialise it in three different ways.

Definition 3 *A predicate* ϕ *over* $run(\Pi)$ *is opaque w.r.t. the observation function obs if, for every run* $(s, \lambda) \in \phi$, *there is a run* $(s', \lambda') \notin \phi$ *such that* $obs(s, \lambda) = obs(s', \lambda')$. *Moreover,* ϕ *is called: initial-opaque / final-opaque / total-opaque if respectively the following hold:*

- *there is a predicate* ϕ' *over* S_0 *such that for every run* (s, λ) *of* Π*, we have* $\phi(s, \lambda) = \phi'(s)$.
- *there is a predicate* ϕ' *over* S *such that for every run* (s, λ) *of* Π*, we have* $\phi(s, \lambda) = \phi'(s \oplus \lambda)$.
- *there is a predicate* ϕ' *over* S^* *such that for every run* $(s, l_1 \ldots l_n)$ *of* Π*, we have* $\phi(s, l_1 \ldots l_n) = \phi'(s, s \oplus l_1, \ldots, s \oplus l_1 \ldots l_n)$.

In the first of above three cases, we will often write $s \in \phi$ *whenever* $(s, \lambda) \in \phi$.

Initial-opacity has been illustrated by the dining cryptographers example (in [4] with two cryptographers and [3] with three). It would appear that it is suited to modelling situations in which initialisation information such as crypto keys, etc., needs to be kept secret. More generally, situations in which confidential information can be modelled in terms of initially resolved non-determinism can be captured in this way. Final-opacity models situations where the final result of a computation needs to be secret. Total-opacity is a generalisation of the two other properties asking not only the result of the computation and its parameters to be secret but also the states visited during computation.

Proposition 1. *Let* ϕ *and* ϕ' *be two predicates over* $run(\Pi)$*. If* ϕ *is opaque w.r.t. an observation function obs and* $\phi' \Rightarrow \phi$*, then* ϕ' *is opaque w.r.t. obs.*

Proof. Follows directly from definitions. ∎

2 Opacity in Security

The goal of this section is to show how our notion of opacity relates to other concepts commonly used in the formal security community. We will compare opacity to forms of anonymity and non-interference, as well as discuss its application to security protocols.

2.1 Anonymity

Anonymity is concerned with the preservation of secrecy of identity through the obscuring of the actions of that identity. It is a function of the behaviour of the underlying (anonymising) system, as well as being dependent on capability of the observer.

The static, dynamic and orwellian forms of observation function presented in Definition 2 model three different strengths of observer. We now introduce two observation functions needed to render anonymity in terms of suitable opacity properties.

Let $\Pi = (S, L, \Delta, S_0)$ be an LTS fixed for the rest of this section, and $A = \{a_1, \ldots, a_n\} \subseteq L$ be a set of labels over which anonymity is being considered. Moreover, let $\alpha, \alpha_1, \ldots, \alpha_n \notin L$ be fresh labels.
The first observation function, obs^s_A, is static and defined so that $obs^s_A(\lambda)$ is obtained from λ by replacing each occurrence of a_i by α. The second observation

function, obs^d_A, is dynamic and defined thus: let $a_{i_1}, \ldots, a_{i_q}$ $(q \geq 0)$ be all the distinct labels of A appearing within λ listed in the (unique) order in which they appeared for the first time in λ; then $obs(\lambda)$ is obtained from λ by replacing each occurrence of a_{i_j} by α_j. For example,

$$obs^s_{\{a,b\}}(acdba) = \alpha cd\alpha\alpha \quad \text{and} \quad obs^d_{\{a,b\}}(acdba) = \alpha_1 cd\alpha_2\alpha_1.$$

Strong anonymity In [18], a definition of strong anonymity is presented for the process algebra CSP. In our (LTS) context, this definition translates as follows.

Definition 4 *Π is strongly anonymous w.r.t. A if $\mathcal{L}(\Pi) = \mathcal{L}(\Pi')$, where Π' is obtained from Π by replacing each transition (s, a_i, s') with n transitions: $(s, a_1, s'), \ldots, (s, a_n, s')$.*

In our framework, we have that

Definition 5 *Π is O-anonymous w.r.t. A if, for every sequence $\mu \in A^*$, the predicate ϕ_μ over the runs of Π defined by*

$$\phi_\mu(s, \lambda) = \big(len(\lambda|_A) = len(\mu) \wedge \lambda|_A \neq \mu\big)$$

is opaque w.r.t. obs^s_A.

We want to ensure that every possible sequence μ (with appropriate length restrictions) of anonymised actions is a possible sequence within the LTS. In Definition 5 above, the opacity of the predicate ϕ_μ ensures that the sequence μ is a possible history of anonymised actions, because it is the only sequence for which the predicate ϕ_μ is false, and so ϕ_μ can only be opaque if μ is a possible sequence.

Theorem 1. *Π is O-anonymous w.r.t. A* iff *it is strongly anonymous w.r.t. A.*

Proof. We first observe that the strong anonymity w.r.t. A is equivalent to

$$\{\lambda' \in L^* \mid \exists\, \lambda \in \mathcal{L}(\Pi) : obs^s_A(\lambda') = obs^s_A(\lambda)\} \subseteq \mathcal{L}(\Pi)\ . \tag{1}$$

We show that Π is O-anonymous w.r.t. A iff (1) holds.

($\Longrightarrow$) Suppose that $\lambda \in \mathcal{L}(\Pi)$ and $\lambda' \in L^*$ are such that $obs^s_A(\lambda') = obs^s_A(\lambda)$. Clearly, if $\lambda'|_A = \lambda|_A$ then $\lambda' = \lambda \in \mathcal{L}(\Pi)$, so we assume that $\lambda'|_A \neq \lambda|_A$. Then, for some $s \in S_0$, we have that $\phi_\mu(s, \lambda)$ holds, where $\mu = \lambda'|_A$. Hence, by the opacity of ϕ_μ w.r.t. obs^s_A, there is $(s', \lambda'') \in run(\Pi)$ such that $obs^s_A(\lambda'') = obs^s_A(\lambda)$ and $\phi_\mu(s', \lambda'')$ does not hold. Thus $\lambda' = \lambda'' \in \mathcal{L}(\Pi)$. As a result, (1) holds.

($\Longleftarrow$) Suppose that $\mu \in A^*$ and $\phi_\mu(s, \lambda)$ holds. Then $len(\lambda|_A) = len(\mu)$ and $\lambda|_A \neq \mu$. Let $\lambda' \in L^*$ be the unique sequence such that $\lambda'|_A = \mu$ and $obs^s_A(\lambda') = obs^s_A(\lambda)$. By (1), there is s' such that $(s', \lambda') \in run(\Pi)$. Clearly, $obs^s_A(\lambda') = obs^s_A(\lambda)$ and $\phi_\mu(s', \lambda')$ does not hold. As a result, ϕ_μ is opaque w.r.t. obs^s_A, and so Π is O-anonymous w.r.t. A. ∎

Weak anonymity A natural extension of strong anonymity is *weak anonymity*[3]. This models easily the notion of *pseudo-anonymity*: actions performed by the same party can be correlated, but the identity of the party cannot be determined.

Definition 6 *Π is weakly anonymous w.r.t. A if $\pi(\mathcal{L}(\Pi)) \subseteq \mathcal{L}(\Pi)$, for every permutation π over the set A.*

In our framework, we have that

Definition 7 *Π is weak-O-anonymous if, for every sequence $\mu \in A^*$, the predicate ϕ_μ over the runs of Π introduced in Definition 5 is opaque w.r.t. obs^d_A.*

Theorem 2. *Π is weak-O-anonymous w.r.t. A* iff *it is weak-anonymous w.r.t. A.*

Proof. We first observe that $obs^d_A(\lambda) = obs^d_A(\lambda')$ *iff* there is a permutation π over the set A such that $\pi(\lambda) = \lambda'$, and so showing weak anonymity w.r.t. A is equivalent to showing that

$$\{\lambda' \in L^* \mid \exists\, \lambda \in \mathcal{L}(\Pi) : obs^d_A(\lambda') = obs^d_A(\lambda)\} \subseteq \mathcal{L}(\Pi) .$$

The proof then follows similar lines to that of Theorem 1, with obs^d_A playing the role of obs^s_A. ∎

Other observation functions Dynamic observation functions can model for example the *downgrading* of a channel. Before the downgrade nothing can be seen, after the downgrade the observer is allowed to see all transmissions on that channel. A suitable formulation would be as follows.

Suppose that A represents the set of all possible messages on a confidential channel, and $\delta \in L \setminus A$ represents an action of downgrading that channel. Then $obs(\lambda)$ is obtained from λ by deleting each occurrence of a_i which is preceded (directly or indirectly) by an occurrence of δ. In other words, if the downgrade action appears earlier in the run, then the messages on the channel are observed in the clear, otherwise nothing is observed.

Orwellian observation functions can model conditional or escrowed anonymity, where someone can be anonymous when they initially interact with the system, but some time in the future their identity can be revealed, as outlined below.

Suppose that there are n identities Id_i, each identity being capable of performing actions represented by $a_i \in A$. Moreover, $\alpha \notin L$ represents the encrypted observation of any of these actions, and $\rho_i \in L \setminus A$ represents the action of identity Id_i being revealed. Then $obs(\lambda)$ is obtained from λ by replacing each occurrence of a_i by α, provided that ρ_i never occurs within λ.

[3] We believe that this formulation of weak anonymity was originally due to Ryan and Schneider.

2.2 Non-Interference

Opacity can be linked to a particular formulation of non-interference. A discussion of non-interference can be found in [9] and [20]. The basic idea is that labels are split into two sets, *High* and *Low*. *Low* labels are visible by anyone, whereas *High* labels are private. Then, a system is non-interfering if it is not possible for an outside observer to gain any knowledge about the presence of *High* labels in the original run (the observer only sees *Low* labels). This notion is in fact a restriction of standard non-interference. It was orginally called non-inference in [14], and is called strong non-deterministic non-interference in [10].

Definition 8 *Π satisfies non-inference if $\mathcal{L}(\Pi)\mid_{Low} \subseteq \mathcal{L}(\Pi)$.*

In other words, for any run (s, λ) of Π, there exists a run (s', λ') such that λ' is λ with all the labels in *High* removed.

The notion of non-interference (and in particular non-inference) is close to opacity as stated by the two following properties. First, we show that it is possible to transform certain initial opacity properties into non-inference properties.

Proposition 2. *Any initial opacity problem involving static observation function can be reduced to a non-inference problem.*

Proof. Let $\Pi = (S, L, \Delta, S_0)$ be an LTS, *obs* defined through *obs*$'$ (see Definition 2) be a static observation function, and ϕ defined through ϕ' (see Definition 3) be an initial opacity predicate.
We construct a new LTS $\Pi' = (S', L', \Delta', S'_0)$ such that:

- $S' = S \cup \{s' \mid s \in S_0\}$ where each s' is a fresh state.
- $L' = obs'(L) \cup \{h\}$ where $h \notin obs'(L)$ is a fresh label.
- Δ' is obtained from Δ by replacing each $(s, l, r) \in \Delta$ by $(s, obs'(l), r)$, and adding, for each $s \in S_0$, a new transition (s', h, s).
- $S'_0 = \{s \mid s \in S_0 \setminus \phi\} \cup \{s' \mid s \in S_0 \cap \phi\}$.

We then consider a non-inference problem for Π' with $Low = obs'(L)$ and $High = \{h\}$, and below we show that Π' satisfies non-inference *iff* for Π the opacity property ϕ w.r.t. *obs* holds. We assume that Π' is deterministic; otherwise we replace it by its deterministic version.

($\Longrightarrow$) Suppose that $(s, \lambda) \in run(\Pi) \cap \phi$. Then $(s', h\ obs(\lambda)) \in run(\Pi')$. Thus, by non-inference of Π', there is $(r, \kappa) \in run(\Pi')$ such that $obs(\lambda) = \kappa$ and $r \in S_0 \setminus \phi$. Hence there is $(r, \mu) \in run(\Pi)$ such that $obs(\mu) = \kappa = obs(\lambda)$. Consequently, the opacity of ϕ w.r.t. *obs* holds.

($\Longleftarrow$) Suppose that $(r, \kappa) \in run(\Pi')$ and $\kappa|_{Low} \neq \kappa$. Then $\kappa = h\rho$ and $r = s'$, for some ρ and $s \in S_0 \cap \phi$. Hence there is $(s, \lambda) \in run(\Pi)$ such that $obs(\lambda) = \rho$ and, by the opacity of ϕ holding for *obs*, there is $(r, \psi) \in run(\Pi)$ such that $r \notin \phi$ and $obs(\psi) = obs(\lambda)$. In turn, this means that $(r, obs(\psi)) \in run(\Pi')$. We finally have $\kappa|_{Low} = \rho = obs(\lambda) = obs(\psi)$, and so Π' satisfies non-inference. ∎

A kind of converse result also holds, in the sense that one can transform any non-inference property to a general opacity property.

Proposition 3. *Any non-inference problem can be reduced to an opacity problem.*

Proof. Let $(S, High \cup Low, \Delta, S_0)$ be an LTS. We define ϕ as a predicate over $run(\Pi)$ so that $\phi(s, \lambda)$ holds *iff* $\lambda|_{Low} \neq \lambda$. Moreover, *obs* is defined as a static observation function such that $obs(\lambda) = \lambda|_{Low}$. Below we show that Π satisfies non-inference *iff* for Π the opacity property ϕ w.r.t. *obs* holds.

($\Longrightarrow$) Suppose that $(s, \lambda) \in run(\Pi) \cap \phi$. Then $\lambda|_{Low} \neq \lambda$ and so, by the non-inference of Π, there is s' such that $(s', \lambda|_{Low}) \in run(\Pi)$. Clearly, $(s', \lambda|_{Low}) \notin \phi$ and $obs(s', \lambda|_{Low}) = obs(s, \lambda)$, since $(\lambda|_{Low})|_{Low} = \lambda|_{Low}$. As a result, the opacity of ϕ w.r.t. *obs* holds.

($\Longleftarrow$) Suppose that $(s, \lambda) \in run(\Pi)$ and $\lambda|_{Low} \neq \lambda$. Then $(s, \lambda) \in run(\Pi) \cap \phi$ and so, by the the opacity property ϕ w.r.t. *obs*, there is $(s', \lambda') \in run(\Pi) \setminus \phi$ such that $obs(\lambda) = obs(\lambda')$. Thus $\lambda'|_{Low} = \lambda'$ and $\lambda'|_{Low} = \lambda|_{Low}$. Hence $\lambda|_{Low} \in \mathcal{L}(\Pi)$. As a result, Π satisfies non-inference. ∎

Non-interference in general makes a distinction between public (*Low*) and private (*High*) messages, and any revelation of a high message breaks the non-interference property. We believe that the ability to fine-tune the *obs* function may make opacity better suited to tackling the problem of *partial information flow*, where a message could provide some partial knowledge and it may take a collection of such leakages to move the system into a compromised state.

2.3 Security Protocols

Opacity was introduced in the context of security protocols in [13]. With one restriction, the current version of opacity is still applicable to protocols. Namely, since we require the number of initial states to be finite, the initial choices made by the various honest agents must come from bounded sets.

To formalise opacity for protocols in the present framework, labels will be *messages* defined by the simple grammar

$$m ::= a \mid \langle m, m \rangle \mid \{m\}_m$$

where a ranges over a set A of *atomic* messages; $\langle m_1, m_2 \rangle$ represents the pairing (concatenation) of messages m_1 and m_2; and $\{m_1\}_{m_2}$ is the encoding of message m_1 using message m_2. A subset K of A is the set of *keys*, each key k in K having an inverse denoted by k^{-1}. The notation $E \vdash m$, where m is a message and E is a finite set of messages (environment), comes from Dolev-Yao theory [7] and denotes the fact that m is deducible from E.

Two messages, m_1 and m_2, are *similar* for environment E *iff* $E \vdash m_1 \sim m_2$ where $\sim$ is the smallest (w.r.t. set inclusion) binary relation satisfying the following:

$$\frac{a \in Atoms}{a \sim a} \qquad \frac{u_1 \sim u_2 \quad v_1 \sim v_2}{\langle u_1, v_1 \rangle \sim \langle u_2, v_2 \rangle} \qquad \frac{E \vdash k^{-1} \quad u \sim v}{\{u\}_k \sim \{v\}_k} \qquad \frac{\neg\, E \vdash k^{-1} \quad \neg\, E \vdash k'^{-1}}{\{u\}_k \sim \{v\}_{k'}}$$

In other words, messages are similar if it is not feasible for an intruder to distinguish them using the knowledge E. Such a notion was introduced in [2], where it was shown to be sound in the computational model, and its generalisation including the case of equational theories appears in [1].

To state which part of a message is visible from the outside, we will use the notion of a *pattern* [2], which adds a new message $\square$ to the above grammar, representing undecryptable messages. Then, $pattern(m, E)$ is the accessible skeleton of m using messages in E as knowledge and $E \vdash m_1 \sim m_2 \Leftrightarrow pattern(m_1, E) = pattern(m_2, E)$. It is defined thus:

$$\begin{aligned} pattern(a, E) &= a \\ pattern(\langle m_1, m_2 \rangle, E) &= \langle pattern(m_1, E), pattern(m_2, E) \rangle \\ pattern(\{m_1\}_{m_2}, E) &= \begin{cases} \{pattern(m_1)\}_{m_2} & \text{if } E \vdash m_2 \\ \square & \text{otherwise .} \end{cases} \end{aligned}$$

To simplify the presentation, we assume that a security protocol is represented by an LTS $\Pi = (S, L, \Delta, S_0)$ (for protocols semantics, see [11]). As protocols are commonly interested in initial opacity (opacity on the value of one of the parameter, e.g., a vote's value), the predicate ϕ will be a suitable subset of S_0. The observation function *obs* will be orwellian with $obs(l_i, \lambda) = pattern(l_i, E)$, where E is the set of messages appearing in λ. (note that, in the case of a bounded protocol, an m-orwellian function will be sufficient). Then, opacity of ϕ w.r.t. *obs* is equivalent to the concept introduced in [13].

3 Opacity Checking

Opacity is a very general concept and many instantiations of it are undecidable. This is even true when LTSs are finite. We will formulate such a property as Proposition 5 (part 4), but first we state a general non-decidability result.

Proposition 4. *Opacity is undecidable.*

Proof. We will show that the reachability problem for Turing machines is reducible to (final) opacity. Let TM be a Turing machine and s be its (non-initial) state. We construct an instance of the final opacity as follows: Π is given by the operational semantics of TM, the observation function *obs* is constant, and ϕ returns true *iff* the final state of a run is different from s. Since s is reachable in TM *iff* ϕ is final opaque w.r.t. *obs*, opacity is undecidable. ∎

It follows from the above proposition that the undecidability of the reachability problem for a class of machines generating LTSs renders opacity undecidable. We will therefore restrict ourselves to Petri nets, a rich model of computation in which the reachability problem is still decidable [17]. Furthermore, Petri nets are well-studied structures and there is a wide range of tools and algorithms for their verification.

3.1 Petri Nets

We will use Petri nets with weighted arcs [17], and give their operational semantics in terms of *transition sequences*.[4] Note that this varies slightly from the one used in [4] where the *step sequence* semantics allowed multiple transitions to occur simultaneously. Here, transitions are clearly separated.

A (weighted) *net* is a triple $N = (P, T, W)$ such that P and T are disjoint finite sets, and $W : (T \times P) \cup (P \times T) \to \mathbb{N}$. The elements of P and T are respectively the *places* and *transitions*, and W is the *weight function* of N. In diagrams, places are drawn as circles, and transitions as rectangles. If $W(x, y) \geq 1$ for some $(x, y) \in (T \times P) \cup (P \times T)$, then (x, y) is an *arc* leading from x to y. As usual, arcs are annotated with their weight if this is 2 or more. The *pre-* and *post-multiset* of a transition $t \in T$ are multisets of places, $\textsc{pre}_N(t)$ and $\textsc{post}_N(t)$, respectively given by

$$\textsc{pre}_N(t)(p) = W(p, t) \text{ and } \textsc{post}_N(t)(p) = W(t, p),$$

for all $p \in P$. A *marking* of a net N is a multiset of places. Following the standard terminology, given a marking M of N and a place $p \in P$, we say that p is marked if $M(p) \geq 1$ and that $M(p)$ is the number of tokens in p. In diagrams, M will be represented by drawing in each place p exactly $M(p)$ tokens (black dots). Transitions represent actions which may occur at a given marking and then lead to a new marking. A transition t is *enabled* at a marking M if $M \geq \textsc{pre}_N(t)$. Thus, in order for t to be enabled at M, for each place p, the number of tokens in p under M should at least be equal to the total number of tokens that are needed as an input to t, respecting the weights of the input arcs. If t is enabled at M, then it can be *executed* leading to the marking $M' = M - \textsc{pre}_N(t) + \textsc{post}_N(t)$. This means that the execution of t 'consumes' from each place p exactly $W(p, t)$ tokens and 'produces' in each place p exactly $W(t, p)$ tokens. If the execution of t leads from M to M' we write $M[t\rangle M'$ and call M' *reachable* from M. A *marked Petri net* $\Sigma = (N, S_0)$ comprises a net $N = (P, T, W)$ and a finite set of initial markings S_0. It generates the LTS $\Pi_\Sigma = (S, T, \Delta, S_0)$ where S is the set of all the markings reachable from the markings in S_0, T is the set of labels, and Δ is defined by $(M, t, M') \in \Delta$ if $M[t\rangle M'$. The language of Σ is that of Π_Σ.

In the case of Petri nets, there are still some undecidable opacity problems.

Proposition 5. *The following problems are undecidable for Petri nets:*

1. *Initial opacity when considering a static observation function.*
2. *Initial opacity when considering a state-based static observation function.*
3. *Initial opacity when considering an orwellian observation function even in the case of finite LTSs generated by marked nets.*
4. *Opacity when considering a constant observable function even in the case of finite LTSs generated by a marked nets.*

[4] It should be stressed that the transitions in the Petri net context correspond to the labels rather than arcs in the LTS framework.

Proof. Below we reduce three undecidable problems to suitable variants of opacity, in each case defining a marked Petri net Σ as well as observation function obs and opacity predicate ϕ for the runs of Π_Σ. The first two are related to Petri nets: the language inclusion problem [12], and the reachable markings inclusion problem [16]. The third one is the Post Correspondence Problem (PCP).

(1) Let $\Sigma_i = (N_i, \{M_i\})$, for $i = 1, 2$, be two marked Petri nets. We first construct their isomorphic copies $\Sigma'_i = (N'_i, \{M'_i\})$, for $i = 1, 2$, in such a way that each transition or place x in Σ_i is renamed to (x, i), and $M'_i = \{(s_1, i), \ldots, (s_{k_i}, i)\}$ where $M_i = \{s_1, \ldots, s_{k_i}\}$. Then:

- $\Sigma = (N, \{M'_1, M'_2\})$ is a marked net such that N is the union of N'_1 an N'_2.
- obs is static and given by $obs'(t, i) = t$, for each transition (t, i) in Σ.
- ϕ is true *iff* the first marking of a run is M'_1.

Since the language of Σ_1 is included in that of Σ_2 *iff* ϕ is initial opaque w.r.t. obs, part (1) holds.

(2) Let $\Sigma_i = (N_i, \{M_i\})$, for $i = 1, 2$, and Σ be the three marked Petri nets as in the proof of part (1). Then:

- We modify Σ in such a way that each transition (t, i) is replaced by two identically connected copies, (t', i) and (t'', i). We then add to Σ three fresh places, p', p and p'', and two fresh transitions, u and u', in such a way that their arcs are as follows: $W(p', u) = W(u, p) = W(p, u') = W(u', p'') = 1$, $W(p', (t', i)) = W((t', i), p') = 1$ and $W(p'', (t'', i)) = W((t'', i), p'') = 1$, for each transition (t, i) of Σ. Next, to obtain the initial markings, we add one copy of p' to both M'_1 and M'_2.
- obs is state-oriented and given by $obs'(M) = \{s_1, \ldots, s_m\}$ for any marking $M = \{p, (s_1, i_1), \ldots, (s_m, i_m)\}$, and $obs'(M) = \epsilon$ otherwise. (Note that such an observation function allows one to inspect at most one state of a given run.)
- ϕ is true *iff* the first marking of a run is $M'_1 + \{p'\}$.

Since the set of reachable markings of Σ_1 is included in that of Σ_2 *iff* ϕ is initial opaque w.r.t. obs, part (2) holds.

(3) Let us consider an instance of PCP with (a_i, b_i), for $i = 1, \ldots, n$. Then:

- Σ consists of a net $(\{s, s'\}, \{(a_1, 1), (b_1, 1), \ldots, (a_n, n), (b_n, n)\}, W)$ and the initial markings $S_0 = \{\{s\}, \{s'\}\}$, with the arcs given by

$$W(s, (a_i, i)) = W((a_i, i), s) = W(s', (b_i, i)) = W((b_i, i), s') = 1$$

  for $i = 1, \ldots, n$. Clearly, Π_Σ is finite.
- obs is orwellian and depends only on the sequence $\lambda = (x_1, i_1) \ldots (x_m, i_m)$ returning $x_1 \ldots x_m i_1 \ldots i_m$ (note that $x_1 \ldots x_m$ is the concatenation of labels which are words).
- ϕ is true *iff* the first marking of a run is $\{s\}$.

Since the instance of PCP has a solution *iff* ϕ is initial opaque w.r.t. obs, part (3) holds.

(4) Let us consider an instance of PCP with (a_i, b_i), for $i = 1, \ldots, n$. Then:

– Σ consists of a net $(\{s\}, \{1, \ldots, n\}, W)$ and the initial marking $S_0 = \{\{s\}\}$, where the arcs are given by $W(s, i) = W(i, s) = 1$ for $i = 1, \ldots, n$. Clearly, Π_Σ is finite.
– *obs* always returns ϵ.
– $\phi(\{s\}, i_1 \ldots i_m)$ is true *iff* $m \geq 1 \Rightarrow a_{i_1} \ldots a_{i_m} \neq b_{i_1} \ldots b_{i_m}$.

Since the instance of PCP has a solution *iff* ϕ is opaque w.r.t. *obs*, part (4) holds. ∎

An analysis of the proof of the last result identifies two sources for the complexity of the opacity problem. The first one is the complexity of the studied property, captured through the definition of ϕ. In particular, the latter may be used to encode undecidable problems and so in practice one should presumably restrict the interest to relatively straightforward versions of opacity, such as the initial opacity. The second source is the complexity of the observation function, and it is presumably reasonable to restrict the interest to some simple classes of observation functions, such as the static observation functions. This should not, however, be considered as a real drawback since the initial opacity combined with an n-orwellian observation function yields an opacity notion which is powerful enough to deal, for example, with bounded security protocols (section 2.3).

What now follows is a crucial result stating that initial opacity with an n-orwellian observation function is decidable provided that the LTS generated by a marked Petri net is finite[5]. In fact, this result could be generalised to any finite LTS.

Proposition 6. *In the case of a finite LTS, initial opacity w.r.t. an n-orwellian observation function is decidable.*

Proof. The result was shown in [4] using regular language inclusion for $n = 1$. Here, we will re-use this result after reducing the case of $n = 2$ to that of $n = 1$ (the proposed reduction can easily be extended to any $n > 2$).

Let $\Pi = (S, L, \Delta, S_0)$ be a finite LTS, for which a 2-orwellian observation function *obs*, and initial opacity predicate ϕ, are given. We define an LTS $\Pi' = (S', L', \Delta', S'_0)$ together with a static observation function obs' and initial opacity predicate ϕ' for the runs of Π', as follows.

– S' comprises all triples (α, s, β) such that $s \in S$ and one of the following holds:
  - α is the label of an arc incoming to s and β is the label of an arc outgoing from s.
  - α is the label of an arc incoming to s and $\beta = \epsilon$.
  - $s \in S_0$, $\alpha = \epsilon$ and β is the label of an arc outgoing from s.

  Moreover, the triples from the third case form S'_0.
– Δ' comprises all $\big((\alpha, s_1, \beta), l, (\beta, s_2, \gamma)\big)$ such that $(s_1, \beta, s_2) \in \Delta$ and one of the following holds (below we also give the value of $obs'(l)$):

[5] Note that the finiteness of LTS is decidable, and can be checked using the standard coverability tree construction [17].

  - $\alpha = \epsilon = \gamma$, $l = \beta$ and $obs'(l) = obs(\beta)$.
  - $\alpha = \epsilon \neq \gamma$, $l = \beta\gamma$ and $obs'(l) = obs(\beta\gamma)$.
  - $\alpha \neq \epsilon = \gamma$, $l = \alpha\beta$ and $obs'(l) = obs(\alpha\beta)$.
  - $\alpha \neq \epsilon \neq \gamma$, $l = \alpha\beta\gamma$ and $obs'(l) = obs(\alpha\beta\gamma)$.
- ϕ' is true for $(\alpha, s, \beta) \in S_0'$ *iff* ϕ was true for s.

We then observe that the opacity problem for ϕ w.r.t. *obs* is equivalent to the opacity problem for ϕ' w.r.t. *obs'*. Hence, since the new LTS is finite, the former is decidable. ∎

The last result is an extension of the main result given in [4] which stated the same property for $n = 1$ (as well as for two other kinds of opacity mentioned earlier on).

3.2 Approximation of Opacity

As initial opacity is, in general, undecidable when LTSs are allowed to be infinite, we propose in this section a technique which might allow to verify it, at least in some cases, using a technique close to *abstract interpretation* [5, 6]. It uses an abstraction of opacity called under/over-opacity.

Definition 9 *For $i = 1, 2, 3$, let Π_i be an LTS. Moreover, let obs_i be an observation function and ϕ_i a predicate for the runs of Π_i such that the following hold:*

$$(\forall\, \xi \in run(\Pi_1) \cap \phi_1)\ (\exists\, \xi' \in run(\Pi_2) \cap \phi_2)\ obs_1(\xi) = obs_2(\xi')$$
$$(\forall\, \xi \in run(\Pi_3) \setminus \phi_3)\ (\exists\, \xi' \in run(\Pi_1) \setminus \phi_1)\ obs_3(\xi) = obs_1(\xi')\ .$$

Then ϕ_1 is under/over-opaque (or simply uo-opaque) w.r.t. obs_1 if for every $\xi \in run(\Pi_2) \cap \phi_2$ there is $\xi' \in run(\Pi_3) \setminus \phi_3$ such that $obs_3(\xi) = obs_1(\xi')$.

Intuitively, Π_2 provides an over-approximation of the runs satisfying ϕ_1, while Π_3 provides an under-approximation of those runs that do not satisfy ϕ_1.

Proposition 7. *Uo-opacity w.r.t. obs_1 implies opacity w.r.t. obs_1.*

Proof. Follows directly from definitions. ∎

Given Π_1, obs_1 and ϕ_1, the idea then is to be able to construct an over-approximation and under-approximation to satisfy the last definition. A possible way of doing this in the case of marked Petri nets is described next.

Uo-opacity for Petri nets Suppose that $\Sigma = (N, S_0)$ is a marked Petri net, $\Pi_1 = \Pi_\Sigma$, obs_1 is a static observation function for Π_1 and $\phi_1 \subseteq S_0$ is an initial opacity predicate for Π_1.

Deriving over-approximation The over-approximation is obtained by generating the coverability graph Π_2 of Σ (see [8] for details), starting from the initial nodes in $S_0 \cap \phi_1$. The only modification of the original algorithm needed is that in our setup there may be several starting nodes $S_0 \cap \phi_1$ rather than just one. However, this is a small technical detail. The observation function obs_2 is static and defined in the same way as obs_1. The predicate ϕ_2 is true for all the initial nodes $S_0 \cap \phi_1$. Crucially, Π_2 is always a finite LTS.

Proposition 8. $(\forall \xi \in run(\Pi_1) \cap \phi_1)\,(\exists \xi' \in run(\Pi_2) \cap \phi_2)\ obs_1(\xi) = obs_2(\xi')$.

Proof. Recall that the nodes in Π_2 are ω-markings, where a place can be assigned the value ω to indicate an unbounded number of tokens; moreover, consuming/producing a token from/to ω leads to ω.

It suffices to prove $\{\lambda \in L^* \mid \exists\, s_0 \in S_0 : (s_0, \lambda) \in \phi\} \subseteq \mathcal{L}(\Pi_2)$. Suppose that $(s_0, \lambda) \in run(\Pi_1) \cap \phi_1$. We will show, by induction on the length of λ, that there is $(s_0', \lambda) \in run(\Pi_2)$ such that $s_0' = s_0$ and $s_0 \oplus \lambda$ is covered by $s_0' \oplus \lambda$ (i.e., for every place p the value assigned by $s_0 \oplus \lambda$ is not greater than that assigned by $s_0' \oplus \lambda$).

Since the base case trivially holds, assume that the property is true for $(s_0, \lambda) \in run(\Pi_1) \cap \phi_1$ and that $(s_0, \lambda\, t) \in run(\Pi_1)$. Then, by the induction hypothesis, there is $(s_0', \lambda) \in run(\Pi_2)$ such that $s_0' = s_0$ and $s_0 \oplus \lambda$ is covered by $s_0' \oplus \lambda$. Since t is enabled in Σ at marking $s_0 \oplus \lambda$, and $s_0' \oplus \lambda$ covers the latter, t labels an arc outgoing from $s_0' \oplus \lambda$, leading to some state s. Clearly, s covers $s_0 \oplus (\lambda\, t)$ since no ω entry in $s_0 \oplus \lambda$ can be replaced by a finite value in s. ∎

Deriving under-approximation A straightforward way of finding under-approximation is to impose a maximal finite capacity max for the places of Σ (for example, by using the complement place construction), and then deriving the LTS Π_3 assuming that the initial markings are those in $S_0 \setminus \phi_1$. The observation function obs_3 is static and defined in the same way as obs_1. The predicate ϕ_3 is false for all the initial nodes $S_0 \setminus \phi_1$.

Clearly, Π_3 is always a finite LTS. However, for some Petri nets with infinite reachability graph (as shown later on by the second example), this under-approximation may be too restrictive, even if one takes arbitrarily large bound max. Then, in addition to using instance specific techniques, one may attempt to derive more generous under-approximation, in the following way.

We assume that there are some (invisible) transitions in Σ mapped by obs_1 to ϵ transitions, and propagate the information that a place could become unbounded due to infinite sequence of invisible transitions. The construction resembles the coverability graph generation.

As in the case of the reachability graph, the states in Π_3 are ω-markings (see the proof of Proposition 8). Then Π_3 is built by starting from the initial states $S_0 \setminus \phi_1$, and performing a depth-first exploration. At each visited ω-marking M, we find (for example, using a nested call to a coverability graph generation restricted to the invisible transitions starting from M) whether there exists $M' >$

M reachable from M through invisible transitions only[6]; then we set $M(p) = \omega$, for every place p such that $M'(p) > M(p)$.

Note that the above algorithm may be combined with the capacity based approach and then it always produces a finite Π_3. In general, however, Π_3 is not guaranteed to be finite.

It should be pointed out that Π_3 generated in this way will not, in general, be a deterministic LTS, but this does not matter as the only thing we will be interested in is the language it generates.

Proposition 9. $(\forall \xi \in run(\Pi_3) \setminus \phi_3)\,(\exists \xi' \in run(\Pi_1) \setminus \phi_1)\;obs_3(\xi) = obs_1(\xi')$.

Proof. It suffices to show that $obs_3(\mathcal{L}(\Pi_3)) \subseteq obs_1(\mathcal{L}(\Pi_1))$.
Suppose that $s_0 t_1 s_1 \ldots t_n s_n$ is a path in Π_3 starting from an initial state. We will show, by induction on n that, for every $k \geq 1$, there is $(s_0, \lambda) \in run(\Pi_1)$ such that $obs_1(\lambda) = obs_3(t_1 \ldots t_n)$ and, for every place p, either $s_n(p) \leq (s_0 \oplus \lambda)(p)$ or $\omega = s_n(p) > (s_0 \oplus \lambda)(p) \geq k$.

Since the base case trivially holds, assume that the property holds for n, $s_0 t_1 s_1 \ldots t_n s_n t s$ is a path in Π_3 starting from an initial state, and $k \geq 1$. Moreover, let $t'_1 \ldots t'_q$ $(q \geq 0)$ be a sequence of invisible transitions which was 'responsible' for replacing the marking resulting from executing t in s_n by s.
Given now an arbitrary $m \geq 1$ we can choose sufficiently large $k' \geq k$ such that, after applying the induction hypothesis, there is $(s_0, \lambda) \in run(\Pi_1)$ satisfying:

- $obs_1(\lambda) = obs_3(t_1 \ldots t_n)$ and, for every place p, either $s_n(p) \leq (s_0 \oplus \lambda)(p)$ or $\omega = s_n(p) > (s_0 \oplus \lambda)(p) \geq k'$.
- t followed by m repetitions of the sequence $t'_1 \ldots t'_q$ is executable in Σ from the marking $s_0 \oplus \lambda$.

The latter, in particular, means that the overall effect of the sequence $t'_1 \ldots t'_q$ on the marking of any place is that the number of tokens never decreases (otherwise 'negative' place markings would be eventually generated which is impossible). This in turn means that, by executing $t'_1 \ldots t'_q$ sufficiently many times from the marking $s_0 \oplus (\lambda\, t)$, we may reach a marking s' such that, for every place p, either $s(p) \leq s'(p) \in \mathbb{N}$ or $\omega = s(p) > s'(p) \geq k$. ∎

Deciding uo-opacity Assuming that we have successfully generated over- and under-approximations Π_2 and Π_3, uo-opacity holds *iff*

$$obs_2(\mathcal{L}(\Pi_2)) \subseteq obs_3(\mathcal{L}(\Pi_3))$$

And the latter problem is decidable whenever Π_2 and Π_3 are *finite* LTSs as it then reduces to that of inclusion of two regular languages.

[6] This search does not have to be complete for the method to work, however, the more markings M' we find, the better the overall result is expected to be.

4 Examples

To illustrate our work, we give two examples. The first one is inspired by an anonymity requirement required in the chemical industry. The second describes a simple voting system.

4.1 A Scenario from Chemical Engineering

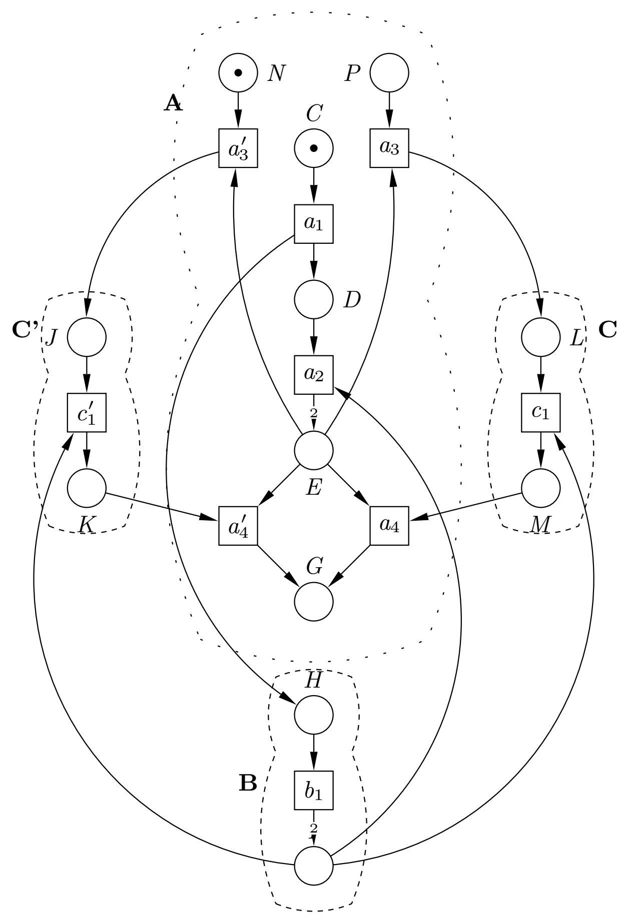


Fig. 1. Petri Net for the chemical industry scenario.

Figure 1 is a Petri net representation of a scenario in the chemical industry. It is adapted from an example presented in [15]. In the example, a chemical development company A asks company B (transition a_1) to prepare a feasibility study into the development of a new chemical. When this is completed (transition b_1) company A is informed of the conclusions (transition a_2). On the basis of these conclusions company A decides to commission a chemical safety report, from either company C (transition a_3) or company C' (transition a'_3). The relevant law allows the chosen company to question company B on aspects of the feasibility study. However, the chosen company is not allowed to reveal its identity to company B, in order to protect the integrity of B's answers. In our example, there are only two possible companies, C and C', so our intention is that from B's point of view, the visible interactions do not reveal the identity of the chosen company.

We may assume that the actions a_3, a'_3, a_4 and a'_4 are not visible to B, as these actions concern only companies A and B.

We choose the (static) observation function of B to be the identity function, except for

$$\begin{array}{ll} obs'(a_3) = obs'(a_4) = \epsilon & obs'(c_1) = \gamma \\ obs'(a'_3) = obs'(a'_4) = \epsilon & obs'(c'_1) = \gamma \end{array}$$

We now demonstrate the set of transitions $\{c_1, c'_1\}$ to be O-anonymous.

If $\lambda = l_i \ldots l_n$, the properties that we require to be opaque w.r.t. obs are:

$$\phi(s, \lambda) = (\exists\, i : l_i = c_1) \text{ and } \phi'(s, \lambda) = (\exists\, i : l_i = c'_1)$$

The two possible sequences of actions of this system are $a_1\, b_1\, a_2\, a_3\, c_1\, a_4$ and $a_1\, b_1\, a_2\, a'_3\, c'_1\, a'_4$, and so the two possible observations of the system are

$$\begin{array}{l} obs(a_1\, b_1\, a_2\, a_3\, c_1\, a_4) = a_1\, b_1\, a_2 \epsilon\gamma\epsilon \\ obs(a_1\, b_1\, a_2\, a'_3\, c'_1\, a'_4) = a_1\, b_1\, a_2 \epsilon\gamma\epsilon \end{array}$$

which are observationally equivalent. The properties ϕ and ϕ' are therefore opaque, and the set $\{c_1, c'_1\}$ is strongly anonymous w.r.t. obs.

4.2 A Simple Voting Scheme

In this example, we consider a vote session allowing only two votes: 1 and 2. We then describe a simple voting scheme in the form of a Petri net (see figure 2). The voting scheme contains two phases. The first one called *voting phase* (when there is a token in Voting) allows any new voter to enter the polling station (transition NV) and vote (transitions $V1$ and $V2$). Votes are stored in two places *Results*1 and *Results*2. A particular voter A is identified, and we formulate our properties with respect to A. After an indeterminate time, the election enters the *counting phase* (when there is a token in Counting, after executing transition C, and no token in Voting). Then the different votes are counted. Votes for 1 are seen via transition $C1$ and vote for 2 via $C2$. This net has one obvious limitation. At the end, there still can be some tokens left in places Results1 and Results2 so this scheme does not ensure that every vote is counted.

We want to verify that the vote cast by A is secret: the two possible initial markings are $\{\mathit{Voting}, 1\}$ and $\{\mathit{Voting}, 2\}$. We prove that it is impossible to detect

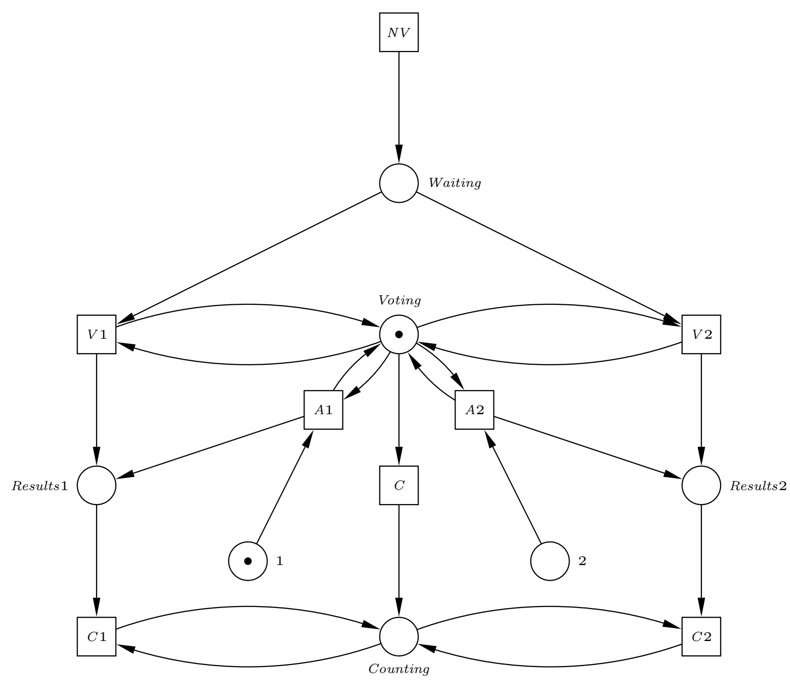


Fig. 2. Net for the voting system.

that “1” was marked (a symmetric argument would show that it is impossible to detect whether “2” was marked). The observation function is static and only transitions $C1$ and $C2$ are visible, i.e., $obs(C1) = C1$, $obs(C2) = C2$ and $obs(t) = \epsilon$ for any other transition t.

To verify opacity, we will use the under/over approximation method. The coverability graph (over-approximation) can be computed (see figure 3) using, for example, Tina [19]. After application of the observation function and simplification, we obtain that $obs_2(\mathcal{L}(\Pi_2)) = \{C_1, C_2\}^*$(see section 3.2 for the definition of Π_2.)

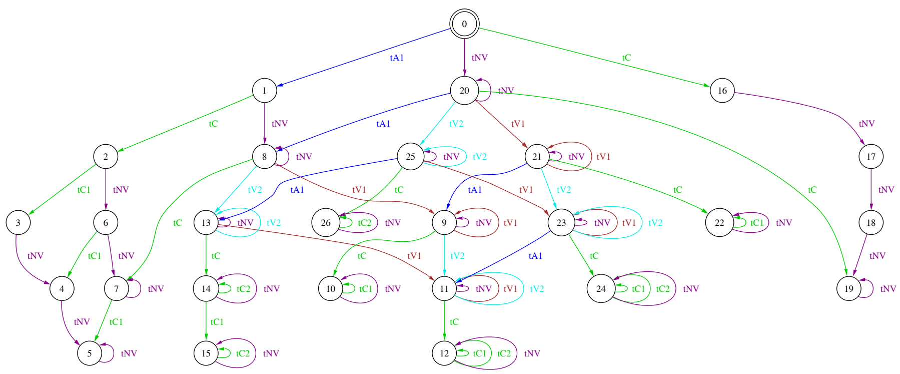
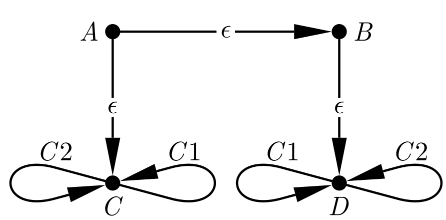


Fig. 3. Coverability graph for the voting system.

However, the simple under approximation using bounded capacity places will not work in this case, as for any chosen maximal capacity max, the language $\mathcal{L}(\Pi_3)$ will be finite whereas $obs_2(\mathcal{L}(\Pi_2))$ is infinite. Thus, we use the second under approximation technique. The following array represents the reachable states of the system starting from marking $\{Voting, 2\}$ using this technique.

	$Waiting$	$Voting$	$Results1$	$Results2$	1	2	$Counting$
A	$\mathbb{N}$	1	$\mathbb{N}$	$\mathbb{N}$	0	1	0
B	$\mathbb{N}$	1	$\mathbb{N}$	$\mathbb{N}$	0	0	0
C	$\mathbb{N}$	0	$\mathbb{N}$	$\mathbb{N}$	0	1	1
D	$\mathbb{N}$	0	$\mathbb{N}$	$\mathbb{N}$	0	0	1

The behaviour of this reachability graph, i.e. $obs_3(\mathcal{L}(\Pi_3))$, is simple:



Thus, the under-approximation is in this case: $obs_3(\mathcal{L}(\Pi_3)) = \{C1, C2\}^*$, and so $obs_2(\mathcal{L}(\Pi_2)) \subseteq obs_3(\mathcal{L}(\Pi_3))$ holds. We can now conclude that opacity of ϕ w.r.t. obs is verified and so the vote cast by A is kept secret.

5 Conclusions and Future Work

We have presented a general definition of opacity that extends previous work. This notion is no longer bound to the Petri net formalism and applies to any labelled transition system. However, restricting ourselves to initial opacity in the case of Petri nets allows us to find some decidability results. Furthermore, in this general model we can show how opacity relates to other information flow properties such as anonymity or non-inference.

However, non decidability results show that the opacity problem is a complex one. Its complexity is related to the complexity of the checked property, the complexity of the adversary's observational capabilities and the complexity of the system. The first point can be addressed by considering initial opacity which is still very expressive. The second one can be simplified by considering only n-orwellian observation functions. To solve the third problem, we can restrict ourselves to finite automata but this causes us to lose significant expressive power.

In the case of infinite Petri nets, over- and under- approximating gives a way of checking opacity. This technique works well in the case of the second example. We intend in future work to find a better abstraction for Petri nets and some well suited abstractions for other formalisms.

A current restriction of opacity is that it only tells you that an adversary cannot deduce *for sure* that a property is verified. The adversary cannot put probabilities on the likelihood of ϕ and $\neg\,\phi$. A potential further line of research is therefore to consider probabilistic opacity, by introducing probabilities on the initial parameters and considering probabilistic LTSs.

Acknowledgements

This research was supported by the EPSRC DIRC, GOLD and SCREEN projects, the RNTL project PROUVE-03V360, the ACI project ROSSIGNOL and DSTL.

References

1. Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. In *31th International Colloquium on Automata, Languages and Programming (ICALP'04)*, 2004.
2. Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, 2000.
3. Jeremy W. Bryans, Maciej Koutny, and Peter Y. A. Ryan. Modelling dynamic opacity using Petri nets with silent actions. In *Proceedings of Formal Aspects of Security and Trust.* Kluwer Academic Press, 2004. To appear.

4. Jeremy W. Bryans, Maciej Koutny, and Peter Y. A. Ryan. Modelling opacity using Petri nets. In *Proceedings of WISP*, 2004. To appear in Electronic Notes in Theoretical Computer Science.
5. Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th ACM Symposium on Principles of Programming Languages*, pages 238–252. ACM Press, 1977.
6. Patrick Cousot and Radhia Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, 1992.
7. Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
8. Alain Finkel. The minimal coverability graph for Petri nets. *Lecture Notes in Computer Science*, 674:210–243, 1993.
9. Riccardo Focardi and Roberto Gorrieri. A taxonomy of trace-based security properties for CCS. In *Proceedings of the Computer Security Foundations Workshop VII (CSFW '94)*, pages 126–137. IEEE, 1994.
10. Riccardo Focardi and Roberto Gorrieri. Classification of Security Properties: Information flow. In *Foundations of Security Analysis and Design*, number 2171 in LNCS, pages 331–396, 2000.
11. Florent Jacquemard, Michaël Rusinowitch, and Laurent Vigneron. Compiling and verifying security protocols. In *Logic Programming and Automated Reasoning*, pages 131–160, 2000.
12. Petr Jancar and Javier Esparza. Deciding finiteness of Petri nets up to bisimulation. In *Automata, Languages and Programming*, pages 478–489, 1996.
13. Laurent Mazaré. Using unification for opacity properties. In *Proceedings of the Workshop on Issues in the Theory of Security (WITS'04)*, 2004.
14. Colin O'Halloran. A Calculus of Information Flow. In *ESORICS*, pages 147–159, 1990.
15. Panos Periorellis, Chris Townson, and Phil English. Structural concepts for trust, contract and security management for a virtual chemical engineering organisation. In *PST 2004*, 2004. http://dev.hil.unb.ca/Texts/PST/.
16. James Lyle Peterson. *Petri Net Theory and The Modeling of Systems*. Prentice Hall, Inc., Englewood Cliffs, Massachusetts,, 1981.
17. Lectures on Petri nets. W.Reisig and G.Rozenberg (Eds.) LNCS 1491 & 1492, 1998.
18. Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *ESORICS*, pages 198–218, 1996.
19. Time Petri Net Analyzer. `http://www.laas.fr/tina/`, 2004.
20. Peter Y.A.Ryan. Mathematical models of computer security. In *Foundations of Security Analysis and Design*, number 2171 in LNCS, 2000.