# 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
[2] Laboratoire VERIMAG; 2, av. de Vignates, Gières, France

**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 an example modelling requirements upon a simple voting system.
**Keywords:** opacity, non-deducibility, anonymity, non-interference, Petri nets, observable behaviour, labelled transition systems.

## 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 protocols [7]. 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 classical secrecy 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. In Section 2, we show how opacity is related to previous work in security. In Section 3, we consider the question of opacity checking. 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 consider a voting scheme, and show how the approximation technique might be used. All the proofs are available in [6].

## 1 Basic Definitions

The set of finite sequences over a set $A$ will be denoted by $A^*$, and the empty sequence by $\epsilon$. The length of a finite sequence $\lambda$ 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 $s' = s''$[1].*
*A run of $\Pi$ is a pair $(s_0, \lambda)$, where $s_0 \in S_0$ and $\lambda = l_1 \dots l_n$ is a finite sequence of labels such that there are states $s_1, \dots, s_n$ satisfying $(s_{i-1}, l_i, s_i)$, for $i = 1, \dots, n$. 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 $\Pi$ 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 $\Theta$ be a set of elements called *observables*. We will now aim at modelling the different capabilities for observing the system modelled by $\Pi$. 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 $\lambda = l_1 \dots l_n$):*

- *static: there is a mapping $obs' : L \to \Theta \cup \{\epsilon\}$ such that for every run $(s, \lambda)$ of $\Pi$, $obs(s, \lambda) = obs'(l_1) \dots obs'(l_n)$.*
- *dynamic: there is a mapping $obs' : L \times L^* \to \Theta \cup \{\epsilon\}$ such that for every run $(s, \lambda)$ of $\Pi$, $obs(s, \lambda) = obs'(l_1, \epsilon) obs'(l_2, l_1) \dots obs'(l_n, l_1 \dots l_{n-1})$.*
- *orwellian: there is a mapping $obs' : L \times L^* \to \Theta \cup \{\epsilon\}$ such that for every run $(s, \lambda)$ of $\Pi$, $obs(s, \lambda) = obs'(l_1, \lambda) \dots obs'(l_n, \lambda)$.*
- *$m$-orwellian: there is a mapping $obs' : L \times L^* \to \Theta \cup \{\epsilon\}$ such that for every run $(s, \lambda)$ of $\Pi$, $obs(s, \lambda) = obs'(l_1, \kappa_1) \dots obs'(l_n, \kappa_n)$, 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 $obs(s, \lambda)$ which is possible as $obs(s, \lambda)$ does not depend on $s$.*

Note that allowing $obs'$ to return $\epsilon$ 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

[1] A nondeterministic LTS can be transformed into a deterministic one through a relabeling that assigns a unique label to each transition.

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 $\phi$ (a predicate over system states and traces) for some run having only access to the result of the observation function. We will identify $\phi$ 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 $\phi$ can be deduced by the observer (note that we are not interested in establishing whether the underlying run does not belong to $\phi$; 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 $\phi$ 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, $\phi$ is called: initial-opaque / final-opaque / total-opaque if respectively the following hold:*

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

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

## 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 $\lambda$ by replacing each occurrence of $a_i$ by $\alpha$. 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 $\lambda$ listed in the (unique) order in which they appeared for the first time in $\lambda$; then $obs(\lambda)$ is obtained from $\lambda$ by replacing each occurrence of $a_{i_j}$ by $\alpha_j$. 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 [22], a definition of strong anonymity is presented for the process algebra CSP. In our (LTS) context, this definition translates as follows.

**Definition 4** *$\Pi$ is strongly anonymous w.r.t. $A$ if $\mathcal{L}(\Pi) = \mathcal{L}(\Pi')$, where $\Pi'$ is obtained from $\Pi$ 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** *$\Pi$ is O-anonymous w.r.t. $A$ if, for every sequence $\mu \in A^*$, the predicate $\phi_\mu$ over the runs of $\Pi$ 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 $\mu$ (with appropriate length restrictions) of anonymised actions is a possible sequence within the LTS. In Definition 5 above, the opacity of the predicate $\phi_\mu$ ensures that the sequence $\mu$ is a possible history of anonymised actions, because it is the only sequence for which the predicate $\phi_\mu$ is false, and so $\phi_\mu$ can only be opaque if $\mu$ is a possible sequence.

**Theorem 1.** *$\Pi$ is O-anonymous w.r.t. $A$* iff *it is strongly anonymous w.r.t. $A$.*

**Weak anonymity** A natural extension of strong anonymity is *weak anonymity*[2]. 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** *$\Pi$ is weakly anonymous w.r.t. $A$ if $\pi(\mathcal{L}(\Pi)) \subseteq \mathcal{L}(\Pi)$, for every permutation $\pi$ over the set $A$.*

In our framework, we have that

**Definition 7** *$\Pi$ is weak-O-anonymous if, for every sequence $\mu \in A^*$, the predicate $\phi_\mu$ over the runs of $\Pi$ introduced in Definition 5 is opaque w.r.t. $obs_A^d$.*

**Theorem 2.** *$\Pi$ is weak-O-anonymous w.r.t. $A$* iff *it is weak-anonymous w.r.t. $A$.*

**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 $\lambda$ by deleting each occurrence of $a_i$ which is preceded (directly or indirectly) by an occurrence of $\delta$. 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 $\lambda$ by replacing each occurrence of $a_i$ by $\alpha$, provided that $\rho_i$ never occurs within $\lambda$.

### 2.2 Non-Interference

Opacity can be linked to a particular formulation of non-interference. A discussion of non-interference can be found in [10] and [21]. The basic idea is that labels

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

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 originally called non-inference in [18], and is called strong non-deterministic non-interference in [11].

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

In other words, for any run $(s, \lambda)$ of $\Pi$, there exists a run $(s', \lambda')$ such that $\lambda'$ is $\lambda$ 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.*

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.*

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 [15]. 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:

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

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 $\Box$ 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:

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

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 [14]). 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 $\phi$ 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 $\lambda$. (note that, in the case of a bounded protocol, an $m$-orwellian function will be sufficient). Then, opacity of $\phi$ w.r.t. *obs* is equivalent to the concept introduced in [15].

## 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.*

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 [20]. 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 [20], and give their operational semantics in terms of *transition sequences*.[3] 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 $\Delta$ is defined by $(M, t, M') \in \Delta$ if $M[t\rangle M'$. The language of $\Sigma$ is that of $\Pi_\Sigma$.

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.*

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 $\phi$. In particular, the latter may be

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

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[4]. In fact, this result could be generalised to any finite LTS; i.e., in the case of a finite LTS, initial opacity w.r.t. an $n$-orwellian observation function is decidable.

### 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 us to verify it, at least in some cases, using what we call under/over-opacity.

**Definition 9** *For $i = 1, 2, 3$, let $\Pi_i$ be an LTS. Moreover, let $obs_i$ be an observation function and $\phi_i$ a predicate for the runs of $\Pi_i$ 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 $\phi_1$ 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, $\Pi_2$ provides an over-approximation of the runs satisfying $\phi_1$, while $\Pi_3$ provides an under-approximation of those runs that do not satisfy $\phi_1$. One can then show that uo-opacity w.r.t. $obs_1$ *implies* opacity w.r.t. $obs_1$. Given $\Pi_1$, $obs_1$ and $\phi_1$, 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 $\Pi_1$ and $\phi_1 \subseteq S_0$ is an initial opacity predicate for $\Pi_1$.

*Deriving over-approximation* The over-approximation is obtained by generating the coverability graph $\Pi_2$ of $\Sigma$ (see [9] 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 $\phi_2$ is true for all the initial nodes $S_0 \cap \phi_1$. Crucially, $\Pi_2$ is always a finite LTS.

[4] Note that the finiteness of LTS is decidable, and can be checked using the standard coverability tree construction [20].

**Proposition 6.** $(\forall\, \xi \in run(\Pi_1) \cap \phi_1)\,(\exists\, \xi' \in run(\Pi_2) \cap \phi_2)\ obs_1(\xi) = obs_2(\xi')$.

*Deriving under-approximation* A straightforward way of finding under-approximation is to impose a maximal finite capacity $max$ for the places of $\Sigma$ (for example, by using the complement place construction), and then deriving the LTS $\Pi_3$ 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 $\phi_3$ is false for all the initial nodes $S_0 \setminus \phi_1$.

Clearly, $\Pi_3$ is always a finite LTS. However, for some Petri nets with an infinite reachability graph (as shown later on by our 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 $\Sigma$ mapped by $obs_1$ to $\epsilon$ 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 $\Pi_3$ are $\omega$-markings (see the proof of Proposition 6). Then $\Pi_3$ is built by starting from the initial states $S_0 \setminus \phi_1$, and performing a depth-first exploration. At each visited $\omega$-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[5]; 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 $\Pi_3$. In general, however, $\Pi_3$ is not guaranteed to be finite.

It should be pointed out that $\Pi_3$ 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 7.** $(\forall\, \xi \in run(\Pi_3) \setminus \phi_3)\,(\exists\, \xi' \in run(\Pi_1) \setminus \phi_1)\ obs_3(\xi) = obs_1(\xi')$.

*Deciding uo-opacity* Assuming that we have generated over- and under- approximations $\Pi_2$ and $\Pi_3$, uo-opacity holds *iff* $obs_2(\mathcal{L}(\Pi_2)) \subseteq obs_3(\mathcal{L}(\Pi_3))$. And the latter problem is decidable whenever $\Pi_2$ and $\Pi_3$ are *finite* LTSs as it then reduces to that of inclusion of two regular languages.

## 4 A Simple Voting Scheme

To illustrate our work, we give an example of a simple voting system. Another one, inspired by an anonymity requirement required in the chemical industry, is described in [6].

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 1). 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

[5] 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.

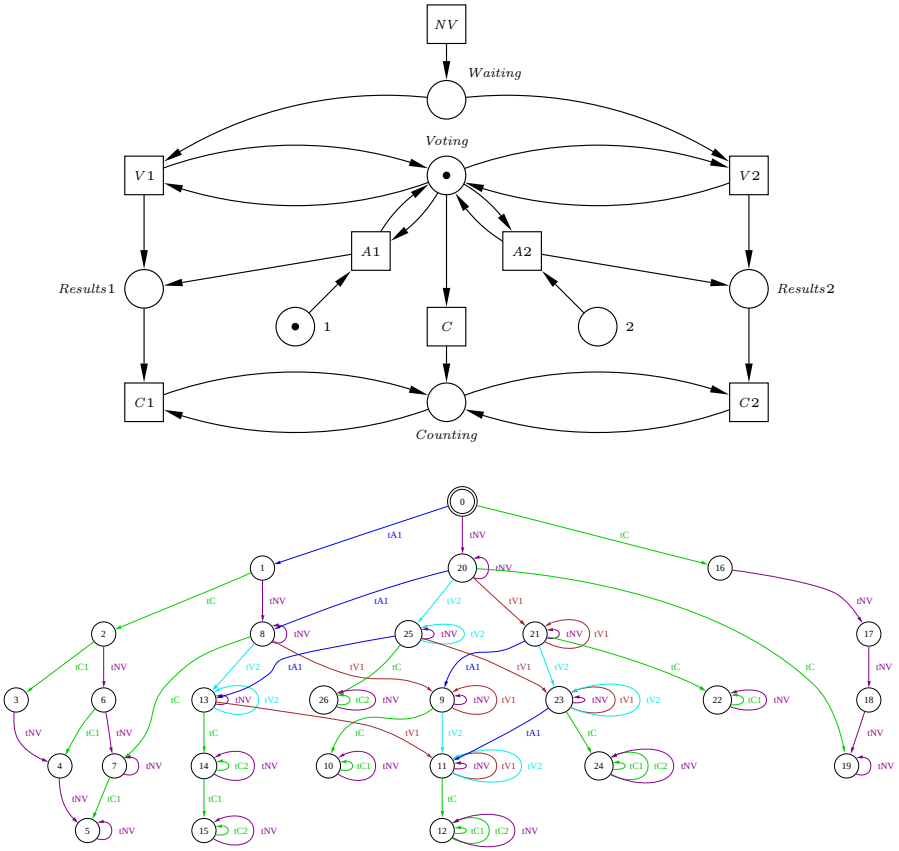



**Fig. 1.** Net for the voting system, and below its coverability graph.

(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.

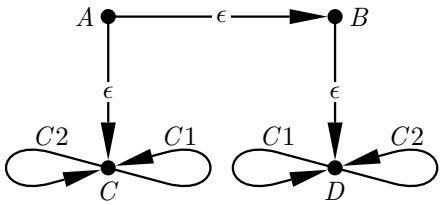
We want to verify that the vote cast by $A$ is secret: the two possible initial markings are $\{Voting, 1\}$ and $\{Voting, 2\}$. We prove that it is impossible to detect that "1" was marked (a symmetric argument would show that it is impossible to detect whether "2" was marked). The observation function 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 1) using, for example, Tina [24]. 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 $\Pi_2$.)

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* | *Results*1 | *Results*2 | 1 | 2 | *Counting* |
|---|---|---|---|---|---|---|---|
| $A$ | $\omega$ | 1 | $\omega$ | $\omega$ | 0 | 1 | 0 |
| $B$ | $\omega$ | 1 | $\omega$ | $\omega$ | 0 | 0 | 0 |
| $C$ | $\omega$ | 0 | $\omega$ | $\omega$ | 0 | 1 | 1 |
| $D$ | $\omega$ | 0 | $\omega$ | $\omega$ | 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 $\phi$ w.r.t. $obs$ is verified and so the vote cast by $A$ is kept secret.

## 5 Related Work

Concepts similar to opacity have been studied using *epistemic logics*, or logics of knowledge [8]. These logics include a "knowledge" operator, representing the case where an agent knows a fact, and are particularly suitable for reasoning about security within a multi-agent context [16, 12, 3]. The semantics can given within a "possible worlds" model: an agent knows a fact in a given world if it is true in every world that the agent considers possible. Opacity appears to be closely related to this knowledge operator, in that a property is opaque when the observer cannot be sure that it is true (see also below). That is, there is a world (a high level trace) that the observer considers possible, in which the fact does not hold. In [25] the notion of *ignorance* is developed, where an agent is ignorant of a fact $\phi$ when it cannot say for certain either that $\phi$ holds or that $\neg\,\phi$ holds. In our terms, an agent would be ignorant of $\phi$ if both $\phi$ and $\neg\,\phi$ were opaque.

There is a clear and strong relationship between our work and that contained in [13], and through it also with that in [8]. For example, final-opacity could be understood, using the terminology of [13], in the following way. To start with, we assume that an agent $i$ is modelled by our $obs$ function and, for every point $(r, m)$, we have $r_i(m) = obs(r, m)$. In other words, the $i$-th agent (in the sense of [13]) is observing the system. To model predicates within our approach, we then use information functions of [13], saying that $f$ is such a function and, for every point $(r, m)$, $f(r, m)$ returns a *true* or *false* value which only depends on

the $m$-th state of the run $r$. Then, applying Definition 3.3 of [13], results in the following rendering of our notion of final-opacity: "for every point $(r, m)$ there is another point $(r', m')$ such that $obs(r, m) = obs(r', m')$ and $f(r, m) = \neg\, f(r', m')$". Indeed, this looks very similar to the definition used by us, but is in fact *strictly stronger*, since our definition should correspond to the following: "for every point $(r, m)$ with $f(r, m) = true$, there is another point $(r', m')$ such that $obs(r, m) = obs(r', m')$ and $f(r', m') = false$". And the notion based on Definition 3.3 of [13] is basically equivalent to opacity of both $f$ and $f' = \neg\, f$. We therefore feel that there is no straightforward way of embedding our approach within that proposed in [13] (and so also [8]). We also feel that the basic reason behind this is that our notion of information hiding is 'asymmetric' in a sense that different values are obscured in possibly different ways. To make this more concrete, we could propose a slight modification of the definition from [13] along the following lines:

> Assume additionally that for every $v$ in the range of $f$ there exists possibly empty set $Mask(v)$ of values in the domain of $f$. Then, if $f$ is a $j$-information function, then agent $j$ mantains $f$-secrecy w.r.t. $i$ in system $\mathcal{R}$ if, for all points $(r, m)$ and values $v \in Mask(f(r, m))$ there is a point $(r', m')$ such that $r_i(m) = r'_i(m')$ and $f(r', m') = v$.

Intuitively, $Mask(v)$ provides sufficient obscurity from the point of view of agent $j$ about the actual value of $v$. In our case we could then set $Mask(true) = false$ and $Mask(false) = \varnothing$ (the latter to indicate that we do not care about the states where our predicate is false). And final-opacity would then be expressible using the modified definition. Our hypothesis is that such a modification constitutes an interesting true weakening of the security notion discussed in [13], and consequently it deserves an investigation in its own right.

## 6 Conclusions

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.

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 our voting example. We intend in future work to find a better abstraction for Petri nets and some well suited abstractions for other formalisms.

Some of the work done within epistemic logic has been with a view to model checking (see [17, 19, 23] for recent examples). Automatic verification is also an important goal of our work, and so exploring the connections between epistemic logic and opacity should prove a strong basis for further research.

## References


1. M.Abadi and V.Cortier: Deciding Knowledge in Security Protocols under Equational Theories. In: ICALP (2004)
2. M.Abadi and P.Rogaway: Reconciling two Views of Cryptography (The computational soundness of formal encryption). In: IFIP TCS2000 (2000)
3. P.Bieber: A Logic of Communication in a Hostile Environment. In: CSFW (1990)
4. J.W.Bryans, M.Koutny and P.Y.A.Ryan: Modelling Opacity using Petri Nets. ENTCS 121 (2005)
5. J.W.Bryans, M.Koutny and P.Y.A.Ryan: Modelling Dynamic Opacity using Petri Nets with Silent Actions. In: FAST (2004)
6. J.W.Bryans, M.Koutny, L.Mazaré and P.Y.A.Ryan: Opacity Generalised to Transition Systems. CS-TR 868, University of Newcastle (2004) `http://www.cs.ncl.ac.uk/research/pubs/trs/abstract.php?number=868`
7. D.Dolev and A.C.Yao: On the Security of Public Key Protocols. IEEE Transactions on Information Theory 29 (1983)
8. R.Fagin, J.Y.Halpern, Y.Moses and M.Y.Vardi: Reasoning about Knowledge. MIT press (1995)
9. A.Finkel: The Minimal Coverability Graph for Petri Nets. LNCS 674 (1993)
10. R.Focardi and R.Gorrieri: A Taxonomy of Trace-Based Security Properties for CCS. In: CSFW (1994)
11. R.Focardi and R.Gorrieri: Classification of Security Properties: Information flow. LNCS 2171 (2000)
12. J.Glasgow, G.Macewen and P.Panangaden: A Logic for Reasoning about Security. ACM Transactions on Computer Systems 10 (1992)
13. J.Y.Halpern and K.O'Neill: Anonymity and Information Hiding in Multiagent Systems. In: CSFW (2003)
14. F.Jacquemard, M.Rusinowitch and L.Vigneron: Compiling and Verifying Security Protocols. In: LPAR (2000)
15. L.Mazaré: Using Unification For Opacity Properties. In: WITS (2004)
16. L.Moser: A Logic of Knowledge and Belief for Reasoning about Security. In: CSFW (1989)
17. W.Nabialek, A.Niewiadomski, W.Penczek, A.Polórla and M.Szreter: Verics 2004: A Model Checker of Real-Time and Multi-agent Systems. In: CS&P (2004)
18. C.O'Halloran: A Calculus of Information Flow. In: ESORICS (1990)
19. F.Raimondi and A.Lomuscio: Verification of Multiagent Systems via Ordered Binary Decision Diagrams: an Algorithm and its Implementation. TR-04-01, King's College (2004)
20. W.Reisig and G.Rozenberg (Eds.): Lectures on Petri Nets. LNCS 1491 & 1492 (1998)
21. P.Y.A.Ryan: Mathematical Models of Computer Security. LNCS 2171 (2000)
22. S.Schneider and A.Sidiropoulos: CSP and Anonymity. In: ESORICS (1996)
23. S.van Otterloo, W.van der Hoek and M.Woolridge: Model Checking a Knowledge Exchange Scenario. In: IJCAI (2003)
24. Time Petri Net Analyzer. `http://www.laas.fr/tina/` (2004)
25. W.van der Hoek and A.Lomuscio: A Logic for Ignorance. ENTCS 85 (2004)