# Measuring the robustness
# Livrable du projet ImpRo
# (ANR-2010-BLAN-0317)

Béatrice Bérard[1], Patricia Bouyer[2], Kim G. Larsen[3], Nicolas Markey[2], John Mullins[4], Ocan Sankur[2], Mathieu Sassolas[5], and Claus Thrane[3]

[1]LIP6, Université Pierre & Marie Curie, CNRS, France
[2]LSV, ENS Cachan, CNRS, France
[3]Dept. Computer Science, Aalborg University, Denmark
[4]École Polytechnique de Montréal, Dept. of Comp. & Soft. Eng., Canada
[5]Département d'Informatique, Université Libre de Bruxelles, Belgique

This document present two works realized in the framework of the ImpRo project and in relation with the task *Quantifying robustness and implementability*. One of the goal of this task is to propose methods to qualify in which manner a system is robust to some properties.

The first work presented in this document studies the opacity of purely probabilistic systems. i.e. systems in which the non-deterministic choices have been replaced by probabilistic distribution over the set of states. The notion of opacity has been previously studied in the case of non-deterministic transition systems to determine whether a system respects some security properties. The idea is the following: an external passive attacker tries to gain information on a secret property through observation of the system executions and the system is considered opaque if secret and non secret executions cannot be distinguished by the attacker. Hence, by instantiating the secret predicate and the observation function, one is able to show several interesting security properties on the modeled systems like for instance anonymity or non-interference.

In this work, the authors propose two probabilistic definitions of opacity with the aim to (1) measure the security breach, instead of simply giving a yes/no answer and (2) measure the opacity robustness, when the system is opaque. Indeed, even when a system is opaque, the distribution of runs in some observation classes can be unbalanced and hence give information to an observer, thus weakening the system. For instance, if a *bip* is emitted after reception of an incoming message in 99% of cases, then an observer clearly gains information from hearing the *bip*.

These definitions are illustrated on several examples given by probabilistic

models. The authors also show how to compute these measures for restricted types of observation, in the special case where the secret predicate is regular.

The joint document is a preliminary version of an article to appear in Mathematical Structures in Computer Science.

The second work studies the concept of robustness and implementability, i.e. whether a system is robust to some perturbation and whether it can be "really" implemented, for systems with real-time behavior considering the model of timed automata. One interesting question, given a model for a real-time system, is to determine whether there exists a robust or implementable model for such a system which respects the same properties. In fact, imagine that given a timed automaton respecting some properties, one want to know whether it is possible to build a system respecting exactly the same properties (for instance having the same set of executions) which is robust or implementable. One idea could also be to establish a measure to determine "how much" the given timed automaton can be transformed into an equivalent robust timed automaton. This work shows that there is no need to introduce such a measure because it is in fact always possible to transform any timed automaton into a robust or implementable one respecting the exact same properties with respect to some given branching temporal logic.

The joint document corresponding to this work has been published under the title *Timed Automata can always be made implementable* at the conference CONCUR (International Conference on Concurrency Theory) in 2011.

*Under consideration for publication in Math. Struct. in Comp. Science*

# Quantifying Opacity[†]

Béatrice Bérard[1], John Mullins[2] and Mathieu Sassolas[1,3,‡]

[1] *Université Pierre & Marie Curie, LIP6/MoVe, CNRS UMR 7606, Paris, France*
[2] *École Polytechnique de Montréal, Dept. of Comp. & Soft. Eng., Montreal (Quebec), Canada*
[3] *Département d'informatique, Université Libre de Bruxelles, Bruxelles, Belgique*

*Received October 2012*

Opacity is a general language-theoretic framework in which several security properties of a system can be expressed. Its parameters are a predicate, given as a subset of runs of the system, and an observation function, from the set of runs into a set of observables. The predicate describes secret information in the system and, in the possibilistic setting, it is opaque if its membership cannot be inferred from observation.
In this paper, we propose several notions of quantitative opacity for probabilistic systems, where the predicate and the observation function are seen as random variables. Our aim is to measure (i) the probability of opacity leakage relative to these random variables and (ii) the level of uncertainty about membership of the predicate inferred from observation. We show how these measures extend possibilistic opacity, we give algorithms to compute them for regular secrets and observations, and we apply these computations on several classical examples. We finally partially investigate the non-deterministic setting.

## 1. Introduction

*Motivations.* Opacity (Mazaré, 2005) is a very general framework where a wide range of security properties can be specified, for a system interacting with a passive attacker. This includes for instance anonymity or non-interference (Goguen and Meseguer, 1982), the basic version of which states that high level actions cannot be detected by low level observations. Non-interference alone cannot capture every type of information flow properties. Indeed, it expresses the complete absence of information flow yet many information flow properties, like anonymity, permits some information flow while peculiar piece of information is required to be kept secret. The notion of opacity was introduced with the aim to provide a uniform description for security properties e.g. non-interference, noninference, various notions of anonymity, key compromise and refresh, downgrading, etc. (Bryans et al., 2008). Ensuring opacity by control was further studied in (Dubreil et al., 2010).

The general idea behind opacity is that a passive attacker should not have worthwhile

[†] Part of this work has been published in the proceedings of Qest'10 (Bérard et al., 2010).
[‡] Corresponding author: mathieu.sassolas@ulb.ac.be – ULB - Campus de la Plaine, Boulevard du Triomphe, CP212, 1050 Brussels, Belgium.

information, even though it can observe the system from the outside. The approach, as many existing information flow-theoretic approaches, is possibilistic. We mean by this that non determinism is used as a feature to model the random mechanism generation for all possible system behaviors. As such, opacity is not accurate enough to take into account two orthogonal aspects of security properties both regarding evaluation of the information gained by a passive attacker.

The first aspect concerns the quantification of security properties. If executions leaking information are negligible with respect to the rest of executions, the overall security might not be compromised. For example if an error may leak information, but appears only in 1% of cases, the program could still be considered safe. The definitions of opacity (Alur et al., 2006; Bryans et al., 2008) capture the existence of at least one perfect leak, but do not grasp such a measure.

The other aspect regards the category of security properties a system has to assume when interacting with an attacker able to make inferences from experiments on the base of statistical analysis. For example, if every time the system goes *bip*, there is 99% chances that action $a$ has been carried out by the server, then every *bip* can be guessed to have resulted from an $a$. Since more and more security protocols make use of randomization to reach some security objectives (Chaum, 1988; Reiter and Rubin, 1998), it becomes important to extend specification frameworks in order to cope with it.

*Contributions.* In this paper we investigate several ways of extending opacity to a purely probabilistic framework. Opacity can be defined either as the capacity for an external observer to deduce that a predicate was true (asymmetrical opacity) or whether a predicate is true or false (symmetrical opacity). Both notions can model relevant security properties, hence deserve to be extended. On the other hand, two directions can be taken towards the quantification of opacity. The first one, which we call *liberal*, evaluates the degree of non-opacity of a system: how big is the security hole? It aims at assessing the probability for the system to yield *perfect* information. The second direction, which is called *restrictive*, evaluates how opaque the system is: how robust is the security? The goal here is to measure how reliable is the information gained through observation. This yields up to four notions of quantitative opacity, displayed in Table 1, which are formally defined in this paper. The choice made when defining these measures was that a value 0 should be meaningful for opacity in the possibilistic sense. As a result, liberal measures are 0 when the system is opaque and restrictive ones are 0 when the system is not.

Moreover, like opacity itself, all these measures can be instantiated into several probabilistic security properties such as probabilistic non-interference and anonymity. We also show how to compute these values in some regular cases and apply the method to the

| | Asymmetric | Symmetric |
|---|---|---|
| Liberal (Security hole) | LPO ($\mathsf{PO}^A_\ell$) | LPSO ($\mathsf{PO}^S_\ell$) |
| Restrictive (Robustness) | RPO ($\mathsf{PO}^A_r$) | RPSO ($\mathsf{PO}^S_r$) |

Table 1. The four probabilistic opacity measures.

dining cryptographers problem and the crowd protocols, re-confirming in passing the correctness result of Reiter and Rubin (Reiter and Rubin, 1998).

Although the measures are defined in systems without nondeterminism, they can be extended to the case of systems scheduled by an adversary. We show that non-memoryless schedulers are requested in order to reach optimum opacity measures.

*Related Work.* Quantitative measures for security properties were first advocated in (Millen, 1987) and (Wittbold and Johnson, 1990). In (Millen, 1987), Millen makes an important step by relating the non-interference property with the notion of mutual information from information theory in the context of a system modeled by a deterministic state machine. He proves that the system satisfies the non-interference property if and only if the mutual information between the high-level input random variable and the output random variable is zero. He also proposes mutual information as a measure for information flow by showing how information flow can be seen as a noisy probabilistic channel, but he does not show how to compute this measure. In (Wittbold and Johnson, 1990) Wittbold and Johnson introduce *nondeducibility on strategies* in the context of a non-deterministic state machine. A system satisfies nondeducibility on strategies if the observer cannot deduce information from the observation by any collusion with a secret user and using any adaptive strategies. They observe that if such a system is run multiple times with feedback between runs, information can be leaked by coding schemes across multiple runs. In this case, they show that a discrete memoryless channel can be built by associating a distribution with the noise process. From then on, numerous studies were devoted to the computation of (covert) channel capacity in various cases (see e.g. (Mantel and Sudbrock, 2009)) or more generally information leakage.

In (Smith, 2009), several measures of information leakage extending these seminal works for deterministic or probabilistic programs with probabilistic input are discussed. These measures quantify the information concerning the input gained by a passive attacker observing the output. Exhibiting programs for which the value of entropy is not meaningful, Smith proposes to consider instead the notions of vulnerability and min-entropy to take in account the fact that some execution could leak a sufficiently large amount of information to allow the environment to guess the remaining secret. As discussed in Section 6, probabilistic opacity takes this in account.

In (Chatzikokolakis et al., 2008), in order to quantify anonymity, the authors propose to model the system (then called *Information Hiding System*) as a noisy channel in the sense of Information Theory: The secret information is modeled by the inputs, the observable information is modeled by the outputs and the two set are related by a conditional probability matrix. In this context, probabilistic information leakage is very naturally specified in terms of mutual information and capacity. A whole hierarchy of probabilistic notions of anonymity have been defined. The approach was completed in (Andrés et al., 2010) where anonymity is computed using regular expressions. More recently, in (Alvim et al., 2010), the authors consider Interactive Information Hiding Systems that can be viewed as channels with memory and feedback.

In (Boreale et al., 2011a), the authors analyze the asymptotic behaviour of attacker's error probability and information leakage in Information Hiding Systems in the context

of an attacker having the capabilities to make exactly one guess after observing $n$ independent executions of the system while the secret information remains invariant. Two cases are studied: the case in which each execution gives rise to a single observation and the case in which each state of an execution gives rise to an observation in the context of *Hidden Markov Models*. The relation of these sophisticated models of attacker with our attacker model is still to clarify. Similar models were also studied in (McIver et al., 2010), where the authors define an ordering w.r.t. probabilistic non-interference.

For systems modeled by process algebras, pioneering work was presented in (Lowe, 2004; Aldini et al., 2004), with channel capacity defined by counting behaviors in discrete time (non-probabilistic) CSP (Lowe, 2004), or various probabilistic extensions of noninterference (Aldini et al., 2004) in a generative-reactive process algebra. Subsequent studies in this area by (Aldini and Bernardo, 2009; Boreale, 2009; Boreale et al., 2010) also provide quantitative measures of information leak, relating these measures with noninterference and secrecy. In (Aldini and Bernardo, 2009), the authors introduce various notions of noninterference in a Markovian process calculus extended with prioritized/probabilistic zero duration actions and untimed actions. In (Boreale, 2009) the author introduces two notions of information leakage in the (non-probabilistic) $\pi$-calculus differing essentially in the assumptions made on the power of the attacker. The first one, called *absolute leakage*, corresponds to the average amount of information that was leaked to the attacker by the program in the context of an attacker with unlimited computational resources is defined in terms of conditional mutual information and follows the earlier results of Millen (Millen, 1987). The second notion, called *leakage rate*, corresponds to the maximal number of bits of information that could be obtained per experiment in the context in which the attacker can only perform a fixed number of tries, each yielding a binary outcome representing success or failure. Boreale also studies the relation between both notions of leakage and proves that they are consistent. The author also investigates compositionality of leakage. Boreale et al. (Boreale et al., 2010) propose a very general framework for reasoning about information leakage in a sequential process calculus over a semiring with some appealing applications to information leakage analysis when instantiating and interpreting the semiring. It appears to be a promising scheme for specifying and analysing regular quantitative information flow like we do in Section 5.

Although the literature on quantifying information leakage or channel capacity is dense, few works actually tried to extend general opacity to a probabilistic setting. A notion of probabilistic opacity is defined in (Lakhnech and Mazaré, 2005), but restricted to properties whose satisfaction depends only on the initial state of the run. The opacity there corresponds to the probability for an observer to guess from the observation whether the predicate holds for the run. In that sense our restrictive opacity (Section 4) is close to that notion. However, the definition of (Lakhnech and Mazaré, 2005) lacks clear ties with the possibilistic notion of opacity. Probabilistic opacity is somewhat related to the notion of *view* presented in (Boreale et al., 2011b) as authors include, like we do, a predicate to their probabilistic model and observation function but probabilistic opacity can hardly be compared with view. Indeed, on one hand, although their setting is different (they work on Information Hiding Systems extended with a view), our predicates over runs could be

viewed as a generalization of the predicates over a finite set of states (properties). On the other hand, a view in their setting is an arbitrary partition of the state space, whereas we partition the runs into only two equivalence classes (corresponding to *true* and *false*).

*Organization of the paper.* In Section 2, we recall the definitions of opacity and the probabilistic framework used throughout the paper. Section 3 and 4 present respectively the liberal and the restrictive version of probabilistic opacity, both for the asymmetrical and symmetrical case. We present in Section 5 how to compute these measures automatically if the predicate and observations are regular. Section 6 compares the different measures and what they allow to detect about the security of the system, through abstract examples and a case study of the Crowds protocol. In Section 7, we present the framework of probabilistic systems dealing with nondeterminism, and open problems that arise in this setting.

## 2. Preliminaries

In this section, we recall the notions of opacity, entropy, and probabilistic automata.

### 2.1. *Possibilistic opacity*

The original definition of opacity was given in (Bryans et al., 2008) for transition systems.

Recall that a transition system is a tuple $\mathcal{A} = \langle \Sigma, Q, \Delta, I \rangle$ where $\Sigma$ is a set of actions, $Q$ is a set of states, $\Delta \subseteq Q \times \Sigma \times Q$ is a set of transitions and $I \subseteq Q$ is a subset of initial states. A *run* in $\mathcal{A}$ is a finite sequence of transitions written as: $\rho = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots \xrightarrow{a_n} q_n$. For such a run, $\mathrm{fst}(\rho)$ (resp. $\mathrm{lst}(\rho)$) denotes $q_0$ (resp. $q_n$). We will also write $\rho \cdot \rho'$ for the run obtained by concatenating runs $\rho$ and $\rho'$ whenever $\mathrm{lst}(\rho) = \mathrm{fst}(\rho')$. The set of runs starting in state $q$ is denoted by $Run_q(\mathcal{A})$ and $Run(\mathcal{A})$ denotes the set of runs starting from some initial state: $Run(\mathcal{A}) = \bigcup_{q \in I} Run_q(\mathcal{A})$.

Opacity qualifies a predicate $\varphi$, given as a subset of $Run(\mathcal{A})$ (or equivalently as its characteristic function $\mathbf{1}_\varphi$), with respect to an *observation function* $\mathcal{O}$ from $Run(\mathcal{A})$ onto a (possibly infinite) set $Obs$ of *observables*. Two runs $\rho$ and $\rho'$ are equivalent w.r.t. $\mathcal{O}$ if they produce the same observable: $\mathcal{O}(\rho) = \mathcal{O}(\rho')$. The set $\mathcal{O}^{-1}(o)$ is called an *observation class*. We sometimes write $[\rho]_\mathcal{O}$ for $\mathcal{O}^{-1}(\mathcal{O}(\rho))$.

A predicate $\varphi$ is opaque on $\mathcal{A}$ for $\mathcal{O}$ if for every run $\rho$ satisfying $\varphi$, there is a run $\rho'$ not satisfying $\varphi$ equivalent to $\rho$.

**Definition 2.1 (Opacity).** Let $\mathcal{A}$ be a transition system and $\mathcal{O} : Run(\mathcal{A}) \to Obs$ a surjective function called observation. A predicate $\varphi \subseteq Run(\mathcal{A})$ is *opaque* on $\mathcal{A}$ for $\mathcal{O}$ if, for any $o \in Obs$, the following holds:

$$\mathcal{O}^{-1}(o) \not\subseteq \varphi.$$

However, detecting whether an event *did not* occur can give as much information as the detection that the same event *did* occur. In addition, as argued in (Alur et al., 2006), the asymmetry of this definition makes it impossible to use with refinement: opacity would

not be ensured in a system derived from a secure one in a refinement-driven engineering process. More precisely, if $\mathcal{A}'$ refines $\mathcal{A}$ and a property $\varphi$ is opaque on $\mathcal{A}$ (w.r.t $\mathcal{O}$), $\varphi$ is not guaranteed to be opaque on $\mathcal{A}'$ (w.r.t $\mathcal{O}$).

Hence we use the symmetric notion of opacity, where a predicate is symmetrically opaque if it is opaque as well as its negation. More precisely:

**Definition 2.2 (Symmetrical opacity).** A predicate $\varphi \subseteq Run(\mathcal{A})$ is *symmetrically opaque* on system $\mathcal{A}$ for observation function $\mathcal{O}$ if, for any $o \in Obs$, the following holds:

$$\mathcal{O}^{-1}(o) \not\subseteq \varphi \text{ and } \mathcal{O}^{-1}(o) \not\subseteq \overline{\varphi}.$$

The symmetrical opacity is a stronger security requirement. Security goals can be expressed as either symmetrical or asymmetrical opacity, depending on the property at stake.

For example non-interference and anonymity can be expressed by opacity properties. Non-interference states that an observer cannot know whether an action $h$ of high-level accreditation occurred only by looking at the actions with low-level of accreditation in the set $L$. So non-interference is equivalent to the opacity of predicate $\varphi_{NI}$, which is true when $h$ occurred in the run, with respect to the observation function $\mathcal{O}_L$ that projects the trace of a run onto the letters of $L$; see Section 3.2 for a full example. We refer to (Bryans et al., 2008) and (Lin, 2011) for other examples of properties using opacity.

When the predicate breaks the symmetry of a model, the asymmetric definition is usually more suited. Symmetrical opacity is however used when knowing $\varphi$ or $\overline{\varphi}$ is equivalent from a security point of view. For example, a noisy channel with binary input can be seen as a system $\mathcal{A}$ on which the input is the truth value of $\varphi$ and the output is the observation $o \in Obs$. If $\varphi$ is symmetrically opaque on $\mathcal{A}$ with respect to $\mathcal{O}$, then this channel is not perfect: there would always be a possibility of erroneous transmission. The ties between channels and probabilistic transition systems are studied in (Andrés et al., 2010) (see discussion in Section 4.2).

### 2.2. *Probabilities and information theory*

Recall that, for a countable set $\Omega$, a *discrete distribution* (or *distribution* for short) is a mapping $\mu : \Omega \to [0, 1]$ such that $\sum_{\omega\in\Omega} \mu(\omega) = 1$. For any subset $E$ of $\Omega$, $\mu(E) = \sum_{\omega\in E} \mu(\omega)$. The set of all discrete distributions on $\Omega$ is denoted by $\mathcal{D}(\Omega)$. A *discrete random variable* with values in a set $\Gamma$ is a mapping $Z : \Omega \to \Gamma$ where $[Z = z]$ denotes the event $\{\omega \in \Omega \mid Z(\omega) = z\}$.

The *entropy* of $Z$ is a measure of the uncertainty or dually, information about $Z$, defined by the expected value of $\log(\mu(Z))$:

$$H(Z) = -\sum_z \mu(Z = z) \cdot \log(\mu(Z = z))$$

where log is the base 2 logarithm.

For two random variables $Z$ and $Z'$ on $\Omega$, the *conditional entropy* of $Z$ *given the event*

$[Z' = z']$ such that $\mu(Z' = z') \neq 0$ is defined by:

$$H(Z|Z' = z') = -\sum_{z} (\mu(Z = z|Z' = z') \cdot \log(\mu(Z = z|Z' = z')))$$

where $\mu(Z = z|Z' = z') = \frac{\mu(Z=z, Z'=z')}{\mu(Z'=z')}$.

The *conditional entropy* of $Z$ *given the random variable* $Z'$ can be interpreted as the average entropy of $Z$ that remains after the observation of $Z'$. It is defined by:

$$H(Z|Z') = \sum_{z'} \mu(Z' = z') \cdot H(Z|Z' = z')$$

The *vulnerability* of a random variable $Z$, defined by $V(Z) = \max_z \mu(Z = z)$ gives the probability of the likeliest event of a random variable. Vulnerability evaluates the probability of a correct guess in one attempt and can also be used as a measure of information by defining *min-entropy* and *conditional min-entropy* (see discussions in (Smith, 2009; Andrés et al., 2010)).

### 2.3. *Probabilistic models*

In this work, systems are modeled using probabilistic automata behaving as finite automata where non-deterministic choices for the next action and state or termination are randomized: this is why they are called "*fully* probabilistic". We follow the model definition of (Segala, 1995), which advocates for the use of this special termination action (denoted here by $\surd$ instead of $\delta$ there). However, the difference lies in the model semantics: we consider only finite runs, which involves a modified definition for the (discrete) probability on the set of runs. Extensions to the non-deterministic setting are discussed in Section 7.

Recall that a finite automaton (FA) is a tuple $\mathcal{A} = \langle \Sigma, Q, \Delta, I, F \rangle$ where $\langle \Sigma, Q, \Delta, I \rangle$ is a finite transition system and $F \subseteq Q$ is a subset of final states. The automaton is deterministic if $I$ is a singleton and for all $q \in Q$ and $a \in \Sigma$, the set $\{q' \mid (q, a, q') \in \Delta\}$ is a singleton. Runs in $\mathcal{A}$, $Run_q(\mathcal{A})$ and $Run(\mathcal{A})$ are defined like in a transition system. A run of an FA is *accepting* if it ends in a state of $F$. The *trace* of a run $\rho = q_0 \xrightarrow{a_1} q_1 \cdots \xrightarrow{a_n} q_n$ is the word $\mathrm{tr}(\rho) = a_1 \cdots a_n \in \Sigma^*$. The *language* of $\mathcal{A}$, written $\mathcal{L}(\mathcal{A})$, is the set of traces of accepting runs starting in some initial state.

Replacing in a FA non-deterministic choices by choices based on a discrete distribution and considering only finite runs result in a *fully probabilistic finite automaton* (FPFA). Consistently with the standard notion of substochastic matrices, we also consider a more general class of automata, *substochastic automata* (SA), which allow us to describe subsets of behaviors from FPFAs, see Figure 1 for examples. In both models, no non-determinism remains, thus the system is to be considered as autonomous: its behaviors do not depend on an exterior probabilistic agent acting as a scheduler for non-deterministic choices.

**Definition 2.3 (Substochastic automaton).** Let $\surd$ be a new symbol representing a termination action. A *substochastic automaton* (SA) is a tuple $\langle \Sigma, Q, \Delta, q_0 \rangle$ where $\Sigma$

is a finite set of actions, $Q$ is a finite set of states, with $q_0 \in Q$ the initial state and $\Delta : Q \to ((\Sigma \times Q) \uplus \{\surd\} \to [0,1])$ is a mapping such that for any $q \in Q$,

$$\sum_{x \in (\Sigma \times Q) \uplus \{\surd\}} \Delta(q)(x) \leq 1$$

$\Delta$ defines substochastically the action and successor from the current state, or the termination action $\surd$.

In SA, we write $q \to \mu$ for $\Delta(q) = \mu$ and $q \xrightarrow{a} r$ whenever $q \to \mu$ and $\mu(a, r) > 0$. We also write $q \cdot \surd$ whenever $q \to \mu$ and $\mu(\surd) > 0$. In the latter case, $q$ is said to be a *final state*.

**Definition 2.4 (Fully probabilistic finite automaton).** A *fully probabilistic automaton* (FPFA) is a particular case of SA where for all $q \in Q$, $\Delta(q) = \mu$ is a distribution in $\mathcal{D}((\Sigma \times Q) \uplus \{\surd\})$ *i.e.*

$$\sum_{x \in (\Sigma \times Q) \uplus \{\surd\}} \Delta(q)(x) = 1$$

and for any state $q \in Q$ there exists a path (with non-zero probability) from $q$ to some final state.

Note that we only target finite runs and therefore we consider a restricted case, where any infinite path has probability 0.

Since FPFA is a subclass of SA, we overload the metavariable $\mathcal{A}$ for both SA and FPFA. The notation above allows to define a run for an SA like in a transition system as a finite sequence of transitions written $\rho = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots \xrightarrow{a_n} q_n$. The sets $Run_q(\mathcal{A})$ and $Run(\mathcal{A})$ are defined like in a transition system. A *complete run* is a (finite) sequence denoted by $\rho \cdot \surd$ where $\rho$ is a run and $\Delta(\mathrm{lst}(\rho))(\surd) > 0$. The set $CRun(\mathcal{A})$ denotes the set of complete runs starting from the initial state. In this work, we consider only such complete runs.

The *trace* of a run for an SA $\mathcal{A}$ is defined like in finite automata. The *language* of a substochastic automaton $\mathcal{A}$, written $\mathcal{L}(\mathcal{A})$, is the set of traces of complete runs starting in the initial state.

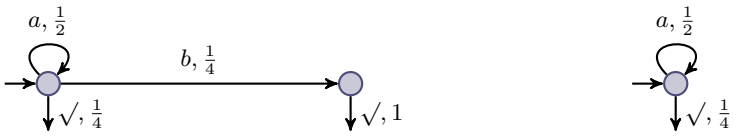
For an SA $\mathcal{A}$, a mapping $\mathbf{P}_\mathcal{A}$ into $[0,1]$ can be defined inductively on the set of complete runs by:

$$\mathbf{P}_\mathcal{A}(q\surd) = \mu(\surd) \qquad \text{and} \qquad \mathbf{P}_\mathcal{A}(q \xrightarrow{a} \rho) = \mu(a, r) \cdot \mathbf{P}_\mathcal{A}(\rho)$$

where $q \to \mu$ and $\mathrm{fst}(\rho) = r$.

The mapping $\mathbf{P}_\mathcal{A}$ is then a discrete distribution on $CRun(\mathcal{A})$. Indeed, the $\surd$ action can be seen as a transition label towards a new sink state $q_\surd$. Then, abstracting from the labels yields a finite Markov chain, where $q_\surd$ is the only absorbing state and the coefficients of the transition matrix are $M_{q,q'} = \sum_{a \in \Sigma} \Delta(q)(a, q')$. The probability for a complete run to have length $n$ is then the probability $p_n$ to reach $q_\surd$ in exactly $n$ steps. Therefore, the probability of all finite complete runs is $\mathbf{P}(CRun(\mathcal{A})) = \sum_n p_n$ and a classical result (Kemeny and Snell, 1976) on absorbing chains ensures that this probability is equal to 1.

Since the probability space is not generated by (prefix-closed) cones, this definition




(a) FPFA $\mathcal{A}_1$ (b) SA $\mathcal{A}_2$

Figure 1. $\mathcal{A}_2$ is the restriction of $\mathcal{A}_1$ to $a^*$.

does not yield the same probability measure as the one from (Segala, 1995). Since opacity properties are not necessarily prefix-closed, this definition is consistent with our approach.

When $\mathcal{A}$ is clear from the context, $\mathbf{P}_{\mathcal{A}}$ will simply be written $\mathbf{P}$.

Since $\mathbf{P}_{\mathcal{A}}$ is a (sub-)probability on $CRun(\mathcal{A})$, for any predicate $\varphi \subseteq CRun(\mathcal{A})$, we have $\mathbf{P}(\varphi) = \sum_{\rho\in\varphi} \mathbf{P}(\rho)$. The measure is extended to languages $K \subseteq \mathcal{L}(\mathcal{A})$ by $\mathbf{P}(K) = \mathbf{P}\left(\mathrm{tr}^{-1}(K)\right) = \sum_{\mathrm{tr}(\rho)\in K} \mathbf{P}(\rho)$.

In the examples of Figure 1, restricting the complete runs of $\mathcal{A}_1$ to those satisfying $\varphi = \{\rho \mid \mathrm{tr}(\rho) \in a^*\}$ yields the SA $\mathcal{A}_2$, and $\mathbf{P}_{\mathcal{A}_1}(\varphi) = \mathbf{P}_{\mathcal{A}_2}(CRun(\mathcal{A}_2)) = \frac{1}{2}$.

A non probabilistic version of any SA is obtained by forgetting any information about probabilities.

**Definition 2.5.** Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0\rangle$ be an SA. The (non-deterministic) finite automaton $unProb(\mathcal{A}) = \langle \Sigma, Q, \Delta', q_0, F\rangle$ is defined by:

— $\Delta' = \{(q, a, r) \in Q \times \Sigma \times Q \mid q \to \mu,\ \mu(a, r) > 0\}$,
— $F = \{q \in Q \mid q \to \mu, \mu(\surd) > 0\}$ is the set of final states.

It is easily seen that $\mathcal{L}(unProb(\mathcal{A})) = \mathcal{L}(\mathcal{A})$.

An observation function $\mathcal{O} : CRun(\mathcal{A}) \to Obs$ can also be easily translated from the probabilistic to the non probabilistic setting. For $\mathcal{A}' = unProb(\mathcal{A})$, we define $unProb(\mathcal{O})$ on $Run(\mathcal{A}')$ by $unProb(\mathcal{O})(q_0 \xrightarrow{a_1} q_1 \cdots q_n) = \mathcal{O}(q_0 \xrightarrow{a_1} q_1 \cdots q_n\surd)$.

## 3. Measuring non-opacity

### 3.1. *Definition and properties*

One of the aspects in which the definition of opacity could be extended to probabilistic automata is by relaxing the universal quantifiers of Definitions 2.1 and 2.2. Instead of wanting that *every* observation class should not be included in $\varphi$ (resp. $\varphi$ or $\overline{\varphi}$ for the symmetrical case), we can just require that *almost all* of them do. To obtain this, we give a measure for the set of runs leaking information. To express properties of probabilistic opacity in an FPA $\mathcal{A}$, the observation function $\mathcal{O}$ is considered as a random variable, as well as the characteristic function $\mathbf{1}_{\varphi}$ of $\varphi$. Both the asymmetrical and the symmetrical notions of opacity can be generalized in this manner.

**Definition 3.1 (Liberal probabilistic opacity).** The *liberal probabilistic opacity* or LPO of predicate $\varphi$ on FPA $\mathcal{A}$, with respect to (surjective) observation function $\mathcal{O}$ :

$CRun \to Obs$ is defined by:

$$\mathsf{PO}^A_\ell(\mathcal{A}, \varphi, \mathcal{O}) = \sum_{\substack{o \in Obs \\ \mathcal{O}^{-1}(o) \subseteq \varphi}} \mathbf{P}(\mathcal{O} = o).$$

The *liberal probabilistic symmetrical opacity* or LPSO is defined by:

$$\begin{aligned} \mathsf{PO}^S_\ell(\mathcal{A}, \varphi, \mathcal{O}) &= \mathsf{PO}^A_\ell(\mathcal{A}, \varphi, \mathcal{O}) + \mathsf{PO}^A_\ell(\mathcal{A}, \overline{\varphi}, \mathcal{O}) \\ &= \sum_{\substack{o \in Obs \\ \mathcal{O}^{-1}(o) \subseteq \varphi}} \mathbf{P}(\mathcal{O} = o) + \sum_{\substack{o \in Obs \\ \mathcal{O}^{-1}(o) \subseteq \overline{\varphi}}} \mathbf{P}(\mathcal{O} = o). \end{aligned}$$

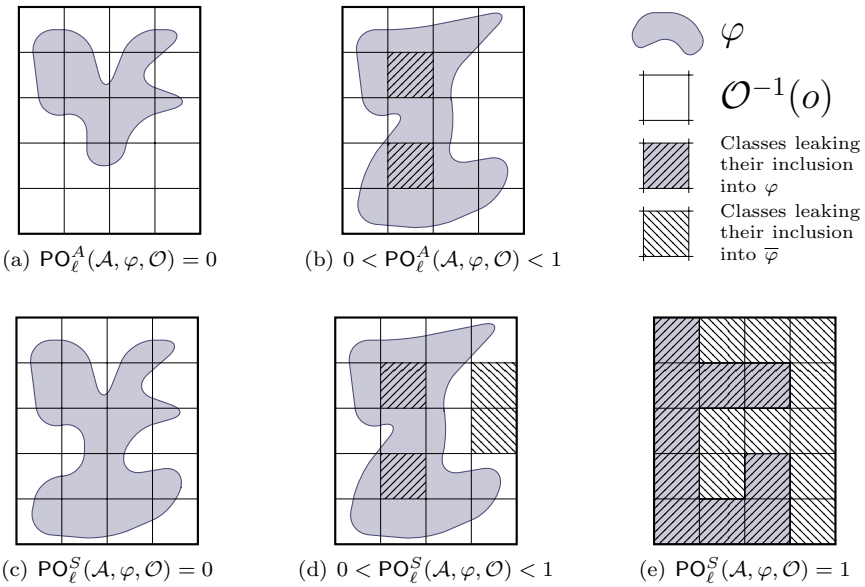
This definition provides a measure of how insecure the system is. The following propositions shows that a null value for these measures coincides with (symmetrical) opacity for the system, which is then secure.

For LPO, it corresponds to classes either overlapping both $\varphi$ and $\overline{\varphi}$ or included in $\overline{\varphi}$ as in Figure 2(a). LPO measures only the classes that leak their inclusion in $\varphi$. So classes included in $\overline{\varphi}$ are not taken into account. On the other extremal point, $\mathsf{PO}^A_\ell(\mathcal{A}, \varphi, \mathcal{O}) = 1$ when $\varphi$ is always true.

When LPSO is null, it means that each equivalence class $\mathcal{O}^{-1}(o)$ overlaps both $\varphi$ and $\overline{\varphi}$ as in Figure 2(c). On the other hand, the system is totally insecure when, observing through $\mathcal{O}$, we have all information about $\varphi$. In that case, the predicate $\varphi$ is a union of equivalence classes $\mathcal{O}^{-1}(o)$ as in Figure 2(e) and this can be interpreted in terms of conditional entropy relatively to $\mathcal{O}$. The intermediate case occurs when some, but not all, observation classes contain only runs satisfying $\varphi$ or only runs not satisfying $\varphi$, as in Figure 2(d).

**Proposition 3.2.**
(1) $0 \leq \mathsf{PO}^A_\ell(\mathcal{A}, \varphi, \mathcal{O}) \leq 1$ and $0 \leq \mathsf{PO}^S_\ell(\mathcal{A}, \varphi, \mathcal{O}) \leq 1$
(2) $\mathsf{PO}^A_\ell(\mathcal{A}, \varphi, \mathcal{O}) = 0$ if and only if $\varphi$ is opaque on $unProb(\mathcal{A})$ with respect to $unProb(\mathcal{O})$.
$\mathsf{PO}^S_\ell(\mathcal{A}, \varphi, \mathcal{O}) = 0$ if and only if $\varphi$ is symmetrically opaque on $unProb(\mathcal{A})$ with respect to $unProb(\mathcal{O})$.
(3) $\mathsf{PO}^A_\ell(\mathcal{A}, \varphi, \mathcal{O}) = 1$ if and only if $\varphi = CRun(\mathcal{A})$.
$\mathsf{PO}^S_\ell(\mathcal{A}, \varphi, \mathcal{O}) = 1$ if and only if $H(\mathbf{1}_\varphi | \mathcal{O}) = 0$.

*Proof of Proposition 3.2.*
(1) The considered events are mutually exclusive, hence the sum of their probabilities never exceeds 1.
(2) First observe that a complete run $r_0 a \dots r_n \surd$ has a non null probability in $\mathcal{A}$ iff $r_0 a \dots r_n$ is a run in $unProb(\mathcal{A})$. Suppose $\mathsf{PO}^A_\ell(\mathcal{A}, \varphi, \mathcal{O}) = 0$. Recall that $\mathcal{O}$ is assumed surjective. Then there is no observable $o$ such that $\mathcal{O}^{-1}(o) \subseteq \varphi$. Conversely, if $\varphi$ is opaque on $unProb(\mathcal{A})$, there is no observable $o \in Obs$ such that $\mathcal{O}^{-1}(o) \subseteq \varphi$, hence the null value for $\mathsf{PO}^A_\ell(\mathcal{A}, \varphi, \mathcal{O})$. The case of LPSO is similar, also taking into account the dual case of $\overline{\varphi}$ in the above.




(a) $\mathsf{PO}^A_\ell(\mathcal{A},\varphi,\mathcal{O}) = 0$ (b) $0 < \mathsf{PO}^A_\ell(\mathcal{A},\varphi,\mathcal{O}) < 1$

(c) $\mathsf{PO}^S_\ell(\mathcal{A},\varphi,\mathcal{O}) = 0$ (d) $0 < \mathsf{PO}^S_\ell(\mathcal{A},\varphi,\mathcal{O}) < 1$ (e) $\mathsf{PO}^S_\ell(\mathcal{A},\varphi,\mathcal{O}) = 1$

Figure 2. Liberal probabilistic asymmetrical and symmetrical opacity.

(3) For LPO, this is straightforward from the definition. For LPSO, $H(\mathbf{1}_\varphi|\mathcal{O}) = 0$ iff

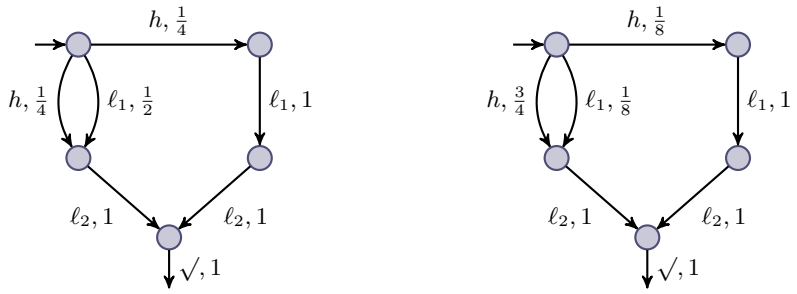
$$\sum_{\substack{o\in Obs\\ i\in\{0,1\}}} \mathbf{P}(\mathbf{1}_\varphi = i|\mathcal{O}=o)\cdot\log(\mathbf{P}(\mathbf{1}_\varphi = i|\mathcal{O}=o)) = 0$$

Since all the terms have the same sign, this sum is null if and only if each of its term is null. Setting for every $o \in Obs$, $f(o) = \mathbf{P}(\mathbf{1}_\varphi = 1|\mathcal{O}=o) = 1 - \mathbf{P}(\mathbf{1}_\varphi = 0|\mathcal{O}=o)$, we have: $H(\mathbf{1}_\varphi|\mathcal{O}) = 0$ iff $\forall\, o \in Obs$, $f(o)\cdot\log(f(o)) + (1-f(o))\cdot\log(1-f(o)) = 0$. Since the equation $x\cdot\log(x)+(1-x)\cdot\log(1-x) = 0$ only accepts 1 and 0 as solutions, it means that for every observable $o$, either all the runs $\rho$ such that $\mathcal{O}(\rho) = o$ are in $\varphi$, or they are all not in $\varphi$. Therefore $H(\mathbf{1}_\varphi|\mathcal{O}) = 0$ iff for every observable $o$, $\mathcal{O}^{-1}(o) \subseteq \varphi$ or $\mathcal{O}^{-1}(o) \subseteq \overline{\varphi}$, which is equivalent to $\mathsf{PO}^S_\ell(\mathcal{A},\varphi,\mathcal{O}) = 1$. □

### 3.2. *Example: Non-interference*

For the systems $\mathcal{A}_3$ and $\mathcal{A}_4$ of Figure 3, we use the predicate $\varphi_{NI}$ which is true if the trace of a run contains the letter $h$. In both cases the observation function $\mathcal{O}_L$ returns the projection of the trace onto the alphabet $\{\ell_1, \ell_2\}$. Remark that this example is an interference property (Goguen and Meseguer, 1982) seen as opacity. Considered unprobabilistically, both systems are interferent since an $\ell_2$ not preceded by an $\ell_1$ betrays the presence of an $h$. However, they differ by how often this case happens.

The runs of $\mathcal{A}_3$ and $\mathcal{A}_4$ and their properties are displayed in Table 2. Then we can see that $[\rho_1]_{\mathcal{O}_L} = [\rho_2]_{\mathcal{O}_L}$ overlaps both $\varphi_{NI}$ and $\overline{\varphi_{NI}}$, while $[\rho_3]_{\mathcal{O}_L}$ is contained totally in $\varphi$.




(a) FPA $\mathcal{A}_3$ (b) FPA $\mathcal{A}_4$

Figure 3. Interferent FPAs $\mathcal{A}_3$ and $\mathcal{A}_4$.

| $\mathrm{tr}(\rho)$ | $\mathbf{P}_{\mathcal{A}_3}(\rho)$ | $\mathbf{P}_{\mathcal{A}_4}(\rho)$ | $\in \varphi_{NI}$? | $\mathcal{O}_L(\rho)$ |
|---|---|---|---|---|
| $\mathrm{tr}(\rho_1) = \ell_1\ell_2\surd$ | 1/2 | 1/8 | 0 | $\ell_1\ell_2$ |
| $\mathrm{tr}(\rho_2) = h\ell_1\ell_2\surd$ | 1/4 | 1/8 | 1 | $\ell_1\ell_2$ |
| $\mathrm{tr}(\rho_3) = h\ell_2\surd$ | 1/4 | 3/4 | 1 | $\ell_2$ |

Table 2. Runs of $\mathcal{A}_3$ and $\mathcal{A}_4$.

Hence the LPO can be computed for both systems:

$$\mathsf{PO}^A_\ell(\mathcal{A}_3, \varphi_{NI}, \mathcal{O}_L) = \frac{1}{4} \qquad \mathsf{PO}^A_\ell(\mathcal{A}_4, \varphi_{NI}, \mathcal{O}_L) = \frac{3}{4}$$

Therefore $\mathcal{A}_3$ is more secure than $\mathcal{A}_4$. Indeed, the run that is interferent occurs more often in $\mathcal{A}_4$, leaking information more often.

Note that in this example, LPO and LPSO coincide. This is not always the case. Indeed, in the unprobabilistic setting, both symmetrical and asymmetrical opacity of $\varphi_{NI}$ with respect to $\mathcal{O}_L$ express the intuitive notion that "an external observer does not know whether an action happened or not". The asymmetrical notion corresponds to the definition of *strong nondeterministic non-interference* in (Goguen and Meseguer, 1982) while the symmetrical one was defined as *perfect security property* in (Alur et al., 2006).

## 4. Measuring the robustness of opacity

The completely opposite direction that can be taken to define a probabilistic version is a more paranoid one: how much information is leaked through the system's uncertainty? For example, on Figure 2(c), even though each observation class contains a run in $\varphi$ and one in $\overline{\varphi}$, some classes are *nearly* in $\varphi$. In some other classes the balance between the runs satisfying $\varphi$ and the ones not satisfying $\varphi$ is more even. Hence for each observation class, we will not ask if it is included in $\varphi$, but how likely $\varphi$ is to be true inside this class with a probabilistic measure taking into account the likelihood of classes. This amounts to measuring, inside each observation class, $\overline{\varphi}$ in the case of asymmetrical opacity, and the balance between $\varphi$ and $\overline{\varphi}$ in the case of symmetrical opacity. Note that these new

measures are relevant only for opaque systems, where the previous liberal measures are equal to zero.

In (Bérard et al., 2010), another measure was proposed, based on the notion of mutual information (from information theory, along similar lines as in (Smith, 2009)). However, this measure had a weaker link with possibilistic opacity (see discussion in Section 6). What we call here RPO is a new measure, whose relation with possibilistic opacity is expressed by the second item in Proposition 4.2.

### 4.1. *Restricting Asymmetrical Opacity*

In this section we extend the notion of asymmetrical opacity in order to measure how secure the system is.

*Definition and properties.* In this case, an observation class is more secure if $\varphi$ is less likely to be true. That means that it is easy (as in "more likely") to find a run not in $\varphi$ in the same observation class. Dually, a high probability for $\varphi$ inside a class means that few (again probabilistically speaking) runs will be in the same class yet not in $\varphi$.

Restrictive probabilistic opacity is defined to measure this effect globally on all observation classes. It is tailored to fit the definition of opacity in the classical sense: indeed, if one class totally leaks its presence in $\varphi$, RPO will detect it (second point in Proposition 4.2).

**Definition 4.1.** Let $\varphi$ be a predicate on the complete runs of an FPA $\mathcal{A}$ and $\mathcal{O}$ an observation function. The *restrictive probabilistic opacity* (RPO) of $\varphi$ on $\mathcal{A}$, with respect to $\mathcal{O}$, is defined by

$$\frac{1}{\mathsf{PO}_r^A(\mathcal{A},\varphi,\mathcal{O})} = \sum_{o\in Obs} \mathbf{P}(\mathcal{O}=o)\cdot\frac{1}{\mathbf{P}(\mathbf{1}_\varphi = 0 \mid \mathcal{O}=o)}$$

RPO is the harmonic means (weighted by the probabilities of observations) of the probability that $\varphi$ is false in a given observation class. The harmonic means averages the leakage of information inside each class. Since security and robustness are often evaluated on the weakest link, more weight is given to observation classes with the higher leakage, *i.e.* those with probability of $\varphi$ being false closest to 0.

The following proposition gives properties of RPO.

**Proposition 4.2.**
(1) $0 \leq \mathsf{PO}_r^A(\mathcal{A},\varphi,\mathcal{O}) \leq 1$
(2) $\mathsf{PO}_r^A(\mathcal{A},\varphi,\mathcal{O}) = 0$ if and only if $\varphi$ is not opaque on $unProb(\mathcal{A})$ with respect to $unProb(\mathcal{O})$.
(3) $\mathsf{PO}_r^A(\mathcal{A},\varphi,\mathcal{O}) = 1$ if and only if $\varphi = \emptyset$.

*Proof.* The first point immediately results from the fact that RPO is a means of values between 0 and 1.

From the definition above, RPO is null if and only if there is one class that is contained

in $\varphi$. Indeed, this corresponds to the case where the value of $\frac{1}{\mathbf{P}(\mathbf{1}_\varphi=0|\mathcal{O}=o)}$ goes to $+\infty$, for some $o$, as well as the sum.

Thirdly, if $\varphi$ is always false, then RPO is 1 since it is a means of probabilities all of value 1. Conversely, if RPO is 1, because it is defined as an average of values between 0 and 1, then all these values must be equal to 1, hence for each $o$, $\mathbf{P}(\mathbf{1}_\varphi = 0 \mid \mathcal{O} = o) = 1$ which means that $\mathbf{P}(\mathbf{1}_\varphi = 0) = 1$ and $\varphi$ is false. □

*Example: Debit Card System.* Consider a Debit Card system in a store. When a card is inserted, an amount of money $x$ to be debited is entered, and the client enters his PIN number (all this being gathered as the action Buy($x$)). The amount of the transaction is given probabilistically as an abstraction of the statistics of such transactions. Provided the PIN is correct, the system can either directly allow the transaction, or interrogate the client's bank for solvency. In order to balance the cost associated with this verification (bandwidth, server computation, etc.) with the loss induced if an insolvent client was debited, the decision to interrogate the bank's servers is taken probabilistically according to the amount of the transaction. When interrogated, the bank can reject the transaction with a certain probability[†] or accept it. This system is represented by the FPA $\mathcal{A}_{\text{card}}$ of Figure 4.

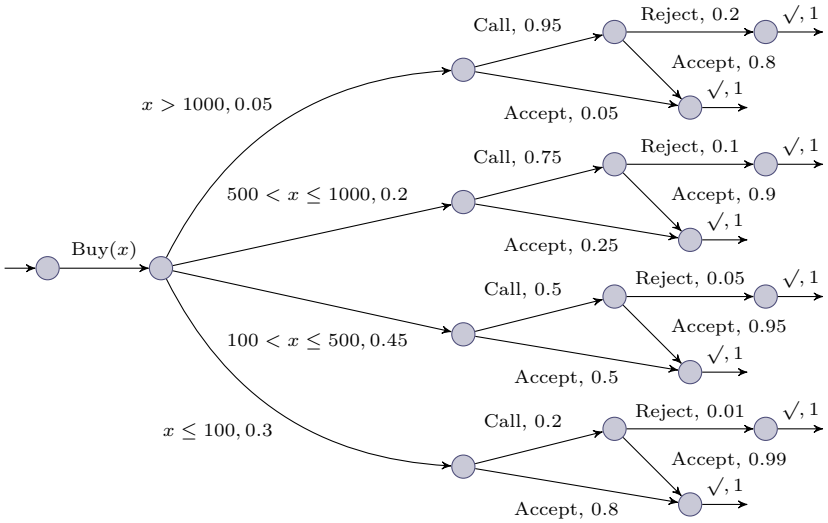



Figure 4. The Debit Card system $\mathcal{A}_{\text{card}}$.

Now assume an external observer can only observe if there has been a call or not to the bank server. In practice, this can be achieved, for example, by measuring the time taken for the transaction to be accepted (it takes longer when the bank is called), or by spying on the telephone line linking the store to the bank's servers (detecting activity on the network or idleness). Suppose what the external observer wants to know is whether

[†] Although the bank process to allow or forbid the transaction is deterministic, the statistics of the result can be abstracted into probabilities.

the transaction was worth more than 500€. By using RPO, one can assess how this knowledge can be derived from observation.

Formally, in this case the observables are $\{\varepsilon, \text{Call}\}$, the observation function $\mathcal{O}_{\text{Call}}$ being the projection on $\{\text{Call}\}$. The predicate to be hidden to the user is represented by the regular expression $\varphi_{>500} = \Sigma^*(\text{“}x > 1000\text{”} + \text{“}500 < x \leq 1000\text{”})\Sigma^*$ (where $\Sigma$ is the whole alphabet). By definition of RPO:

$$\begin{aligned}\frac{1}{\mathsf{PO}_r^A(\mathcal{A}_{\text{card}}, \varphi_{>500}, \mathcal{O}_{\text{Call}})} &= \mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon) \cdot \frac{1}{\mathbf{P}(\neg\varphi_{>500} | \mathcal{O}_{\text{Call}} = \varepsilon)} \\ &\quad + \mathbf{P}(\mathcal{O}_{\text{Call}} = \text{Call}) \cdot \frac{1}{\mathbf{P}(\neg\varphi_{>500} | \mathcal{O}_{\text{Call}} = \text{Call})}\end{aligned}$$

Computing successively $\mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon)$, $\mathbf{P}(\neg\varphi_{>500} | \mathcal{O}_{\text{Call}} = \varepsilon)$, $\mathbf{P}(\mathcal{O}_{\text{Call}} = \text{Call})$, and $\mathbf{P}(\neg\varphi_{>500} | \mathcal{O}_{\text{Call}} = \text{Call})$ (see Appendix A), we obtain:

$$\mathsf{PO}_r^A(\mathcal{A}_{\text{card}}, \varphi_{>500}, \mathcal{O}_{\text{Call}}) = \frac{28272}{39377} \simeq 0.718.$$

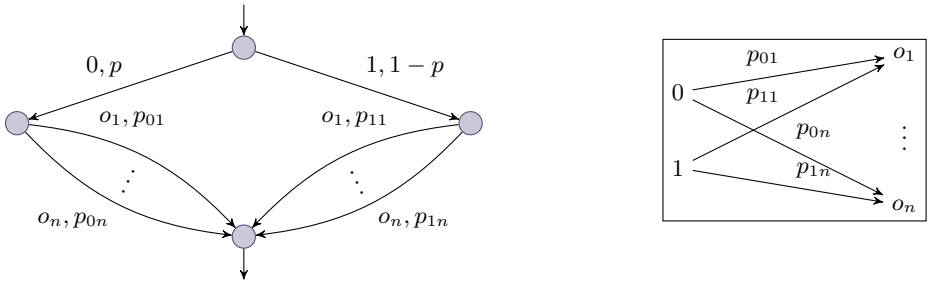
The notion of asymmetrical opacity, however, fails to capture security in terms of opacity for both $\varphi$ and $\overline{\varphi}$. And so does the RPO measure. Therefore we define in the next section a quantitative version of symmetrical opacity.

### 4.2. *Restricting symmetrical opacity*

Symmetrical opacity offers a sound framework to analyze the secret of a binary value. For example, consider a binary channel with $n$ outputs. It can be modeled by a tree-like system branching on 0 and 1 at the first level, then branching on observables $\{o_1, \ldots, o_n\}$, as in Figure 5. If the system wishes to prevent communication, the secret of predicate “the input of the channel was 1” is as important as the secret of its negation; in this case “the input of the channel was 0”. Such case is an example of *initial opacity* (Bryans et al., 2008), since the secret appears only at the start of each run. Note that any system with initial opacity and a finite set of observables can be transformed into a channel (Andrés et al., 2010), with input distribution $(p, 1-p)$, which is the distribution of the secret predicate over $\{\varphi, \overline{\varphi}\}$.

*Definition and properties.* Symmetrical opacity ensures that for each observation class $o$ (reached by a run), the probability of both $\mathbf{P}(\varphi \mid o)$ and $\mathbf{P}(\overline{\varphi} \mid o)$ is strictly above 0. That means that the lower of these probabilities should be above 0. In turn, the lowest of these probability is exactly the complement of the vulnerability (since $\mathbf{1}_\varphi$ can take only two values). That is, the security is measured with the probability of error in one guess (inside a given observation class). Hence, a system will be secure if, in each observation class, $\varphi$ is *balanced* with $\overline{\varphi}$.

**Definition 4.3 (Restrictive probabilistic symmetric opacity).** Let $\varphi$ be a predicate on the complete runs of an FPA $\mathcal{A}$ and $\mathcal{O}$ an observation function. The *restrictive*




(a) A system with initial secret

(b) Channel with binary input

Figure 5. A system and its associated channel.

*probabilistic symmetric opacity* (RPSO) of $\varphi$ on $\mathcal{A}$, with respect to $\mathcal{O}$, is defined by

$$\mathsf{PO}_r^S(\mathcal{A},\varphi,\mathcal{O}) = \frac{-1}{\sum_{o\in Obs}\mathbf{P}(\mathcal{O}=o)\cdot\log\left(1-V(\mathbf{1}_\varphi \mid \mathcal{O}=o)\right)}$$

where $V(\mathbf{1}_\varphi \mid \mathcal{O}=o) = \max_{i\in\{0,1\}}\mathbf{P}(\mathbf{1}_\varphi = i \mid \mathcal{O}=o)$.

Remark that the definition of RPSO has very few ties with the definition of RPO. Indeed, it is linked more with the notion of possibilistic symmetrical opacity than with the notion of quantitative asymmetrical opacity, and thus RPSO is not to be seen as an extension of RPO.

In the definition of RPSO, taking $-\log(1-V(\mathbf{1}_\varphi \mid \mathcal{O}=o))$ allows to give more weight to very imbalanced classes, up to infinity for classes completely included either in $\varphi$ or in $\overline{\varphi}$. Along the lines of (Smith, 2009), the logarithm is used in order to produce a measure in terms of bits instead of probabilities. These measures are then averaged with respect to the probability of each observation class. The final inversion ensures that the value is between 0 and 1, and can be seen as a normalization operation. The above motivations for the definition of RPSO directly yield the following properties:

**Proposition 4.4.**
(1) $0 \leq \mathsf{PO}_r^S(\mathcal{A},\varphi,\mathcal{O}) \leq 1$
(2) $\mathsf{PO}_r^S(\mathcal{A},\varphi,\mathcal{O}) = 0$ if and only if $\varphi$ is *not* symmetrically opaque on $unProb(\mathcal{A})$ with respect to $unProb(\mathcal{O})$.
(3) $\mathsf{PO}_r^S(\mathcal{A},\varphi,\mathcal{O}) = 1$ if and only if $\forall o \in Obs,\ \mathbf{P}(\mathbf{1}_\varphi = 1 \mid \mathcal{O}=o) = \frac{1}{2}$.

*Proof of Proposition 4.4.*
(1) Since the vulnerability of a random variable that takes only two values is between $\frac{1}{2}$ and 1, we have $1-V(\mathbf{1}_\varphi \mid \mathcal{O}=o) \in [0,\frac{1}{2}]$ for all $o \in Obs$. So $-\log(1-V(\mathbf{1}_\varphi \mid \mathcal{O}=o)) \in [1,+\infty[$ for any $o$. The (arithmetic) means of these values is thus contained within the same bounds. The inversion therefore yields a value between 0 and 1.
(2) If $\varphi$ is not symmetrically opaque, then for some observation class $o$, $\mathcal{O}^{-1}(o) \subseteq \varphi$ or $\mathcal{O}^{-1}(o) \subseteq \overline{\varphi}$. In both cases, $V(\mathbf{1}_\varphi \mid \mathcal{O}) = 1$, so $-\log(1-V(\mathbf{1}_\varphi \mid \mathcal{O}=o)) = +\infty$ and the average is also $+\infty$. Taking the limit for the inverse gives the value 0 for RPSO. Conversely, if RPSO is 0, then its inverse is $+\infty$, which can only occur if one of

the $-\log(1 - V(\mathbf{1}_\varphi \mid \mathcal{O} = o))$ is $+\infty$ for some $o$. This, in turn, means that some $V(\mathbf{1}_\varphi \mid \mathcal{O} = o)$ is 1, which means the observation class of $o$ is contained either in $\varphi$ or in $\overline{\varphi}$.

(3) $\mathsf{PO}_r^S(\mathcal{A}, \varphi, \mathcal{O}) = 1$ iff $\sum_{o \in Obs} \mathbf{P}(\mathcal{O} = o) \cdot (-\log(1 - V(\mathbf{1}_\varphi \mid \mathcal{O} = o))) = 1$. Since this is an average of values above 1, this is equivalent to $-\log(1 - V(\mathbf{1}_\varphi \mid \mathcal{O} = o)) = 1$ for all $o \in Obs$, *i.e.* $V(\mathbf{1}_\varphi \mid \mathcal{O} = o) = \frac{1}{2}$ for all $o$. In this particular case, we also have $V(\mathbf{1}_\varphi \mid \mathcal{O} = o) = \frac{1}{2}$ iff $\mathbf{P}(\mathbf{1}_\varphi = 1 \mid \mathcal{O} = o) = \frac{1}{2}$ which concludes the proof. □

*Example 1: Sale protocol.* We consider the sale protocol from (Alvim et al., 2010), depicted in Figure 6. Two products can be put on sale, either a cheap or an expensive one, and two clients, either a rich or a poor one, may want to buy it. The products are put on sale according to a distribution ($\alpha$ and $\overline{\alpha} = 1 - \alpha$) while buyers behave probabilistically (through $\beta$ and $\gamma$) although differently according to the price of the item on sale. The

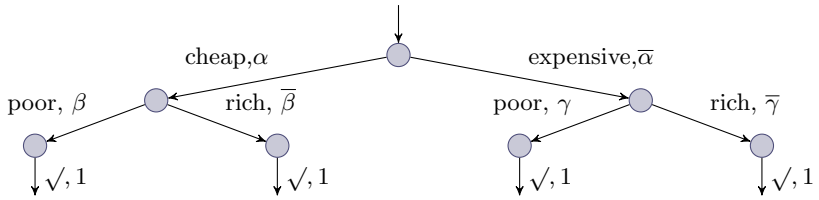
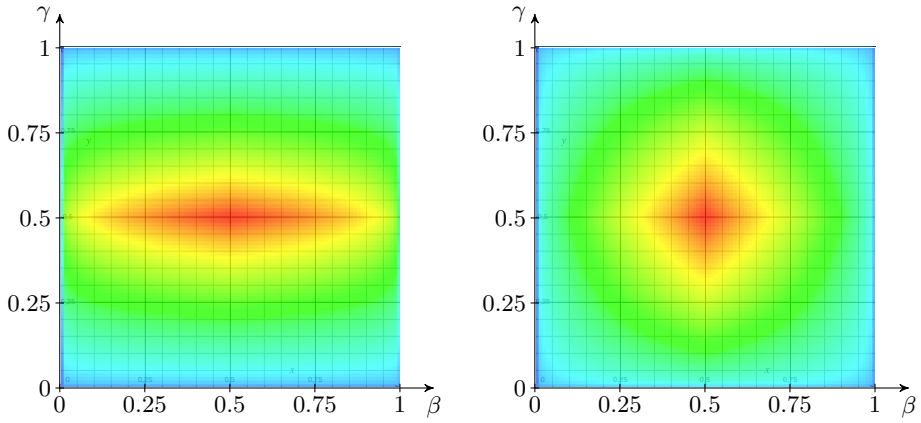



Figure 6. A simple sale protocol represented as an FPA $\mathcal{S}ale$.




(a) $\mathsf{PO}_r^S(\mathcal{S}ale, \varphi_{\text{poor}}, Price)$ when $\alpha = \frac{1}{8}$ (b) $\mathsf{PO}_r^S(\mathcal{S}ale, \varphi_{\text{poor}}, Price)$ when $\alpha = \frac{1}{2}$

Figure 7. RPSO for the sale protocol.

price of the item is public, but the identity of the buyer should remain secret. Hence the observation function *Price* yields *cheap* or *expensive*, while the secret is, without loss of symmetry, the set $\varphi_{\text{poor}}$ of runs ending with *poor*. The bias introduced by the preference of, say, a cheap item by the poor client betrays the secret identity of the buyer. RPSO

allows to measure this bias, and more importantly, to compare the bias obtained globally for different values of the parameters $\alpha$, $\beta$, and $\gamma$.

More formally, we have:

$$\mathbf{P}(Price = \text{cheap}) = \alpha \qquad \mathbf{P}(Price = \text{expensive}) = \overline{\alpha}$$

$$V(\mathbf{1}_{\varphi_{\text{poor}}} \mid \mathcal{O} = \text{cheap}) = \max(\beta, \overline{\beta}) \qquad V(\mathbf{1}_{\varphi_{\text{poor}}} \mid \mathcal{O} = \text{expensive}) = \max(\gamma, \overline{\gamma})$$

$$\mathsf{PO}_r^S(\mathcal{S}ale, \varphi_{\text{poor}}, Price) = \frac{-1}{\alpha \cdot \log(\min(\beta, \overline{\beta})) + \overline{\alpha} \cdot \log(\min(\gamma, \overline{\gamma}))}.$$

The variations of RPSO w.r.t. to $\beta$ and $\gamma$ is depicted for several values of $\alpha$ in Figure 7, red meaning higher value for RPSO. Thus, while the result is symmetric for $\alpha = \frac{1}{2}$, the case where $\alpha = \frac{1}{8}$ gives more importance to the fluctuations of $\gamma$.

*Example 2: Dining Cryptographers Protocol.* Introduced in (Chaum, 1988), this problem involves three cryptographers $C_1$, $C_2$ and $C_3$ dining in a restaurant. At the end of the meal, their master secretly tells each of them if they should be paying: $p_i = 1$ iff cryptographer $C_i$ pays, and $p_i = 0$ otherwise. Wanting to know if one of the cryptographers paid or if the master did, they follow the following protocol. They flip a coin with each of their neighbor, the third one not seeing the result of the flip, marking $f_{i,j} = 0$ if the coin flip between $i$ and $j$ was heads and $f_{i,j} = 1$ if it was tails. Then each cryptographer $C_i$, for $i \in \{1, 2, 3\}$, announces the value of $r_i = f_{i,i+1} \oplus f_{i,i-1} \oplus p_i$ (where '$3 + 1 = 1$', '$1 - 1 = 3$' and '$\oplus$' represents the XOR operator). If $\bigoplus_{i=1}^{3} r_i = 0$ then no one (*i.e.* the master) paid, if $\bigoplus_{i=1}^{3} r_i = 1$, then one of the cryptographers paid, but the other two do not know who he is.

Here we will use a simplified version of this problem to limit the size of the model. We consider that some cryptographer paid for the meal, and adopt the point of view of $C_1$ who did not pay. The anonymity of the payer is preserved if $C_1$ cannot know if $C_2$ or $C_3$ paid for the meal. In our setting, the predicate $\varphi_2$ is, without loss of symmetry, "$C_2$ paid". Note that predicate $\varphi_2$ is well suited for analysis of symmetrical opacity, since detecting that $\varphi_2$ is false gives information on who paid (here $C_3$). The observation function lets $C_1$ know the results of its coin flips ($f_{1,2}$ and $f_{1,3}$), and the results announced by the other cryptographers ($r_2$ and $r_3$). We also assume that the coin used by $C_2$ and $C_3$ has a probability of $q$ to yield heads, and that the master flips a fair coin to decide if $C_2$ or $C_3$ pays. It can be assumed that the coins $C_1$ flips with its neighbors are fair, since it does not affect anonymity from $C_1$'s point of view. In order to limit the (irrelevant) interleaving, we have made the choice to fix the ordering between the coin flips.

The corresponding FPA $\mathcal{D}$ is depicted on Figure 8 where all $\surd$ transitions with probability 1 have been omitted from final (rectangular) states. On $\mathcal{D}$, the runs satisfying predicate $\varphi_2$ are the ones where action $p_2$ appears. The observation function $\mathcal{O}_1$ takes a run and returns the sequence of actions over the alphabet $\{h_{1,2}, t_{1,2}, h_{1,3}, t_{1,3}\}$ and the final state reached, containing the value announced by $C_2$ and $C_3$.

There are 16 possible complete runs in this system, that yield 8 equiprobable observ-

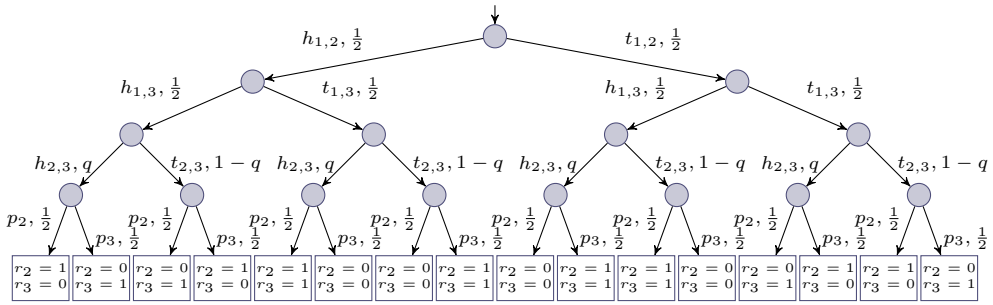



Figure 8. The FPA corresponding to the Dining Cryptographers protocol.

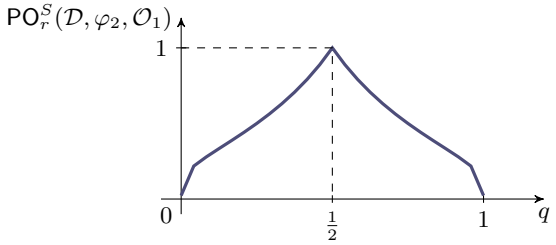



Figure 9. Evolution of the restrictive probabilistic symmetric opacity of the Dining Cryptographers protocol when changing the bias on the coin.

ables:

$$\begin{aligned} Obs = \{&(h_{1,2}h_{1,3}(r_2 = 1, r_3 = 0)), (h_{1,2}h_{1,3}(r_2 = 0, r_3 = 1)),\\ &(h_{1,2}t_{1,3}(r_2 = 0, r_3 = 0)), (h_{1,2}t_{1,3}(r_2 = 1, r_3 = 1)),\\ &(t_{1,2}h_{1,3}(r_2 = 0, r_3 = 0)), (t_{1,2}h_{1,3}(r_2 = 1, r_3 = 1)),\\ &(t_{1,2}t_{1,3}(r_2 = 1, r_3 = 0)), (t_{1,2}t_{1,3}(r_2 = 0, r_3 = 1))\ \} \end{aligned}$$

Moreover, each observation results in a run in which $C_2$ pays and a run in which $C_3$ pays, this difference being masked by the secret coin flip between them. For example, runs $\rho_h = h_{1,2}h_{1,3}h_{2,3}p_2(r_2 = 1, r_3 = 0)$ and $\rho_t = h_{1,2}h_{1,3}t_{2,3}p_3(r_2 = 1, r_3 = 0)$ yield the same observable $o_0 = h_{1,2}h_{1,3}(r_2 = 1, r_3 = 0)$, but the predicate is true in the first case and false in the second one. Therefore, if $0 < q < 1$, the unprobabilistic version of $\mathcal{D}$ is opaque. However, if $q \neq \frac{1}{2}$, for each observable, one of them is *more likely* to be lying, therefore paying. In the aforementioned example, when observing $o_0$, $\rho_h$ has occurred with probability $q$, whereas $\rho_t$ has occurred with probability $1 - q$. RPSO can measure this advantage globally.

For each observation class, the vulnerability of $\varphi_2$ is $\max(q, 1 - q)$. Hence the RPSO will be

$$\mathsf{PO}_r^S(\mathcal{D}, \varphi_2, \mathcal{O}_1) = \frac{-1}{\log(\min(q, 1 - q))}$$

The variations of the RPSO when changing the bias on $q$ are depicted in Figure 9. Analysis of RPSO according to the variation of $q$ yields that the system is perfectly secure if there is no bias on the coin, and insecure if $q = 0$ or $q = 1$.

## 5. Computing opacity measures

We now show how all measures defined above can be computed for regular predicates and simple observation functions. The method relies on a synchronized product between an SA $\mathcal{A}$ and a deterministic FA $\mathcal{K}$, similarly to (Courcoubetis and Yannakakis, 1998). This product (which can be considered pruned of its unreachable states and states not reaching a final state) constrains the unprobabilistic version of $\mathcal{A}$ by synchronizing it with $\mathcal{K}$. The probability of $\mathcal{L}(\mathcal{K})$ is then obtained by solving a system of equations associated with this product. The computation of all measures results in applications of this operation with several automata.

### 5.1. *Computing the probability of a substochastic automaton*

Given an SA $\mathcal{A}$, a system of equations can be derived on the probabilities for each state to yield an accepting run. This allows to compute the probability of all complete runs of $\mathcal{A}$ by a technique similar to those used in (Courcoubetis and Yannakakis, 1998; Hansson and Jonsson, 1994; Bianco and de Alfaro, 1995) for probabilistic verification.

**Definition 5.1 (Linear system of a substochastic automata).** Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0 \rangle$ be a substochastic automaton where any state can reach a final state. The *linear system associated with* $\mathcal{A}$ is the following system $\mathcal{S}_\mathcal{A}$ of linear equations over $\mathbb{R}$:

$$\mathcal{S}_\mathcal{A} = \left( X_q = \sum_{q' \in Q} \alpha_{q,q'} X_{q'} + \beta_q \right)_{q \in Q}$$

$$\text{where} \quad \alpha_{q,q'} = \sum_{a \in \Sigma} \Delta(q)(a, q') \text{ and } \beta_q = \Delta(q)(\surd)$$

When non-determinism is involved, for instance in Markov Decision Processes (Courcoubetis and Yannakakis, 1998; Bianco and de Alfaro, 1995), two systems of inequations are needed to compute maximal and minimal probabilities. Here, without non-determinism, both values are the same, hence Lemma 5.2 is a particular case of the results in (Courcoubetis and Yannakakis, 1998; Bianco and de Alfaro, 1995), where uniqueness is ensured by the hypothesis (any state can reach a final state). The probability can thus be computed in polynomial time by solving the linear system associated with the SA.

**Lemma 5.2.** Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0 \rangle$ be a substochastic automaton and define for all $q \in Q$, $L_q^\mathcal{A} = \mathbf{P}(CRun_q(\mathcal{A}))$. Then $(L_q^\mathcal{A})_{q \in Q}$ is the unique solution of the system $\mathcal{S}_\mathcal{A}$.

### 5.2. *Computing the probability of a regular language*

In order to compute the probability of a language inside a system, we build a substochastic automaton that corresponds to the intersection of the system and the language, then compute the probability as above.

**Definition 5.3 (Synchronized product).** Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0 \rangle$ be a substochastic automaton and let $\mathcal{K} = \langle Q \times \Sigma \times Q, Q_K, \Delta_K, q_K, F \rangle$ be a deterministic finite automaton.

The synchronized product $\mathcal{A}||\mathcal{K}$ is the substochastic automaton $\langle \Sigma, Q \times Q_K, \Delta', (q_0, q_K)\rangle$ where transitions in $\Delta'$ are defined by: if $q_1 \to \mu \in \Delta$, then $(q_1, r_1) \to \nu \in \Delta'$ where for all $a \in \Sigma$ and $(q_2, r_2) \in Q \times Q_K$,

$$\nu(a,(q_2,r_2)) = \begin{cases} \mu(a,q_2) & \text{if } r_1 \xrightarrow{q_1,a,q_2} r_2 \in \Delta_K \\ 0 & \text{otherwise} \end{cases}$$

$$\text{and} \quad \nu(\surd) = \begin{cases} \mu(\surd) & \text{if } r_1 \in F \\ 0 & \text{otherwise} \end{cases}$$

In this synchronized product, the behaviors are constrained by the finite automaton. Actions not allowed by the automaton are trimmed, and states can accept only if they correspond to a valid behavior of the DFA. Note that this product is defined on SA in order to allow several intersections. The correspondence between the probability of a language in a system and the probability of the synchronized product is laid out in the following lemma.

**Lemma 5.4.** Let $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0\rangle$ be an SA and $K$ a regular language over $Q \times \Sigma \times Q$ accepted by a deterministic finite automaton $\mathcal{K} = \langle Q \times \Sigma \times Q, Q_K, \Delta_K, q_K, F\rangle$. Then

$$\mathbf{P}_{\mathcal{A}}(K) = L^{\mathcal{A}||\mathcal{K}}_{(q_0,q_K)}$$

*Proof.* Let $\rho \in CRun(\mathcal{A})$ with $\mathrm{tr}(\rho) \in K$ and $\rho = q_0 \xrightarrow{a_1} q_1 \cdots \xrightarrow{a_n} q_n\surd$. Since $\mathrm{tr}(\rho) \in K$ and $\mathcal{K}$ is deterministic, there is a unique run $\rho_K = q_K \xrightarrow{a_1} r_1 \cdots \xrightarrow{a_n} r_n$ in $\mathcal{K}$ with $r_n \in F$. Then the sequence $\rho' = (q_0, q_K) \xrightarrow{a_1} (q_1, r_1) \cdots \xrightarrow{a_n} (q_n, r_n)$ is a run of $\mathcal{A}||\mathcal{K}$. There is a one-to one-match between runs of $\mathcal{A}||\mathcal{K}$ and pairs of runs in $\mathcal{A}$ and $\mathcal{K}$ with the same trace. Moreover,

$$\begin{aligned} \mathbf{P}_{\mathcal{A}||\mathcal{K}}(\rho') &= \Delta'(q_0,q_K)(a_1,(q_1,r_1)) \times \cdots \times \Delta'(q_n,r_n)(\surd) \\ &= \Delta(q_0)(a_1,q_1) \times \cdots \times \Delta(q_n)(\surd) \\ \mathbf{P}_{\mathcal{A}||\mathcal{K}}(\rho') &= \mathbf{P}_{\mathcal{A}}(\rho). \end{aligned}$$

Hence

$$\mathbf{P}_{\mathcal{A}}(K) = \sum_{\{\rho|\mathrm{tr}(\rho)\in K\}} \mathbf{P}_{\mathcal{A}}(\rho) = \sum_{\rho' \in Run(\mathcal{A}||\mathcal{K})} \mathbf{P}_{\mathcal{A}||\mathcal{K}}(\rho') = \mathbf{P}_{\mathcal{A}||\mathcal{K}}(Run(\mathcal{A}||\mathcal{K}))$$

and therefore from Lemma 5.2, $\mathbf{P}_{\mathcal{A}}(K) = L^{\mathcal{A}||\mathcal{K}}_{(q_0,q_0')}$. □

### 5.3. *Computing all opacity measures*

All measures defined previously can be computed as long as, for $i \in \{0,1\}$ and $o \in Obs$, all probabilities

$$\mathbf{P}(\mathbf{1}_\varphi = i) \qquad \mathbf{P}(\mathcal{O} = o) \qquad \mathbf{P}(\mathbf{1}_\varphi = i, \mathcal{O} = o)$$

can be computed. Indeed, even deciding whether $\mathcal{O}^{-1}(o) \subseteq \varphi$ can be done by testing $\mathbf{P}(\mathcal{O} = o) > 0 \wedge \mathbf{P}(\mathbf{1}_\varphi = 0, \mathcal{O} = o) = 0$.

Now suppose $Obs$ is a finite set, $\varphi$ and all $\mathcal{O}^{-1}(o)$ are regular sets. Then one can build

deterministic finite automata $\mathcal{A}_\varphi$, $\mathcal{A}_{\overline{\varphi}}$, $\mathcal{A}_o$ for $o \in Obs$ that accept respectively $\varphi$, $\overline{\varphi}$, and $\mathcal{O}^{-1}(o)$.

Synchronizing automaton $\mathcal{A}_\varphi$ with $\mathcal{A}$ and pruning it yields a substochastic automaton $\mathcal{A}||\mathcal{A}_\varphi$. By Lemma 5.4, the probability $\mathbf{P}(\mathbf{1}_\varphi = 1)$ is then computed by solving the linear system associated with $\mathcal{A}||\mathcal{A}_\varphi$. Similarly, one obtain $\mathbf{P}(\mathbf{1}_\varphi = 0)$ (with $\mathcal{A}_{\overline{\varphi}}$), $\mathbf{P}(\mathcal{O} = o)$ (with $\mathcal{A}_o$), $\mathbf{P}(\mathbf{1}_\varphi = 1, \mathcal{O} = o)$ (synchronizing $\mathcal{A}||\mathcal{A}_\varphi$ with $\mathcal{A}_o$), and $\mathbf{P}(\mathbf{1}_\varphi = 0, \mathcal{O} = o)$ (synchronizing $\mathcal{A}||\mathcal{A}_{\overline{\varphi}}$ with $\mathcal{A}_o$).

**Theorem 5.5.** Let $\mathcal{A}$ be an FPA. If $Obs$ is a finite set, $\varphi$ is a regular set and for $o \in Obs$, $\mathcal{O}^{-1}(o)$ is a regular set, then for $\mathsf{PO} \in \{\mathsf{PO}^A_\ell, \mathsf{PO}^S_\ell, \mathsf{PO}^A_r, \mathsf{PO}^S_r\}$, $\mathsf{PO}(\mathcal{A}, \varphi, \mathcal{O})$ can be computed.

The computation of opacity measures is done in polynomial time in the size of $Obs$ and DFAs $\mathcal{A}_\varphi$, $\mathcal{A}_{\overline{\varphi}}$, $\mathcal{A}_o$.

A prototype tool implementing this algorithm was developed in Java (Eftenie, 2010), yielding numerical values for measures of opacity.

## 6. Comparison of the measures of opacity

In this section we compare the discriminating power of the measures discussed above. As described above, the liberal measures evaluate the leak, hence 0 represents the best possible value from a security point of view, producing an opaque system. For such an opaque system, the restrictive measure evaluate the robustness of this opacity. As a result, 1 is the best possible value.

### 6.1. *Abstract examples*

The values of these metrics are first compared for extremal cases of Figure 10. These values are displayed in Table 3.

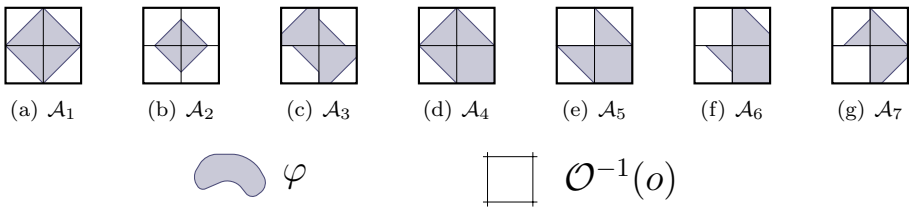



Figure 10. Example of repartition of probabilities of $\mathbf{1}_\varphi$ and $\mathcal{O}$ in 7 cases.

First, the system $\mathcal{A}_1$ of Figure 10(a) is intuitively very secure since, with or without observation, an attacker has no information whether $\varphi$ was true or not. This security is reflected in all symmetrical measures, with highest scores possibles in all cases. It is nonetheless deemed more insecure for RPO, since opacity is perfect when $\varphi$ is always false.

The case of $\mathcal{A}_2$ of Figure 10(b) only differs from $\mathcal{A}_1$ by the global repartition of $\varphi$

| System | (a) $\mathcal{A}_1$ | (b) $\mathcal{A}_2$ | (c) $\mathcal{A}_3$ | (d) $\mathcal{A}_4$ | (e) $\mathcal{A}_5$ | (f) $\mathcal{A}_6$ | (g) $\mathcal{A}_7$ |
|---|---|---|---|---|---|---|---|
| LPO | 0 | 0 | 0 | $\frac{1}{4}$ | $\frac{1}{4}$ | $\frac{1}{4}$ | 0 |
| LPSO | 0 | 0 | 0 | $\frac{1}{4}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{4}$ |
| RPO | $\frac{1}{2}$ | $\frac{3}{4}$ | $\frac{3}{8}$ | 0 | 0 | 0 | $\frac{12}{25}$ |
| RPSO | 1 | $\frac{1}{2}$ | $\frac{1}{2}$ | 0 | 0 | 0 | 0 |

Table 3. Values of the different opacity measures for systems of Figure 10(a)-(g).

in $Run(\mathcal{A})$. The information an attacker gets comes not from the observation, but from $\varphi$ itself. Therefore RPSO, which does not remove the information available before observation, evaluates this system as less secure than $\mathcal{A}_1$. Measures based on information theory (Smith, 2009; Bérard et al., 2010) would consider this system as secure. However, such measures lack strong ties with opacity, which depend only on the information available to the observer, wherever this information comes from. In addition, RPO finds $\mathcal{A}_2$ more secure than $\mathcal{A}_1$: $\varphi$ is verified less often. Note that the complement would not change the value for symmetrical measures, while being insecure for RPO (with $\mathsf{PO}_r^A = \frac{1}{4}$).

However, since each observation class is considered individually, RPSO does not discriminate $\mathcal{A}_2$ and $\mathcal{A}_3$ of Figure 10(c). Here, the information is the same in each observation class as for $\mathcal{A}_2$, but the repartition of $\varphi$ gives no advantage at all to an attacker without observation.

When the system is not opaque (resp. symmetrically opaque), RPO (resp. RPSO) cannot discriminate them, and LPO (resp. LPSO) becomes relevant. For example, $\mathcal{A}_4$ is not opaque for the classical definitions, therefore $\mathsf{PO}_r^A = \mathsf{PO}_r^S = 0$ and both $\mathsf{PO}_\ell^A > 0$ and $\mathsf{PO}_\ell^S > 0$.

System, $\mathcal{A}_5$ of Figure 10(e) has a greater $\mathsf{PO}_\ell^S$ than $\mathcal{A}_4$. However, LPO is unchanged since the class completely out of $\varphi$ is not taken into account. Remark that system $\mathcal{A}_7$ is opaque but not symmetrically opaque, hence the relevant measures are $\mathsf{PO}_\ell^S$ and $\mathsf{PO}_r^A$. Also note that once a system is not opaque, the repartition of classes that do not leak information is not taken into account, hence equal values in the cases of $\mathcal{A}_5$ and $\mathcal{A}_6$.

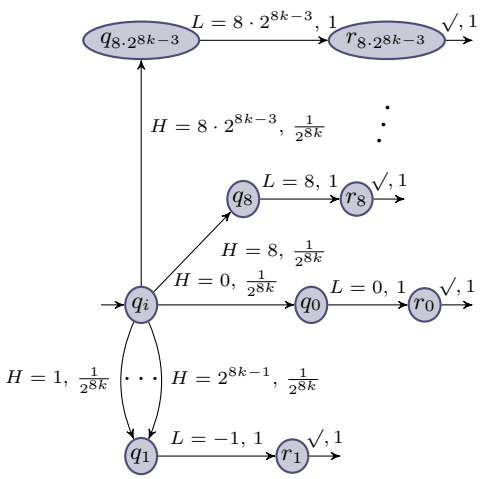
### 6.2. *A more concrete example*

Consider the following programs $P_1$ and $P_2$, inspired from (Smith, 2009), where $k$ is a given parameter, `random` select uniformly an integer value (in binary) between its two arguments and & is the bitwise *and*:

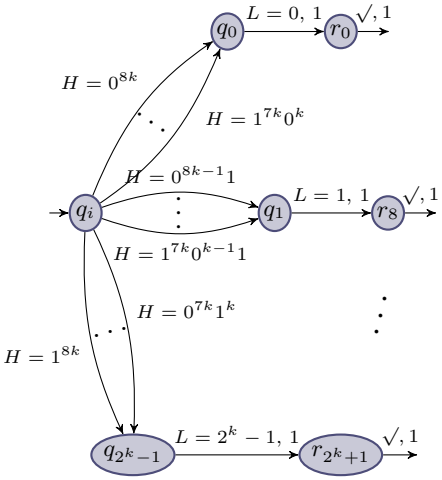
$P_1$:
$H := \texttt{random}(0, 2^{8k} - 1)$;
if $H \mod 8 = 0$ then
  $L := H$
else
  $L := -1$
fi

$P_2$:
$H := \texttt{random}(0, 2^{8k} - 1)$;
$L := H \ \& \ 0^{7k}1^k$

In both cases, the value of $H$, an integer over $8k$ bits, is supposed to remain secret, and cannot be observed directly, while the value of $L$ is public. Thus the observation is




(a) FPA $\mathcal{A}_{P_1}$




(b) FPA $\mathcal{A}_{P_2}$; all edges stemming from $q_i$ have equal probability $\frac{1}{2^{8k}}$

Figure 11. FPAs for programs $P_1$ and $P_2$.

| Program | $\mathsf{PO}^A_\ell$ | $\mathsf{PO}^S_\ell$ | $\mathsf{PO}^A_r$ | $\mathsf{PO}^S_r$ |
|---|---|---|---|---|
| $P_1$ | $\frac{1}{8}$ | 1 | 0 | 0 |
| $P_2$ | 0 | 0 | $1 - \frac{1}{2^{7k}}$ | $\frac{1}{7k}$ |

Table 4. Opacity measures for programs $P_1$ and $P_2$.

the "$L := \ldots$" action. Intuitively, $P_1$ divulges the exact value of $H$ with probability $\frac{1}{8}$. On the other hand, $P_2$ leaks the value of one eighth of its bits (the least significant ones) at every execution.

These programs can be translated into FPAs $\mathcal{A}_{P_1}$ and $\mathcal{A}_{P_2}$ of Figure 11. In order to have a boolean predicate, the secret is not the value of variable $H$, but whether $H = L$: $\varphi_= = \{(H = x)(L = x) \mid x \in \{0, \ldots, 2^{8k} - 1\}\}$. Non opacity then means that the attacker discovers the secret value. Weaker predicates can also be considered, like equality of $H$ with a particular value or $H$ belonging to a specified subset of values, but we chose the simplest one. First remark that $\varphi_=$ is not opaque on $P_1$ in the classical sense (both symmetrically or not). Hence both RPO and RPSO are null. On the other hand, $\varphi_=$ is opaque on $P_2$, hence LPO and LPSO are null. The values for all measures are gathered in Table 4. Note that only restrictive opacity for $P_2$ depends on $k$. This comes from the fact that in all other cases, both $\varphi_=$ and the equivalence classes scale at the same rate with $k$. In the case of $P_2$, adding length to the secret variable $H$ *dilutes* $\varphi_=$ inside each class. Hence the greater $k$ is, the hardest it is for an attacker to know that $\varphi_=$ is true, thus to crack asymmetrical opacity. Indeed, it will tend to get false in most cases, thus providing an easy guess, and a low value for symmetrical opacity.

### 6.3. *Crowds protocol*

The anonymity protocol known as *crowds* was introduced in (Reiter and Rubin, 1998) and recently studied in the probabilistic framework in (Chatzikokolakis et al., 2008; Andrés et al., 2010). When a user wants to send a message (or request) to a server without the latter knowing the origin of the message, the user routes the message through a crowd of $n$ users. To do so, it selects a user randomly in the crowd (including himself), and sends him the message. When a user receives a message to be routed according to this protocol, it either sends the message to the server with probability $1-q$ or forwards it to a user in the crowd, with probability $q$. The choice of a user in the crowd is always equiprobable. Under these assumptions, this protocol is known to be secure, since no user is more likely than another to be the actual initiator; indeed its RPO is very low. However, there can be $c$ corrupt users in the crowd which divulge the identity of the person that sent the message to them. In that case, if a user sends directly a message to a corrupt user, its identity is no longer protected. The goal of corrupt users is therefore not to transmit messages, hence they cannot initiate the protocol. The server and the corrupt users cooperate to discover the identity of the initiator. RPO can measure the security of this system, depending on $n$ and $c$.

First, consider our protocol as the system in Figure 12. In this automaton, states $1,\ldots,n-c$ corresponding to honest users are duplicated in order to differentiate their behavior as initiator or as the receiver of a message from the crowd. The predicate we want to be opaque is $\varphi_i$ that contains all the runs in which $i$ is the initiator of the request. The observation function $\mathcal{O}$ returns the penultimate state of the run, *i.e.* the honest user that will be seen by the server or a corrupt user.

For sake of brevity, we write '$i \rightsquigarrow$' to denote the event "a request was initiated by $i$" and '$\rightsquigarrow j$' when "$j$ was detected by the adversary", which means that $j$ sent the message either to a corrupt user or to the server, who both try to discover who the initiator was. The abbreviation $i \rightsquigarrow j$ stands for $i \rightsquigarrow \wedge \rightsquigarrow j$. Notation '$\neg i \rightsquigarrow$' means that "a request was initiated by someone else than $i$"; similarly, combinations of this notations are used in the sequel. We also use the Kronecker symbol $\delta_{ij}$ defined by $\delta_{ij}=1$ if $i=j$ and 0 otherwise.

*Computation of the probabilities.* All probabilities $\mathbf{P}(i \rightsquigarrow j)$ can be automatically computed using the method described in Section 5. For example, $\mathbf{P}(1 \rightsquigarrow (n-c))$, the probability for the first user to initiate the protocol while the last honest user is detected, can be computed from substochastic automaton $\mathcal{C}_n^c || \mathcal{A}_{1 \rightsquigarrow (n-c)}$ depicted on Figure 13. In this automaton, the only duplicated state remaining is $1'$.

This SA can also be represented by a transition matrix (like a Markov chain), which is given in Table 5. An additional column indicates the probability for the $\surd$ action, which ends the run (here it is either 1 if the state is final and 0 if not).

The associated system is represented in Table 6 where $L_S$ corresponds to the "Server" state. Each line of the system is given by the outgoing probabilities of the corresponding state in the SA, or alternatively by the corresponding line of the matrix. Resolving it

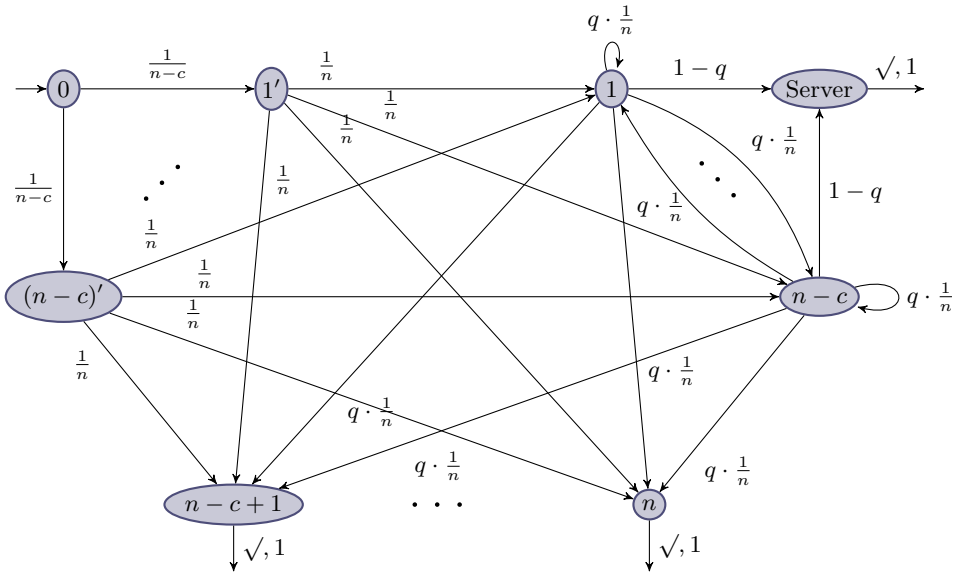



Figure 12. FPA $\mathcal{C}_n^c$ for Crowds protocol with $n$ users, among whom $c$ are corrupted.

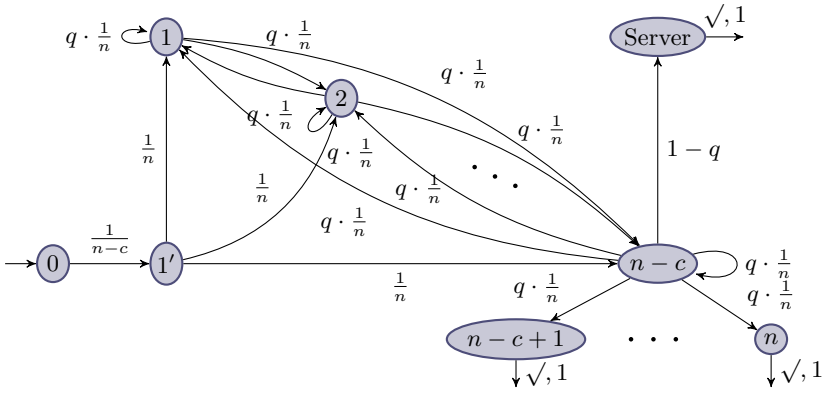



Figure 13. SA $\mathcal{C}_n^c || \mathcal{A}_{1 \rightsquigarrow (n-c)}$ corresponding to runs where user 1 initiates the protocol and user $(n-c)$ is detected.

(see Appendix B) yields, $L_i = \frac{q}{n}$ for all $i \in \{1, \ldots, n-c-1\}$, $L_{n-c} = 1 - \frac{q \cdot (n-c-1)}{n}$, $L_{1'} = \frac{1}{n}$, and $L_0 = \frac{1}{(n-c) \cdot n}$. Therefore, $\mathbf{P}(1 \rightsquigarrow (n-c)) = \frac{1}{(n-c) \cdot n}$.

In this case, simple reasoning on the symmetries of the model allows to derive other probabilities $\mathbf{P}(i \rightsquigarrow j)$. Remark that the probability for a message to go directly from initiator to the adversay (who cannot be the server) is $\frac{c}{n}$: it only happens if a corrupt user is chosen by the initiator. If a honest user is chosen by the initiator, then the length of the path will be greater, with probability $\frac{n-c}{n}$. By symmetry all honest users

| | 0 | $1'$ | 1 | $\cdots$ | $n-c$ | $n-c+1$ | $\cdots$ | $n$ | *Server* | $\surd$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | $\frac{1}{n-c}$ | 0 | $\cdots$ | 0 | 0 | $\cdots$ | 0 | 0 | 0 |
| $1'$ | 0 | 0 | $\frac{1}{n}$ | $\cdots$ | $\frac{1}{n}$ | 0 | $\cdots$ | 0 | 0 | 0 |
| 1 | 0 | 0 | $q\cdot\frac{1}{n}$ | $\cdots$ | $q\cdot\frac{1}{n}$ | 0 | $\cdots$ | 0 | 0 | 0 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $n-c-1$ | 0 | 0 | $q\cdot\frac{1}{n}$ | $\cdots$ | $q\cdot\frac{1}{n}$ | 0 | $\cdots$ | 0 | 0 | 0 |
| $n-c$ | 0 | 0 | $q\cdot\frac{1}{n}$ | $\cdots$ | $q\cdot\frac{1}{n}$ | $q\cdot\frac{1}{n}$ | $\cdots$ | $q\cdot\frac{1}{n}$ | $1-q$ | 0 |
| $n-c+1$ | 0 | 0 | 0 | $\cdots$ | 0 | 0 | $\cdots$ | 0 | 0 | 1 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $n$ | 0 | 0 | 0 | $\cdots$ | 0 | 0 | $\cdots$ | 0 | 0 | 1 |
| *Server* | 0 | 0 | 0 | $\cdots$ | 0 | 0 | $\cdots$ | 0 | 0 | 1 |

Table 5. The matrix giving the transition probabilities between the states of $\mathcal{C}_n^c||\mathcal{A}_{1\rightsquigarrow(n-c)}$.

$$\left\{\begin{array}{rcl} L_0 &=& \frac{1}{n-c}\cdot L_{1'} \\ L_{1'} &=& \sum_{i=1}^{n-c}\frac{1}{n}\cdot L_i \\ L_1 &=& \sum_{i=1}^{n-c}\frac{q}{n}\cdot L_i \\ &\vdots& \\ L_{n-c-1} &=& \sum_{i=1}^{n-c}\frac{q}{n}\cdot L_i \end{array}\right. \qquad \left\{\begin{array}{rcl} L_{n-c} &=& (1-q)\cdot L_S + \sum_{i=1}^{n}\frac{q}{n}\cdot L_i \\ L_{n-c+1} &=& 1 \\ &\vdots& \\ L_n &=& 1 \\ L_S &=& 1 \end{array}\right.$$

Table 6. Linear system associated to SA $\mathcal{C}_n^c||\mathcal{A}_{1\rightsquigarrow(n-c)}$ of Figure 13.

have equal probability to be the initiator, and equal probability to be detected. Hence $\mathbf{P}(i\rightsquigarrow) = \mathbf{P}(\rightsquigarrow j) = \frac{1}{n-c}$.

Event $i\rightsquigarrow j$ occurs when $i$ is chosen as the initiator (probability $\frac{1}{n-c}$), and either (1) if $i=j$ and $i$ chooses a corrupted user to route its message, or (2) if a honest user is chosen and $j$ sends the message to a corrupted user or the server (the internal route between honest users before $j$ is irrelevant). Therefore

$$\begin{aligned} \mathbf{P}(i\rightsquigarrow j) &= \frac{1}{n-c}\cdot\left(\delta_{ij}\cdot\frac{c}{n}+\frac{1}{n-c}\cdot\frac{n-c}{n}\right) \\ \mathbf{P}(i\rightsquigarrow j) &= \frac{1}{n-c}\cdot\left(\delta_{ij}\cdot\frac{c}{n}+\frac{1}{n}\right) \end{aligned}$$

The case when $i$ is not the initiator is derived from this probability:

$$\begin{aligned} \mathbf{P}(\neg i\rightsquigarrow j) &= \sum_{\substack{k=1\\k\neq i}}^{n-c}\mathbf{P}(k\rightsquigarrow j) \\ \mathbf{P}(\neg i\rightsquigarrow j) &= \frac{1}{n-c}\cdot\left((1-\delta_{ij})\cdot\frac{c}{n}+\frac{n-c-1}{n}\right) \end{aligned}$$

Conditional probabilities thus follow:

$$\mathbf{P}(i\rightsquigarrow \mid \rightsquigarrow j) = \frac{\mathbf{P}(i\rightsquigarrow j)}{\mathbf{P}(\rightsquigarrow j)} = \delta_{ij}\cdot\frac{c}{n}+\frac{1}{n}$$

$$\mathbf{P}(\neg i \rightsquigarrow \mid \rightsquigarrow j) = \frac{\mathbf{P}(\neg i \rightsquigarrow j)}{\mathbf{P}(\rightsquigarrow j)} = (1-\delta_{ij}) \cdot \frac{c}{n} + \frac{n-c-1}{n}$$

Interestingly, these probabilities do not depend on $q$[‡].

*Computation of RPO.* From the probabilities above, one can compute an analytical value for $\mathsf{PO}_r^A(\mathcal{C}_n^c, \mathbf{1}_{\varphi_i}, \mathcal{O})$.

$$\begin{aligned}
\frac{1}{\mathsf{PO}_r^A(\mathcal{C}_n^c, \varphi_i, \mathcal{O})} &= \sum_{j=1}^{n-c} \mathbf{P}(\rightsquigarrow j) \frac{1}{\mathbf{P}(\neg i \rightsquigarrow \mid \rightsquigarrow j)} \\
&= (n-c-1) \cdot \frac{1}{n-c} \cdot \frac{n}{n-1} + \frac{1}{n-c} \cdot \frac{n}{n-c-1} \\
&= \frac{n}{n-c} \left( \frac{n-c-1}{n-1} + \frac{1}{n-c-1} \right) \\
\frac{1}{\mathsf{PO}_r^A(\mathcal{C}_n^c, \varphi_i, \mathcal{O})} &= \frac{n \cdot (n^2 + c^2 - 2nc - n + 2c)}{(n-c) \cdot (n-1) \cdot (n-c-1)}
\end{aligned}$$

Hence

$$\mathsf{PO}_r^A(\mathcal{C}_n^c, \varphi_i, \mathcal{O}) = \frac{(n-c) \cdot (n-1) \cdot (n-c-1)}{n \cdot (n^2 + c^2 - 2nc - n + 2c)}$$

which tends to 1 as $n$ increases to $+\infty$ (for a fixed number of corrupted users). The evolution of RPO is represented in Figure 14(a) where blue means low and red means high. If the proportion of corrupted users is fixed, say $n = 4c$, we obtain

$$\mathsf{PO}_r^A(\mathcal{C}_{4c}^c, \varphi_i, \mathcal{O}) = \frac{(4c-1) \cdot (9c-3)}{4c \cdot (9c-2)}$$

which also tends to 1 as the crowds size increases. When there are no corrupted users,

$$\mathsf{PO}_r^A(\mathcal{C}_n^0, \varphi_i, \mathcal{O}) = \frac{n-1}{n},$$

which is close to 1, but never exactly, since $\varphi_i$ is not always false, although of decreasing proportion as the crowds grows. This result has to be put in parallel with the one from (Reiter and Rubin, 1998), which states that crowds is secure since each user is "beyond suspicion" of being the initiator, but "absolute privacy" is not achieved.

*Computation of RPSO.* From the probabilities computed above, we obtain that if $i \neq j$,

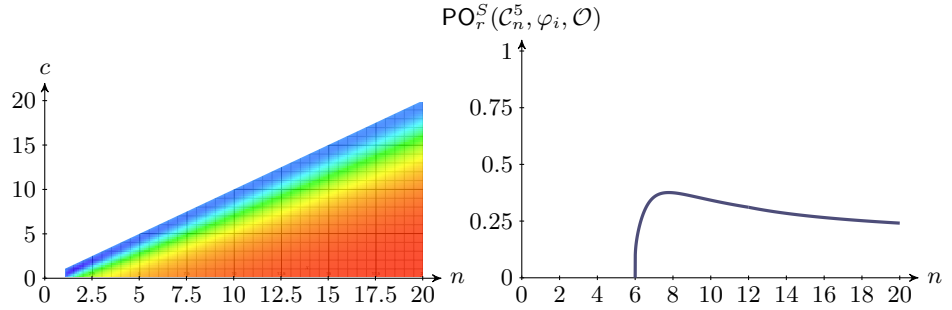
$$V(i \rightsquigarrow \mid \rightsquigarrow j) = \max\left(\frac{1}{n}, \frac{n-1}{n}\right) = \frac{1}{n}\max(1, n-1).$$

Except in the case of $n = 1$ (when the system is non-opaque, hence $\mathsf{PO}_r^S(\mathcal{C}_1^0, \varphi_1, \mathcal{O}) = 0$), $V(i \rightsquigarrow \mid \rightsquigarrow j) = \frac{n-1}{n}$.

In the case when $i = j$

$$V(i \rightsquigarrow \mid \rightsquigarrow i) = \max\left(\frac{c+1}{n}, \frac{n-c-1}{n}\right) = \frac{1}{n}\max(c+1, n-c-1).$$

[‡] This stems from the fact that the original models had either the server or the corrupt users as attackers, not both at the same time.




(a) Evolution of $\mathsf{PO}_r^A(\mathcal{C}_n^c, \varphi_i, \mathcal{O})$ with $n$ and $c$. Red meaning a value close to 1 and blue meaning close to 0.

(b) Evolution of $\mathsf{PO}_r^S(\mathcal{C}_n^c, \varphi_i, \mathcal{O})$ with $n$ when $c = 5$.

Figure 14. Evolution of restrictive opacity with the size of the crowd.

That means the vulnerability for the observation class corresponding to the case when $i$ is actually detected depends on the proportion of corrupted users in the crowd. Indeed, $V(i \rightsquigarrow \mid \rightsquigarrow i) = \frac{c+1}{n}$ if and only if $n \leq 2(c+1)$. The two cases shall be separated.

**When $n \leq 2(c+1)$.** The message is initially more likely to be sent to a corrupt user or to the initiator himself than to any other user in the crowd:

$$\begin{aligned}\sum_{j=1}^{n-c} \mathbf{P}(\rightsquigarrow j) \cdot \log(1 - V(i \rightsquigarrow \mid \rightsquigarrow j)) &= \frac{(n-c-1) \cdot \log\left(\frac{1}{n}\right) + \log\left(\frac{n-c-1}{n}\right)}{n-c} \\ &= \frac{1}{n-c} \cdot (\log(n-c-1) - (n-c) \cdot \log(n)) \\ &= \frac{\log(n-c-1)}{n-c} - \log(n)\end{aligned}$$

$$\text{Hence} \quad \mathsf{PO}_r^S(\mathcal{C}_n^c, \varphi_i, \mathcal{O}) = \frac{1}{\log(n) - \frac{\log(n-c-1)}{n-c}}$$

**When $n > 2(c+1)$.** The message is initially more likely to be sent to a honest user different from the initiator:

$$\begin{aligned}\sum_{j=1}^{n-c} \mathbf{P}(\rightsquigarrow j) \cdot \log(1 - V(i \rightsquigarrow \mid \rightsquigarrow j)) &= \frac{(n-c-1) \cdot \log\left(\frac{1}{n}\right) + \log\left(\frac{c+1}{n}\right)}{n-c} \\ &= \frac{1}{n-c} \cdot (\log(c+1) - (n-c) \cdot \log(n)) \\ &= \frac{\log(c+1)}{n-c} - \log(n)\end{aligned}$$

$$\text{Hence} \quad \mathsf{PO}_r^S(\mathcal{C}_n^c, \varphi_i, \mathcal{O}) = \frac{1}{\log(n) - \frac{\log(c+1)}{n-c}}$$

The evolution of RPSO for $c = 5$ is depicted in Figure 14(b).

One can see that actually the RPSO decreases when $n$ increases. That is because when there are more users in the crowd, user $i$ is less likely to be the initiator. Hence the

predicate chosen does not model anonymity as specified in (Reiter and Rubin, 1998) but a stronger property since RPSO is based on the definition of symmetrical opacity. Therefore it is meaningful in terms of security properties only when both the predicate and its negation are meaningful.

## 7. Dealing with nondeterminism

The measures presented above were all defined in the case of fully probabilistic finite automata. However, some systems present nondeterminism that cannot reasonably be abstracted away. For example, consider the case of a system, in which a malicious user Alice can control certain actions. The goal of Alice is to establish a covert communication channel with an external observer Bob. Hence she will try to influence the system in order to render communication easier. Therefore, the actual security of the system as observed by Bob should be measured against the best possible actions for Alice. Formally, Alice is a scheduler who, when facing several possible output distributions $\{\mu_1, \ldots, \mu_n\}$, can choose whichever distribution $\nu$ on $\{1, \ldots, n\}$ as weights for the $\mu_i$s. The security as measured by opacity is the minimal security of all possible successive choices.

### 7.1. *The nondeterministic framework*

Here we enlarge the setting of probabilistic automata considered before with nondeterminism. There are several outgoing distribution from a given state instead of a single one.

**Definition 7.1 (Nondeterministic probabilistic automaton).** A *nondeterministic probabilistic automaton* (NPA) is a tuple $\langle \Sigma, Q, \Delta, q_0 \rangle$ where

— $\Sigma$ is a finite set of actions;
— $Q$ is a finite set of states;
— $\Delta : Q \to \mathcal{P}(\mathcal{D}((\Sigma \times Q) \uplus \{\surd\}))$ is a nondeterministic probabilistic transition function;
— $q_0$ is the initial state;

where $\mathcal{P}(A)$ denotes the set of finite subsets of $A$.

The choice over the several possible distributions is made by the *scheduler*. It does not, however, selects one distribution to be used, but can give weight to the possible distributions.

**Definition 7.2 (Scheduler).** A scheduler on $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0 \rangle$ is a function

$$\sigma : Run(\mathcal{A}) \to \mathcal{D}(\mathcal{D}((\Sigma \times Q) \uplus \{\surd\}))$$

such that $\sigma(\rho)(\nu) > 0 \Rightarrow \nu \in \Delta(\mathrm{lst}(\rho))$.

The set of all schedulers for $\mathcal{A}$ is denoted $Sched_{\mathcal{A}}$ (the dependence on $\mathcal{A}$ will be omitted if clear from the context).

Observe that the choice made by a scheduler can depend on the (arbitrarily long) history of the execution. A scheduler $\sigma$ is *memoryless* if there exists a function $\sigma' : Q \to$

$\mathcal{D}(\mathcal{D}((\Sigma \times Q) \uplus \{\surd\}))$ such that $\sigma(\rho) = \sigma'(\mathrm{lst}(\rho))$. Hence a memoryless scheduler takes only into account the current state.

**Definition 7.3 (Scheduled NPA).** NPA $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0 \rangle$ scheduled by $\sigma$ is the (infinite) FPFA $\mathcal{A}_{/\sigma} = \langle \Sigma, Run(\mathcal{A}), \Delta', \varepsilon \rangle$ where

$$\Delta'(\rho)(a, \rho') = \sum_{\mu \in \Delta(q)} \sigma(\rho)(\mu) \cdot \mu(a, q') \quad \text{if } \rho' = \overbrace{q_0 \to \cdots \to q}^{\rho} \xrightarrow{a} q'$$

and $\Delta'(\rho)(a, \rho') = 0$ otherwise.

A scheduled NPA behaves as an FPFA, where the outgoing distribution is the set of all possible distributions weighted by the scheduler.

All measures defined in this paper on fully probabilistic finite automata can be extended to non-deterministic probabilistic automata. First note that all measures can be defined on infinite systems, although they cannot in general be computed automatically, even with proper restrictions on predicate and observables. From the security point of view, opacity in the case of an NPA should be the measure for the FPFA obtained with the worst possible scheduler. Hence the leak evaluated by the liberal measures (LPO and LPSO) is the greatest possible, and the robustness evaluated by the restrictive measures (RPO and RPSO) is the weakest possible.

**Definition 7.4.** Let $\mathcal{A}$ be an NPA, $\varphi$ a predicate, and $\mathcal{O}$ an observation function.

$$\text{For } \mathsf{PO} \in \{\mathsf{PO}^A_\ell, \mathsf{PO}^S_\ell\}, \quad \widehat{\mathsf{PO}}(\mathcal{A}, \varphi, \mathcal{O}) = \max_{\sigma \in Sched} \mathsf{PO}(\mathcal{A}_{/\sigma}, \varphi, \mathcal{O}).$$

$$\text{For } \mathsf{PO} \in \{\mathsf{PO}^A_r, \mathsf{PO}^S_r\}, \quad \widehat{\mathsf{PO}}(\mathcal{A}, \varphi, \mathcal{O}) = \min_{\sigma \in Sched} \mathsf{PO}(\mathcal{A}_{/\sigma}, \varphi, \mathcal{O}).$$

### 7.2. *The expressive power of schedulers*

In the context of analysis of security systems running in a hostile environment, it is quite natural to consider the scheduler to be under control of the adversary. However if not constrained this gives the adversary an unreasonably strong power even for obviously secure systems as it can reveal certain secrets. Also several classes of schedulers have been proposed in order to avoid considering unrealistic power of unconstrained schedulers and the ability of these classes to reach supremum probabilities (Giro and D'Argenio, 2009). We now investigate this problem for quantitative opacity.

First we show that memoryless schedulers are not sufficiently expressive, with the following counterexample.

**Theorem 7.5.** There exists an NPA $\mathcal{B}$ such that the value $\widehat{\mathsf{PO}^A_r}(\mathcal{B}, \varphi, \mathcal{O})$ cannot be reached by a memoryless scheduler.

*Proof.* Consider the NPA $\mathcal{B}$ of Figure 15. Transitions on $a$ and $b$ going to state $q_1$ (along with the westbound $\surd$) are part of the same probabilistic transition indicated by the arc linking the outgoing edges (and similarly eastbound). Let $\varphi$ be the (regular) predicate consisting of runs whose trace projected onto $\{a, b\}$ is in $(ab)^+ + (ab)^*a$ (so $a$

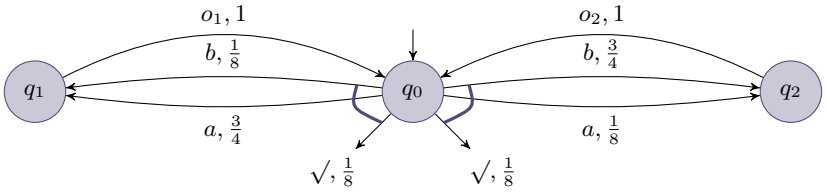
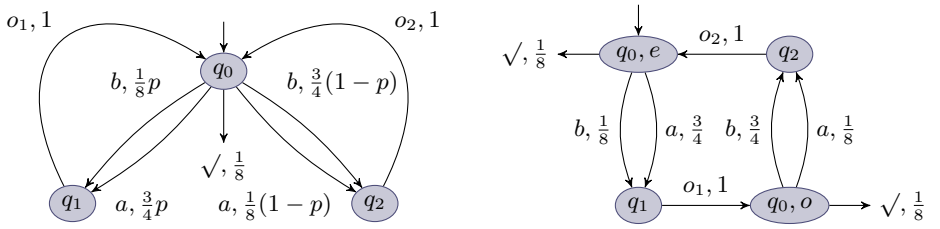



Figure 15. A nondeterministic probabilistic automaton $\mathcal{B}$.




(a) Fully probabilistic finite automaton $\mathcal{B}_{/\sigma_p}$ (b) Fully probabilistic finite automaton $\mathcal{B}_{/\sigma_m}$

Figure 16. Scheduled automata.

and $b$ must alternate). Let $\mathcal{O}$ be the observation function that keeps the last $o_i$ of the run. Hence there are only three observables, $\varepsilon$, $o_1$, and $o_2$. Intuitively, a scheduler can introduce a bias in the next letter read from state $q_0$.

First consider a memoryless scheduler $\sigma_p$. It can only choose once what weight will be affected to each transition. This choice is parametrized by probability $p$ that represents the weight of probability of the $q_1$ transition. The scheduled NPA $\mathcal{B}_{/\sigma_p}$ is depicted on Figure 16(a). The probabilities can be computed using the technique laid out in Section 5. We obtain the following probabilities (see Appendix C.1 for details):

$$\mathbf{P}(\varepsilon) = \frac{1}{8} \qquad \mathbf{P}(o_1) = \frac{7}{8} \cdot p \qquad \mathbf{P}(o_2) = \frac{7}{8} \cdot (1-p) \qquad \mathbf{P}(\varphi, \varepsilon) = 0$$

$$\mathbf{P}(\varphi, o_1) = \frac{p}{25p^2 - 25p + 58} \cdot \frac{5p + 49}{8} \qquad \mathbf{P}(\varphi, o_2) = \frac{1-p}{25p^2 - 25p + 58} \cdot \frac{15p + 7}{4}$$

Which yields

$$\frac{1}{\mathsf{PO}_r^A(\mathcal{B}_{/\sigma_p}, \varphi, \mathcal{O})} = \frac{1}{8} + \frac{49}{8} \cdot f(p) \cdot \left( \frac{p}{7f(p) - 5p - 49} + \frac{1-p}{7f(p) - 30p - 14} \right)$$

with $f(p) = 25p^2 - 25p + 58$ (see Appendix C.2). It can be shown[§] that regardless of $p$, $\mathsf{PO}_r^A(\mathcal{B}_{/\sigma_p}, \varphi, \mathcal{O})$ never falls below 0.88.

Now consider a scheduler $\sigma_m$ with memory who will try to maximize the realization of $\varphi$. In order to achieve that, it introduces a bias towards taking the letter which will fulfill $\varphi$: first an $a$, then a $b$, etc. Hence on the even positions, it will choose only transition

[§] With the help of tools such as WolframAlpha.

to $q_1$ (with probability 1) while it will choose the transition to $q_2$ on odd positions. The resulting FPFA is depicted on Figure 16(b). In this case, the probabilities of interest are:

$$\mathbf{P}(\varepsilon) = \frac{1}{8} \qquad \mathbf{P}(o_1) = \frac{7}{15} \qquad \mathbf{P}(o_2) = \frac{7}{8} \cdot \frac{7}{15}$$

$$\mathbf{P}(\varphi, \varepsilon) = 0 \qquad \mathbf{P}(\varphi, o_1) = \frac{3}{14} \qquad \mathbf{P}(\varphi, o_2) = \frac{3}{4} \cdot \frac{3}{14}$$

Probability $\mathbf{P}(o_1)$ can be obtained by noticing that the execution has to stop after an odd number of letters from $\{a, b\}$ have been read. The probability of stopping after exactly $n$ letters from $a$ or $b$ is $\frac{1}{8} \cdot \left(\frac{7}{8}\right)^n$. Therefore

$$\mathbf{P}(o_1) \;=\; \frac{1}{8} \cdot \sum_{i \geq 0} \left(\frac{7}{8}\right)^{2i+1} \;=\; \frac{1}{8} \cdot \frac{7}{8} \cdot \frac{1}{1 - \frac{49}{64}} \;=\; \frac{1}{8} \cdot \frac{7}{8} \cdot \frac{64}{15} \;=\; \frac{7}{15}.$$

Similar reasoning yield the other probabilities. The computation of RPO from these values (see Appendix C.3) gives $\mathsf{PO}_r^A(\mathcal{B}_{/\sigma_m}, \varphi, \mathcal{O}) = \frac{88192}{146509} \simeq 0.60$.

Hence a lower security is achieved by a scheduler provided it has (a finite amount of) memory. □

Note that this example used RPO, but a similar argument could be adapted for the other measures.

### 7.3. *Restricted schedulers*

What made a scheduler with memory more powerful than the one without in the counterexample of Section 7.2 was the knowledge of the truth value of $\varphi$ and exactly what was observed. More precisely, if the predicate and the observables are regular languages represented by finite deterministic and complete automata (FDCA), schedulers can be restricted to choices according to the current state of these automata and the state of the system. We conjecture that this knowledge is sufficient to any scheduler to compromise security at the best of its capabilities.

Let $\varphi \subseteq CRun(\mathcal{A})$ be a regular predicate represented by an FDCA $\mathcal{A}_\varphi$. Let $\mathcal{O} : CRun(\mathcal{A}) \to \{o_1, \ldots, o_n\}$ be an observation function such that for $1 \leq i \leq n$, $\mathcal{O}^{-1}(o_i)$ is a regular set represented by an FDCA $\mathcal{A}_{o_i}$. Consider the synchronized product $\mathcal{A}_{\varphi,\mathcal{O}} = \mathcal{A}_\varphi || \mathcal{A}_{o_1} || \ldots || \mathcal{A}_{o_n}$, which is also an FDCA, and denote by $Q_{\varphi,\mathcal{O}}$ its set of states. Let $\mathcal{A}_{\varphi,\mathcal{O}}(\rho)$ be the state of $\mathcal{A}_{\varphi,\mathcal{O}}$ reached after reading $\rho$.

**Definition 7.6 (Restricted $(\varphi, \mathcal{O})$-scheduler).** A scheduler $\sigma$ for $\mathcal{A}$ is said $(\varphi, \mathcal{O})$-restricted if there exists a function $\sigma' : (Q_{\varphi,\mathcal{O}} \times Q) \to \mathcal{D}(\mathcal{D}((\Sigma \times Q) \uplus \{\surd\}))$ such that for any run $\rho \in Run(\mathcal{A})$, $\sigma(\rho) = \sigma'(\mathcal{A}_{\varphi,\mathcal{O}}(\rho), \mathrm{lst}(\rho))$.

Remark that memoryless schedulers are always $(\varphi, \mathcal{O})$-restricted.

**Proposition 7.7.** If $\sigma$ is $(\varphi, \mathcal{O})$-restricted, then $\mathcal{A}_{/\sigma}$ is isomorphic to a finite FPFA.

These schedulers keep all information about the predicate and the observation. We conjecture that the relevant supremum is reached by a $(\varphi, \mathcal{O})$-restricted scheduler.

*Sketch of proof of Proposition 7.7* It can be shown that if $\sigma$ is $(\varphi, \mathcal{O})$-restricted, then:
(1) $\sigma$ is a memoryless scheduler for the product $\mathcal{A}||\mathcal{A}_{\varphi,\mathcal{O}}$
(2) and $(\mathcal{A}||\mathcal{A}_{\varphi,\mathcal{O}})_{/\sigma} = \mathcal{A}_{/\sigma}$. □

## 8. Conclusion

In this paper we introduced two dual notions of probabilistic opacity. The liberal one measures the probability for an attacker observing a random execution of the system to be able to gain information he can be sure about. The restrictive one measures the level of certitude in the information acquired by an attacker observing the system. The extremal cases of both these notions coincide with the possibilistic notion of opacity, which evaluates the existence of a leak of sure information. These notions yield measures that generalize either the case of asymmetrical or symmetrical opacity, thus providing four measures.

However, probabilistic opacity is not always easy to compute, especially if there are an infinite number of observables. Nevertheless, automatic computation is possible when dealing with regular predicates and finitely many regular observation classes. A prototype tool was implemented in Java, and can be used for numerical computation of opacity values.

In future work we plan to explore more of the properties of probabilistic opacity, to instantiate it to known security measures (anonymity, non-interference, etc.). Also, we want to extend the study of the non-deterministic case, by investigating the expressiveness of schedulers.

**Acknowledgments.** We wish to thank Catuscia Palamidessi for interesting discussions on this work. Thanks also to the reviewers for their helpful comments and to Adrian Eftenie for implementing the TPOT tool.

The authors are supported by projects ImpRo (ANR-2010-BLAN-0317) (French Government), CoChaT (DIGITEO-2009-27HD) (Région Île de France), inVEST (ERC Starting Grant 279499), the NSERC Discovery Individual grant No. 13321 (Government of Canada), and by the FQRNT Team grant No. 167440 (Quebec's Government).

## References

Aldini, A. and Bernardo, M. (2009). A General Framework for Nondeterministic, Probabilistic, and Stochastic Noninterference. In Degano, P. and Viganò, L., editors, *Proc. of Foundations and Applications of Security Analysis; ARSPA-WITS 2009*, volume 5511 of *Lecture Notes in Computer Science*, pages 191–245. Springer.

Aldini, A., Bravetti, M., and Gorrieri, R. (2004). A process-algebraic approach for the analysis of probabilistic noninterference. *Journal of Computer Security*, 12(2):191–245.

Alur, R., Černý, P., and Zdancewic, S. (2006). Preserving secrecy under refinement. In *Proc. of the 33rd Intl. Colloquium on Automata, Languages and Programming (ICALP'06)*, volume 4052 of *LNCS*, pages 107–118. Springer.

Alvim, M. S., Andrés, M. E., and Palamidessi, C. (2010). Information flow in interactive systems. In Gastin, P. and Laroussinie, F., editors, *Proceedings of the 21th International Conference on Concurrency Theory (CONCUR'10)*, volume 6269 of *LNCS*, pages 102–116. Springer.

Andrés, M. E., Palamidessi, C., van Rossum, P., and Smith, G. (2010). Computing the leakage of information-hiding systems. In *Proc. 16th Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'10)*, volume 6015 of *LNCS*, pages 373–389. Springer.

Bérard, B., Mullins, J., and Sassolas, M. (2010). Quantifying opacity. In Ciardo, G. and Segala, R., editors, *Proceedings of the 7th International Conference on Quantitative Evaluation of Systems (QEST'10)*, pages 263–272. IEEE Computer Society.

Bianco, A. and de Alfaro, L. (1995). Model checking of probabilistic and nondeterministic systems. In *Proc. 15th Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'95)*, volume 1026 of *LNCS*, pages 499–513. Springer.

Boreale, M. (2009). Quantifying information leakage in process calculi. *Information and Computation*, 207(6):699–725.

Boreale, M., Clark, D., and Gorla, D. (2010). A semiring-based trace semantics for processes with applications to information leakage analysis. In Calude, C. S. and Sassone, V., editors, *Proc. of 6th IFIP TC 1/WG 2.2 International Conference, TCS 2010, Held as Part of WCC 2010 (IFIP TCS)*, volume 323 of *IFIP*, pages 340–354. Springer.

Boreale, M., Pampaloni, F., and Paolini, M. (2011a). Asymptotic Information Leakage under One-Try Attacks. In Hofmann, M., editor, *Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011*, volume 6604 of *Lecture Notes in Computer Science*, pages 396–410. Springer.

Boreale, M., Pampaloni, F., and Paolini, M. (2011b). Quantitative information flow, with a view. In Atluri, V. and Díaz, C., editors, *Proc. of 16th European Symposium on Research in Computer Security (ESORICS 2011)*, volume 6879 of *Lecture Notes in Computer Science*, pages 588–606. Springer.

Bryans, J. W., Koutny, M., Mazaré, L., and Ryan, P. Y. A. (2008). Opacity generalised to transition systems. *Intl. Jour. of Information Security*, 7(6):421–435.

Chatzikokolakis, K., Palamidessi, C., and Panangaden, P. (2008). Anonymity protocols as noisy channels. *Information and Computation*, 206(2-4):378–401.

Chaum, D. (1988). The dining cryptographers problem: unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75.

Courcoubetis, C. and Yannakakis, M. (1998). Markov Decision Processes and Regular Events. *IEEE Transactions on Automatc Control*, 43(10):1399–1418.

Dubreil, J., Darondeau, P., and Marchand, H. (2010). Supervisory Control for Opacity. *IEEE Transactions on Automatic Control*, 55(5):1089 –1100.

Eftenie, A. (2010). Computing quantitative opacity with TPOT – `http://www.info.polymtl.ca/~adeft/computing_opacity/`.

Giro, S. and D'Argenio, P. R. (2009). On the expressive power of schedulers in distributed probabilistic systems. *Electronic Notes in Theoretical Computer Science*, 253(3):45–71.

Goguen, J. A. and Meseguer, J. (1982). Security policy and security models. In *Proc. of IEEE Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press.

Hansson, H. and Jonsson, B. (1994). A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535.

Kemeny, J. G. and Snell, J. L. (1976). *Finite Markov Chains*. Undergraduate Texts in Mathemetics. Sringer-Verlag.

Lakhnech, Y. and Mazaré, L. (2005). Probabilistic opacity for a passive adversary and its application to Chaum's voting scheme. Technical Report 4, Verimag.

Lin, F. (2011). Opacity of discrete event systems and its applications. *Automatica*, 47(3):496–503.

Lowe, G. (2004). Defining information flow quantity. *Journal of Computer Security*, 12(3-4):619–653.

Mantel, H. and Sudbrock, H. (2009). Information-theoretic modeling and analysis of interrupt-related covert channels. In Degano, P., Guttman, J., and Martinelli, F., editors, *Proceedings of the Workshop on Formal Aspects in Security and Trust, FAST 2008*, Springer, LNCS 5491, pages 67–81.

Mazaré, L. (2005). Decidability of opacity with non-atomic keys. In *Proc. 2nd Workshop on Formal Aspects in Security and Trust (FAST'04)*, volume 173 of *Intl. Federation for Information Processing*, pages 71–84. Springer.

McIver, A., Meinicke, L., and Morgan, C. (2010). Compositional closure for bayes risk in probabilistic noninterference. In Abramsky, S., Gavoille, C., Kirchner, C., auf der Heide, F. M., and Spirakis, P. G., editors, *Proc. of 37th International Colloquium on Automata, Languages and Programming (ICALP'10), Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 223–235. Springer.

Millen, J. K. (1987). Covert Channel Capacity. In *Proc. of IEEE Symposium on Research in Computer Security and Privacy*, pages 144–161.

Reiter, M. K. and Rubin, A. D. (1998). Crowds: anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92.

Segala, R. (1995). *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, Department of Electrical Engineering and Computer Science.

Smith, G. (2009). On the foundations of quantitative information flow. In *Proc. 12th Intl. Conf. on Foundations of Software Science and Computational Structures (FOSSACS'09)*, pages 288–302. Springer-Verlag.

Wittbold, J. T. and Johnson, D. M. (1990). Information flow in nondeterministic systems. In *Proc. of IEEE Symposium on Research in Computer Security and Privacy*, pages 144–161.

## Appendix A. Computation of RPO for the debit card system

We give here the details of the computation of RPO in the example of the debit cards system of Section 4.1.

$$\begin{aligned}
\mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon) &= \mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon, x > 1000) \;+\; \mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon, 500 < x \leq 1000) \\
&\quad + \mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon, 100 < x \leq 500) \;+\; \mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon, x \leq 100) \\
&= \mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon) \cdot \mathbf{P}(x > 1000) \\
&\quad + \mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon) \cdot \mathbf{P}(500 < x \leq 1000) \\
&\quad + \mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon) \cdot \mathbf{P}(100 < x \leq 500) \\
&\quad + \mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon) \cdot \mathbf{P}(x \leq 100) \\
&= 0.05 \cdot 0.05 + 0.25 \cdot 0.2 + 0.5 \cdot 0.45 + 0.8 \cdot 0.3 \\
\mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon) &= 0.5175
\end{aligned}$$

$$\mathbf{P}(\mathcal{O}_{\text{Call}} = \text{Call}) \;=\; 1 - \mathbf{P}(\mathcal{O}_{\text{Call}} = \varepsilon) \;=\; 0.4825$$

$$\begin{aligned}
\mathbf{P}(\neg\varphi_{>500}|\mathcal{O}_{\text{Call}}=\varepsilon) &= \frac{\mathbf{P}(\neg\varphi_{>500},\mathcal{O}_{\text{Call}}=\varepsilon)}{\mathbf{P}(\mathcal{O}_{\text{Call}}=\varepsilon)}\\
&= \frac{\mathbf{P}(x\leq 100,\mathcal{O}_{\text{Call}}=\varepsilon)\ +\ \mathbf{P}(100<x\leq 500,\mathcal{O}_{\text{Call}}=\varepsilon)}{\mathbf{P}(\mathcal{O}_{\text{Call}}=\varepsilon)}\\
&= \frac{\mathbf{P}(\mathcal{O}_{\text{Call}}=\varepsilon|x\leq 100)\cdot\mathbf{P}(x\leq 100)}{\mathbf{P}(\mathcal{O}_{\text{Call}}=\varepsilon)}\\
&\quad + \frac{\mathbf{P}(\mathcal{O}_{\text{Call}}=\varepsilon|100<x\leq 500)\cdot\mathbf{P}(100<x\leq 500)}{\mathbf{P}(\mathcal{O}_{\text{Call}}=\varepsilon)}\\
&= \frac{0.8\cdot 0.3+0.5\cdot 0.45}{0.5175}\\
\mathbf{P}(\neg\varphi_{>500}|\mathcal{O}_{\text{Call}}=\varepsilon) &= \frac{0.465}{0.5175}\ \simeq\ 0.899
\end{aligned}$$

$$\begin{aligned}
\mathbf{P}(\neg\varphi_{>500}|\mathcal{O}_{\text{Call}}=\text{Call}) &= \frac{\mathbf{P}(\neg\varphi_{>500},\mathcal{O}_{\text{Call}}=\text{Call})}{\mathbf{P}(\mathcal{O}_{\text{Call}}=\text{Call})}\\
&= \frac{\mathbf{P}(x\leq 100,\mathcal{O}_{\text{Call}}=\text{Call})\ +\ \mathbf{P}(100<x\leq 500,\mathcal{O}_{\text{Call}}=\text{Call})}{\mathbf{P}(\mathcal{O}_{\text{Call}}=\text{Call})}\\
&= \frac{\mathbf{P}(\mathcal{O}_{\text{Call}}=\text{Call}|x\leq 100)\cdot\mathbf{P}(x\leq 100)}{\mathbf{P}(\mathcal{O}_{\text{Call}}=\text{Call})}\\
&\quad + \frac{\mathbf{P}(\mathcal{O}_{\text{Call}}=\text{Call}|100<x\leq 500)\cdot\mathbf{P}(100<x\leq 500)}{\mathbf{P}(\mathcal{O}_{\text{Call}}=\text{Call})}\\
&= \frac{0.2\cdot 0.3+0.5\cdot 0.45}{0.4825}\\
\mathbf{P}(\neg\varphi_{>500}|\mathcal{O}_{\text{Call}}=\text{Call}) &= \frac{0.285}{0.4825}\ \simeq\ 0.591
\end{aligned}$$

$$\begin{aligned}
\frac{1}{\mathsf{PO}_r^A(\mathcal{A}_{\text{card}},\varphi_{>500},\mathcal{O}_{\text{Call}})} &= 0.5175\cdot\frac{0.5175}{0.465}+0.4825\cdot\frac{0.4825}{0.285}\\
\frac{1}{\mathsf{PO}_r^A(\mathcal{A}_{\text{card}},\varphi_{>500},\mathcal{O}_{\text{Call}})} &= \frac{39377}{28272}\ \simeq\ 1.393
\end{aligned}$$

The last line was obtained by reducing the one above with the help of the formal computation tool *WolframAlpha*.

## Appendix B. Resolution of the linear system for Crowds protocol

It can be seen in the system of Table 6 (page 27) that $L_1 = L_2 = \cdots = L_{n-c-1} = q\cdot L_{1'}$ and $L_{n-c+1} = \cdots = L_n = L_S = 1$. Therefore, it suffices to eliminate $L_{1'}$ and compute $L_0$, $L_1$ and $L_{n-c}$.

$$\left\{\begin{array}{rcl} L_0 &=& \frac{1}{q(n-c)}\cdot L_1\\ L_1 &=& q\left(\frac{n-c-1}{n}\cdot L_1+\frac{1}{n}\cdot L_{n-c}\right)\\ L_{n-c} &=& 1-\frac{q(n-c)}{n}+L_1\end{array}\right.$$

The line for $L_{n-c}$ is obtained as follows:

$$\begin{aligned}
L_{n-c} &= (1-q)\cdot L_S + \sum_{i=1}^{n} \frac{q}{n}\cdot L_i \\
L_{n-c} &= (1-q)\cdot L_S + \sum_{i=1}^{n-c} \frac{q}{n}\cdot L_i + \sum_{i=n-c+1}^{n} \frac{q}{n}\cdot L_i \\
L_{n-c} &= 1-q+L_1+\frac{q\cdot c}{n} \\
L_{n-c} &= 1-\frac{q(n-c)}{n}+L_1
\end{aligned}$$

This yields, for $L_1$:

$$\begin{aligned}
L_1 &= \frac{q}{n}(n-c)\cdot L_1 + \frac{q}{n}\left(1-\frac{q(n-c)}{n}\right) \\
\frac{n}{q}\cdot L_1 &= (n-c)\cdot L_1 + 1 - \frac{q(n-c)}{n} \\
L_1\left(\frac{n}{q}-(n-c)\right) &= \frac{n-q(n-c)}{n} \\
L_1 &= \frac{q}{n}
\end{aligned}$$

The other values are easily deduced from $L_1$.

## Appendix C. Calculations in the proof of Theorem 7.5

C.1. *Probabilities in $\mathcal{B}_{/\sigma_p}$.*

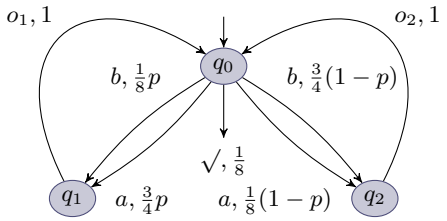
We compute the probabilities of several events in the automaton $\mathcal{B}_{/\sigma_p}$, reproduced on Figure 17(a). Recall that $\varphi = (ab)^+ + (ab)^*a$ and $\mathcal{O}$ is the last letter read, so the set of observables is $Obs = \{\varepsilon, o_1, o_2\}$.

The computation of the probability $\mathbf{P}(\varphi, o_1)$ in FPFA $\mathcal{B}_{/\sigma_p}$ goes as follows. We write $\overline{p} = 1-p$ for brevity. First we build the synchronized product $\mathcal{B}_{/\sigma_p}||\mathcal{A}_\varphi||\mathcal{A}_{o_1}$, as depicted in Figure 18. The linear system of Table 7 is built from this automaton. This system can be trimmed down in order to remove redundancy, and since only the value of $x_{000} = \mathbf{P}(\varphi, o_1)$ is of interest:

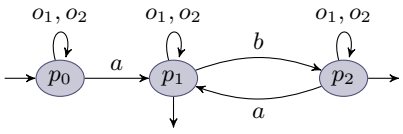
$$\left\{\begin{array}{lcl}
x_{000} &=& \frac{1}{8}\overline{p}\ x_{010} + \frac{3}{4}p\ x_{011} \\
x_{010} &=& \frac{1}{8}p\ x_{021} + \frac{3}{4}\overline{p}\ x_{020} \\
x_{011} &=& \frac{1}{8} + \frac{3}{4}\overline{p}\ x_{020} + \frac{1}{8}p\ x_{021} \\
x_{020} &=& \frac{1}{8}\overline{p}\ x_{010} + \frac{3}{4}p\ x_{011} \\
x_{021} &=& \frac{1}{8} + \frac{1}{8}\overline{p}\ x_{010} + \frac{3}{4}p\ x_{011}
\end{array}\right. \iff \left\{\begin{array}{lcl}
x_{000} &=& \frac{1}{8}\overline{p}\ x_{010} + \frac{3}{4}p\ x_{011} \\
x_{010} &=& \frac{1}{8}p\ x_{021} + \frac{3}{4}\overline{p}\ x_{020} \\
x_{011} &=& \frac{1}{8} + x_{010} \\
x_{020} &=& x_{000} \\
x_{021} &=& \frac{1}{8} + x_{000}
\end{array}\right.$$

We therefore obtain:

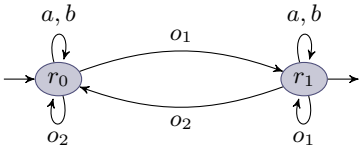
$$\left\{\begin{array}{lcl}
x_{000} &=& \frac{1}{8}\overline{p}\ x_{010} + \frac{3}{4}p\left(\frac{1}{8} + x_{010}\right) \\
x_{010} &=& \frac{1}{8}p\left(\frac{1}{8} + x_{000}\right) + \frac{3}{4}\overline{p}\ x_{000}
\end{array}\right. \text{ So } \left\{\begin{array}{lcl}
x_{000} &=& \frac{1}{8}\overline{p}\ x_{010} + \frac{3}{4}p\left(\frac{1}{8} + x_{010}\right) \\
x_{010} &=& \frac{1}{64}p + \frac{1}{8}p\ x_{000} + \frac{3}{4}\overline{p}\ x_{000}
\end{array}\right.$$



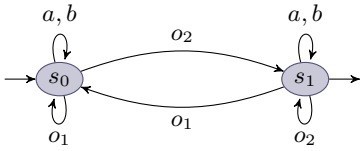

(a) Fully probabilistic finite automaton $\mathcal{B}_{/\sigma_p}$




(b) Deterministic finite automaton $\mathcal{A}_\varphi$




(c) Deterministic finite automaton $\mathcal{A}_{o_1}$




(d) Deterministic finite automaton $\mathcal{A}_{o_2}$

Figure 17. Automata for the computation of $\mathbf{P}(\varphi, o_1)$ and $\mathbf{P}(\varphi, o_2)$.

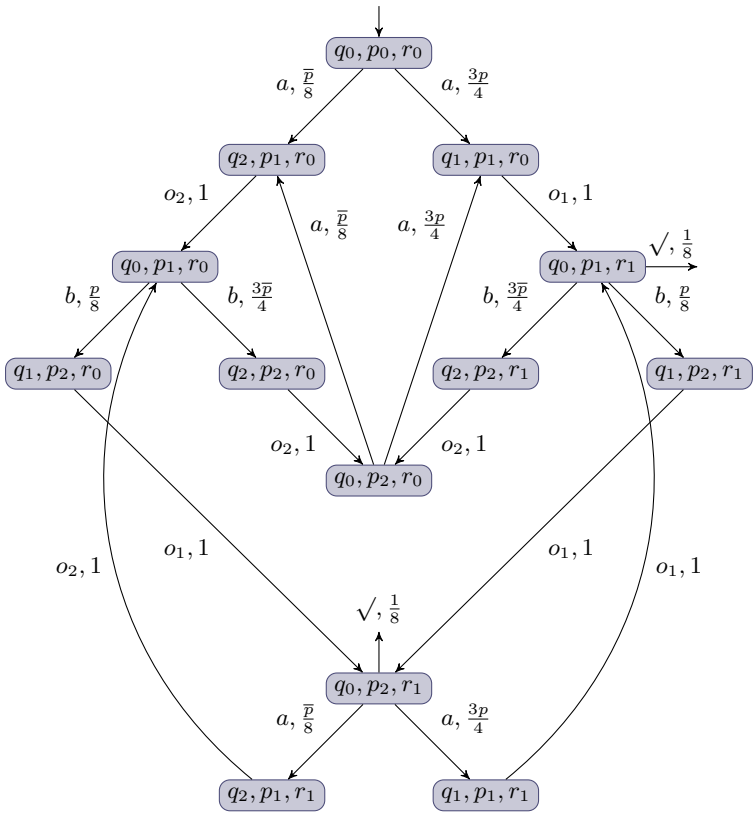



Figure 18. Substochastic Automaton $\mathcal{B}_{/\sigma_p} || \mathcal{A}_\varphi || \mathcal{A}_{o_1}$.

$$\left\{\begin{array}{lcl} x_{000} & = & \frac{1}{8}\overline{p}\ x_{210} + \frac{3}{4}p\ x_{110} \\ x_{210} & = & x_{010} \\ x_{110} & = & x_{011} \\ x_{010} & = & \frac{1}{8}p\ x_{120} + \frac{3}{4}\overline{p}\ x_{220} \\ x_{011} & = & \frac{1}{8} + \frac{3}{4}\overline{p}\ x_{221} + \frac{1}{8}p\ x_{121} \\ x_{120} & = & x_{021} \\ x_{220} & = & x_{020} \\ x_{221} & = & x_{020} \\ x_{121} & = & x_{021} \\ x_{020} & = & \frac{1}{8}\overline{p}\ x_{210} + \frac{3}{4}p\ x_{110} \\ x_{021} & = & \frac{1}{8} + \frac{1}{8}\overline{p}\ x_{211} + \frac{3}{4}p\ x_{111} \\ x_{211} & = & x_{010} \\ x_{111} & = & x_{011} \end{array}\right.$$

Table 7. Linear system associated to the SA $\mathcal{B}_{/\sigma_p}||\mathcal{A}_{\varphi}||\mathcal{A}_{o_1}$. The variables names indicate the corresponding state in the automaton; for example $x_{210}$ corresponds to state $(q_2, p_1, r_0)$.

As a result

$$x_{000} \ = \ \frac{1}{8}\overline{p}\ \left(\frac{1}{64}p + \frac{1}{8}p\ x_{000} + \frac{3}{4}\overline{p}\ x_{000}\right) + \frac{3}{4}p\left(\frac{1}{8} + \frac{1}{64}p + \frac{1}{8}p\ x_{000} + \frac{3}{4}\overline{p}\ x_{000}\right)$$

In the sequel, we replace $x_{000}$ with $x$ for readability's sake.

$$\begin{array}{rcl} x & = & \frac{1}{8}\overline{p}\ \left(\frac{1}{64}p + \frac{1}{8}p\ x + \frac{3}{4}\overline{p}\ x\right) + \frac{3}{4}p\left(\frac{1}{8} + \frac{1}{64}p + \frac{1}{8}p\ x + \frac{3}{4}\overline{p}\ x\right) \\ \iff x & = & \frac{1}{512}p\overline{p} + \frac{1}{64}p\overline{p}x + \frac{3}{32}\overline{p}^2x + \frac{3}{32}p + \frac{3}{256}p^2 + \frac{3}{32}p^2x + \frac{9}{16}p\overline{p}x \\ \iff x\left(1 - \frac{1}{64}p\overline{p} - \frac{3}{32}\overline{p}^2 - \frac{3}{32}p^2 - \frac{9}{16}p\overline{p}\right) & = & \frac{1}{512}p\overline{p} + \frac{3}{32}p + \frac{3}{256}p^2 \\ \iff x & = & \frac{\frac{1}{8}p\overline{p} + 6p + \frac{3}{4}p^2}{64 - p\overline{p} - 6\overline{p}^2 - 6p^2 - 36p\overline{p}} \\ \iff x & = & \frac{\frac{1}{8}p(1-p) + 6p + \frac{3}{4}p^2}{64 - 37p(1-p) - 6(p^2 - 2p + 1) - 6p^2} \\ \iff x & = & \frac{\frac{1}{8}p - \frac{1}{8}p^2 + 6p + \frac{3}{4}p^2}{64 - 37p + 37p^2 - 6p^2 + 12p - 6 - 6p^2} \\ \iff x & = & \frac{1}{8}\cdot p \cdot \frac{5p + 49}{25p^2 - 25p + 58} \end{array}$$

The same technique can be applied to the computation of $\mathbf{P}(\varphi, o_2)$ in $\mathcal{B}_{/\sigma_p}$. The product is depicted on Figure 19, and the linear system obtained boils down to

$$\left\{\begin{array}{lcl} x_{000} & = & \frac{1}{8}\overline{p}\left(\frac{1}{8} + x_{010}\right) + \frac{3}{4}p\ x_{010} \\ x_{010} & = & \frac{1}{8}p\ x_{000} + \frac{3}{4}\overline{p}\left(\frac{1}{8} + x_{000}\right) \end{array}\right.$$

As before, $x_{000}$ is replaced by $x$ for readability; we solve:

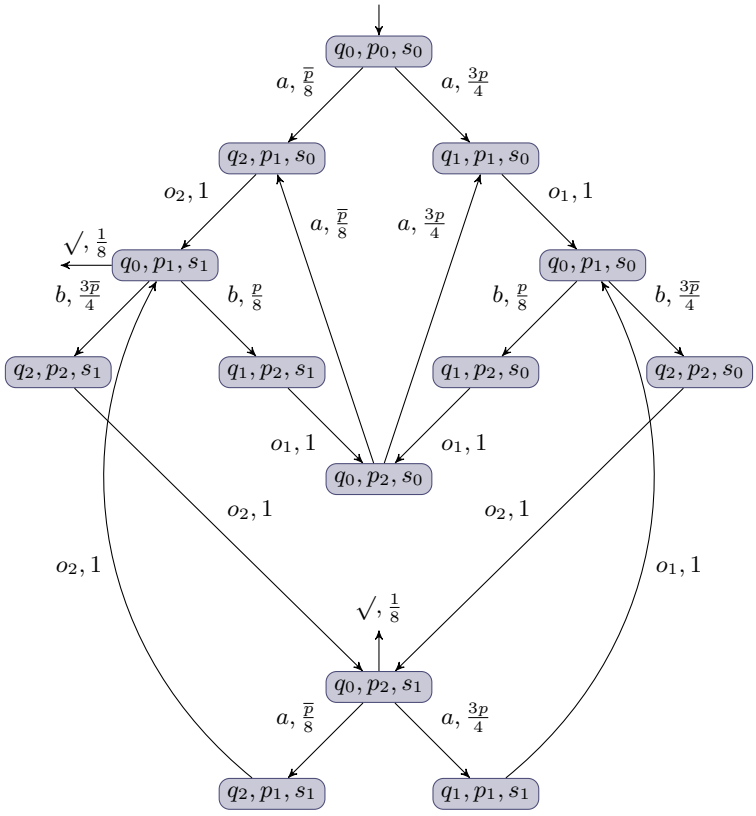



Figure 19. Substochastic Automaton $\mathcal{B}_{/\sigma_p}||\mathcal{A}_{\varphi}||\mathcal{A}_{o_2}$.

$$
\begin{aligned}
x &= \frac{1}{8}\overline{p}\left(\frac{1}{8}+\frac{1}{8}px+\frac{3}{4}\overline{p}\left(\frac{1}{8}+x\right)\right)+\frac{3}{4}p\ \left(\frac{1}{8}px+\frac{3}{4}\overline{p}\left(\frac{1}{8}+x\right)\right)\\
x &= \frac{1}{64}\overline{p}+\frac{1}{64}p\overline{p}x+\frac{3}{256}\overline{p}^2+\frac{3}{32}\overline{p}^2x+\frac{3}{32}p^2x+\frac{9}{128}p\overline{p}+\frac{9}{16}p\overline{p}x\\
x &= \frac{\frac{1}{64}\overline{p}+\frac{3}{256}\overline{p}^2+\frac{9}{128}p\overline{p}}{1-\frac{1}{64}p\overline{p}-\frac{3}{32}\overline{p}^2-\frac{3}{32}p^2-\frac{9}{16}p\overline{p}}\\
x &= \frac{4\overline{p}+3\overline{p}^2+18p\overline{p}}{256-24\overline{p}^2-24p^2-148p\overline{p}}\\
x &= \frac{(1-p)(4+3-3p+18p)}{256-24p^2+48p-24-24p^2-148p+148p^2}\\
x &= \frac{(1-p)(15p+7)}{232-100p+100p^2}\\
x &= \frac{(1-p)(15p+7)}{4(25p^2-25p+58)}
\end{aligned}
$$

C.2. *Computation of RPO for* $\mathcal{B}_{/\sigma_p}$

In the sequel, we write $f(p) = 25p^2 - 25p + 58$. We have $\mathbf{P}(\overline{\varphi}|\varepsilon) = 1$ and

$$\begin{aligned}
\mathbf{P}(\overline{\varphi}|o_1) &= 1 - \frac{\mathbf{P}(\varphi, o_1)}{\mathbf{P}(o_1)} \\
\mathbf{P}(\overline{\varphi}|o_1) &= 1 - \frac{1}{8} \cdot p \cdot \frac{5p+49}{25p^2-25p+58} \cdot \frac{8}{7p} \\
\mathbf{P}(\overline{\varphi}|o_1) &= \frac{7f(p) - 5p - 49}{7f(p)}
\end{aligned}$$

$$\begin{aligned}
\mathbf{P}(\overline{\varphi}|o_2) &= 1 - \frac{\mathbf{P}(\varphi, o_2)}{\mathbf{P}(o_2)} \\
\mathbf{P}(\overline{\varphi}|o_2) &= 1 - \frac{(1-p)(15p+7)}{4(25p^2-25p+58)} \cdot \frac{8}{7(1-p)} \\
\mathbf{P}(\overline{\varphi}|o_2) &= \frac{7f(p) - 30p - 14}{7f(p)}
\end{aligned}$$

$$\begin{aligned}
\frac{1}{\mathsf{PO}_r^A\left(\mathcal{B}_{/\sigma_p}, \varphi, \mathcal{O}\right)} &= \mathbf{P}(\varepsilon) + \mathbf{P}(o_1) \cdot \frac{1}{\mathbf{P}(\overline{\varphi}|o_1)} + \mathbf{P}(o_2) \cdot \frac{1}{\mathbf{P}(\overline{\varphi}|o_2)} \\
\frac{1}{\mathsf{PO}_r^A\left(\mathcal{B}_{/\sigma_p}, \varphi, \mathcal{O}\right)} &= \frac{1}{8} + \frac{7}{8} \cdot p \cdot \frac{7f(p)}{7f(p) - 5p - 49} + \frac{7}{8} \cdot (1-p) \cdot \frac{7f(p)}{7f(p) - 30p - 14} \\
\frac{1}{\mathsf{PO}_r^A\left(\mathcal{B}_{/\sigma_p}, \varphi, \mathcal{O}\right)} &= \frac{1}{8} + \frac{49f(p)}{8}\left(\frac{p}{7f(p) - 5p - 49} + \frac{1-p}{7f(p) - 30p - 14}\right)
\end{aligned}$$

C.3. *Computation of RPO for* $\mathcal{B}_{/\sigma_m}$

We have:

$$\mathbf{P}(\overline{\varphi}|\varepsilon) = 1 \qquad \mathbf{P}(\overline{\varphi}|o_1) = 1 - \frac{3}{14} \cdot \frac{15}{7} = \frac{53}{98} \qquad \mathbf{P}(\overline{\varphi}|o_2) = 1 - \frac{3}{4} \cdot \frac{3}{14} \cdot \frac{15}{7} \cdot \frac{8}{7} = \frac{208}{343}$$

Therefore:

$$\begin{aligned}
\frac{1}{\mathsf{PO}_r^A(\mathcal{B}_{/\sigma_m}, \varphi, \mathcal{O})} &= \frac{1}{8} + \frac{7}{15} \cdot \frac{98}{53} + \frac{7}{8} \cdot \frac{7}{15} \cdot \frac{343}{208} \\
\frac{1}{\mathsf{PO}_r^A(\mathcal{B}_{/\sigma_m}, \varphi, \mathcal{O})} &= \frac{146509}{88192} \\
\mathsf{PO}_r^A(\mathcal{B}_{/\sigma_m}, \varphi, \mathcal{O}) &= \frac{88192}{146509} \\
\mathsf{PO}_r^A(\mathcal{B}_{/\sigma_m}, \varphi, \mathcal{O}) &\simeq 0.60
\end{aligned}$$

# Timed Automata Can Always Be Made Implementable*

Patricia Bouyer[1], Kim G. Larsen[2], Nicolas Markey[1], Ocan Sankur[1], and Claus Thrane[2]

[1] LSV, CNRS & ENS Cachan, France.
{bouyer,markey,sankur}@lsv.ens-cachan.fr

[2] Dept. Computer Science, Aalborg University, Denmark.
{kgl,crt}@cs.aau.dk

**Abstract.** Timed automata follow a mathematical semantics, which assumes perfect precision and synchrony of clocks. Since this hypothesis does not hold in digital systems, properties proven formally on a timed automaton may be lost at implementation. In order to ensure implementability, several approaches have been considered, corresponding to different hypotheses on the implementation platform. We address two of these: A timed automaton is samplable if its semantics is preserved under a discretization of time; it is robust if its semantics is preserved when all timing constraints are relaxed by some small positive parameter.
We propose a construction which makes timed automata implementable in the above sense: From any timed automaton $\mathcal{A}$, we build a timed automaton $\mathcal{A}'$ that exhibits the same behaviour as $\mathcal{A}$, and moreover $\mathcal{A}'$ is both robust and samplable by construction.

## 1 Introduction

Timed automata [3] extend finite-state automata with real-valued variables which measure delays between actions. They provide a powerful yet natural way of modelling real-time systems. They also enjoy decidability of several important problems, which makes them a model of choice for the verification of real-time systems. This has been witnessed over the last twenty years by substantial effort from the verification community to equip timed automata with efficient tool support, which was accompanied by successful applications.

However, timed automata are governed by a mathematical semantics, which assumes continuous and infinitely precise measurement of time, while hardware is digital and imprecise. Hence properties proven at the formal level might be lost when implementing the abstract model of the automaton as a digital circuit or as a program on a physical CPU. Several approaches have been proposed to overcome this discrepancy, with different hypotheses on the implementation

* This work has been partly supported by EU FP7 project Quasimodo (ICT-214755), and by French ANR projects DOTS (ANR-06-SETI-003) and ImpRo (ANR-10-BLAN-0317).

platform (*e.g.* [4, 15, 20, 12, 5, 21]). In this work, we address two such approaches, namely, the sampled semantics and the robustness, which we now detail.

Sampled semantics for timed automata, where all time delays are integer multiples of a rational sampling rate, have been studied in order to capture, for example the behaviour of digital circuits (*e.g.* [4, 8]). In fact, only such instants are observable in a digital circuit, under the timing of a quartz clock. However, for some timed automata, any sampling rate may disable some (possibly required) behaviour [9]. Consequently, a natural problem which has been studied is that of choosing a sampling rate under which a property is satisfied. For safety properties, this problem is undecidable for timed automata [9]; but it becomes decidable for reachability under a slightly different setting [17]. Recently, [1] showed the decidability of the existence of a sampling rate under which the continuous and the sampled semantics recognize the same untimed language.

A prominent approach, originating from [20, 12], for verifying the behavior of real-time programs executed on CPUs, is robust model-checking. It consists in studying the *enlarged semantics* of the timed automaton, where all the constraints are enlarged by a small (positive) perturbation $\Delta$, in order to model the imprecisions of the clock. In some cases [11], this may allow new behaviours in the system, regardless of $\Delta$ (See Fig. 2 on page 8). Such automata are said to be not *robust* to small perturbations. On the other hand, if no new behaviour is added to the system, that is, if the system is robust, then *implementability* on a fast-enough CPU will be ensured [12]. Since its introduction, robust model-checking has been solved for safety properties [20, 11], and for richer linear-time properties [6, 7]. See also [21] for a variant of the implementation model of [12] and a new approach to obtain implementations.

In this paper, we show that timed automata can always be made implementable in both senses. More precisely, given a timed automaton $\mathcal{A}$, we build another timed automaton $\mathcal{B}$ whose semantics under enlargement and under sampling is bisimilar to $\mathcal{A}$. We use a quantitative variant of bisimulation from [14] where the differences between the timings in two systems are bounded above by a parameter $\varepsilon$ (see also [16] for a similar quantitative notion of bisimulation). Our construction is parameterized and provides a bisimilar implementation for any desired precision $\varepsilon > 0$. Moreover, we prove that in timed automata, this notion of bisimulation preserves, up to an error of $\varepsilon$, all properties expressed in a quantitative extension of $\mathsf{CTL}$, also studied in [14].

## 2 Timed Models and Specifications

### 2.1 Timed Transition Systems and Behavioural Relations

A *timed transition system (TTS)* is a tuple $\mathcal{S} = (S, s_0, \Sigma, \mathbb{K}, \to)$, where $S$ is the set of *states*, $s_0 \in S$ the initial state, $\Sigma$ a finite alphabet, $\mathbb{K} \subseteq \mathbb{R}_{\geq 0}$ the time domain which contains 0 and is closed under addition, and $\to \subseteq S \times (\Sigma \cup \mathbb{K}) \times S$ the *transition* relation. We write $s \xrightarrow{\sigma} s'$ instead of $(s, \sigma, s') \in \to$; we also write $s \xrightarrow{d,\sigma} s'$ if $s \xrightarrow{d} s'' \xrightarrow{\sigma} s'$ for $d \in \mathbb{K}$, $\sigma \in \Sigma$ and some state $s''$, and $s \stackrel{\sigma}{\Longrightarrow} s'$

if $s \xrightarrow{d',\sigma} s'$ for some $d' \in \mathbb{K}$. A *run* $\rho$ of $\mathcal{S}$ is a finite or infinite sequence $q_0 \xrightarrow{\tau_0} q_0' \xrightarrow{\sigma_0} q_1 \xrightarrow{\tau_1} q_1' \xrightarrow{\sigma_1} \ldots$, where $q_i \in S$, $\sigma_i \in \Sigma$ and $\tau_i \in \mathbb{K}$ for all $i$. The word $\sigma_0\sigma_1 \ldots \in \Sigma^*$ is the *trace* of $\rho$. We denote by $\mathsf{Trace}(\mathcal{S})$ the set of finite and infinite traces of the runs of $\mathcal{S}$. We define the set of *reachable states* of $\mathcal{S}$, denoted by $\mathsf{Reach}(\mathcal{S})$, as the set of states $s'$ for which some finite run of $\mathcal{S}$ starts from state $s_0$ and ends in state $s'$. A run written on the form $\gamma = q_0 \xrightarrow{d_0,\sigma_0} q_1 \xrightarrow{d_1,\sigma_1} q_2 \ldots$ is a *timed-action path* (or simply path). Each state $q_0 \in S$ admits a set $P(q_0)$ of paths starting at $q_0$. For any path $\gamma$, the suffix $\gamma^j$ is obtained by deleting the first $j$ transitions in $\gamma$, and $\gamma(j) = q_j \xrightarrow{d_j,\sigma_j} q_{j+1}$ is the $j$-th transition in $\gamma$; we also let $\mathsf{state}_j(\gamma) = q_j$, $\gamma(j)_\sigma = \sigma_j$, and $\gamma(j)_d = d_j$.

We consider a quantitative extension of timed bisimilarity introduced in [22]. This spans the gap between timed and time-abstract bisimulations: while the former requires time delays to be matched exactly, the latter ignores timing information altogether. Intuitively, we define two states to be $\varepsilon$-bisimilar, for a given parameter $\varepsilon \geq 0$, if there is a (time-abstract) bisimulation which relates these states in such a way that, at each step, the difference between the time delays of corresponding delay transitions is at most $\varepsilon$. Thus, this parameter allows one to quantify the "timing error" made during the bisimulation. A strong and a weak variant of this notion is given in the following definition.

**Definition 1.** *Given a TTS $(S, s_0, \Sigma, \mathbb{K}, \to)$, and $\varepsilon \geq 0$, a symmetric relation $R_\varepsilon \subseteq S \times S$ is a*

- strong timed $\varepsilon$-bisimulation*, if for any $(s,t) \in R_\varepsilon$ and $\sigma \in \Sigma, d \in \mathbb{K}$,*
  - *$s \xrightarrow{\sigma} s'$ implies $t \xrightarrow{\sigma} t'$ for some $t' \in S$ with $(s',t') \in R_\varepsilon$,*
  - *$s \xrightarrow{d} s'$ implies $t \xrightarrow{d'} t'$ for some $t' \in S$ and $d' \in \mathbb{K}$ with $|d - d'| \leq \varepsilon$ and $(s',t') \in R_\varepsilon$.*
- timed-action $\varepsilon$-bisimulation*, if for any $(s,t) \in R_\varepsilon$, and $\sigma \in \Sigma$, $d \in \mathbb{K}$,*
  - *$s \xrightarrow{d,\sigma} s'$ implies $t \xrightarrow{d',\sigma} t'$ for some $t' \in S$ and $d' \in \mathbb{K}$ with $|d - d'| \leq \varepsilon$ and $(s',t') \in R_\varepsilon$.*

*If there exists a strong timed $\varepsilon$-bisimulation (resp. timed-action $\varepsilon$-bisimulation) $R_\varepsilon$ such that $(s,t) \in R_\varepsilon$, then we write $s \sim_\varepsilon t$ (resp. $s \approx_\varepsilon t$). Furthermore we write $s \sim_{\varepsilon^+} t$ (resp. $s \approx_{\varepsilon^+} t$) whenever for every $\varepsilon' > \varepsilon$, $s \sim_{\varepsilon'} t$ (resp. $s \approx_{\varepsilon'} t$).*

Observe that $s \sim_\varepsilon t$ implies $s \sim_{\varepsilon'} t$ for every $\varepsilon' > \varepsilon$. Also, $s \sim_{\varepsilon^+} t$ does not imply $s \sim_\varepsilon t$ in general (see Fig. 1), and if $s \sim_{\varepsilon^+} t$ but $s \not\sim_\varepsilon t$, then $\varepsilon = \inf\{\varepsilon' > 0 \mid s \sim_{\varepsilon'} t\}$. These observations hold true in the timed-action bisimulation setting as well. Note also that $s \sim_\varepsilon t$ implies $s \approx_\varepsilon t$. Finally, for $\varepsilon > 0$, strong timed or timed-action $\varepsilon$-bisimilarity relations are not equivalence relations in general, but they are when $\varepsilon = 0$.

Last, we define a variant of *ready-simulation* [18] for timed transition systems. For $\mathsf{Bad} \subseteq \Sigma$, we will write $I \sqsubseteq^{\mathsf{Bad}} S$ when $I$ is simulated by $S$ (and time delays are matched exactly) in such a way that at any time during the simulation, any failure (*i.e.*, any action in $\mathsf{Bad}$) enabled in $S$ is also enabled in $I$. So, if $I \sqsubseteq^{\mathsf{Bad}} S$ and $S$ is safe w.r.t. $\mathsf{Bad}$ (*i.e.*, $\mathsf{Bad}$ actions are never enabled), then any


**Fig. 1.** An automaton in which $(s,0) \sim_{0^+} (t,0)$ but $(s,0) \not\sim_0 (t,0)$.

run of $I$ can be executed in $S$ (with exact timings) without enabling any of the Bad-actions. Fig. 2 will provide an automaton illustrating the importance of this notion. More formally:

**Definition 2.** *Given a TTS $(S, s_0, \Sigma, \mathbb{K}, \to)$, and a set $\mathsf{Bad} \subseteq \Sigma$, a relation $R \subseteq S \times S$ is a* ready-simulation w.r.t. Bad *if, whenever $(s,t) \in R$:*

- *for all $\sigma \in \Sigma$ and $d \in \mathbb{K}$, $s \xrightarrow{d,\sigma} s'$ implies $t \xrightarrow{d,\sigma} t'$ for some $t' \in S$ with $(s',t') \in R$,*
- *for all $\sigma \in \mathsf{Bad}$, $t \overset{\sigma}{\Longrightarrow} t'$ implies $s \overset{\sigma}{\Longrightarrow} s'$ for some $s' \in S$.*

*We write $s \sqsubseteq^{\mathsf{Bad}} t$ if $(s,t) \in R$ for some ready-simulation $R$ w.r.t.* Bad.

### 2.2 Timed Automata

Given a set of clocks $\mathcal{C}$, the elements of $\mathbb{R}^{\mathcal{C}}_{\geq 0}$ are referred to as *valuations*. For a subset $X \subseteq \mathcal{C}$, and a valuation $v$, we define $v[X \leftarrow 0]$ as the valuation $v[X \leftarrow 0](x) = v(x)$ for all $x \in \mathcal{C} \setminus X$ and $v[X \leftarrow 0](x) = 0$ for $x \in X$. For any $d \in \mathbb{R}_{\geq 0}$, $v + d$ is the valuation defined by $(v+d)(x) = v(x) + d$ for all $x \in \mathcal{C}$. For any $\alpha \in \mathbb{R}$, we define $\alpha v$ as the valuation obtained by multiplying all components of $v$ by $\alpha$, that is $(\alpha v)(x) = \alpha v(x)$ for all $x \in \mathcal{C}$. Given two valuations $v$ and $v'$, we denote by $v + v'$ the valuation that is the componentwise sum of $v$ and $v'$, that is $(v+v')(x) = v(x) + v'(x)$ for all $x \in \mathcal{C}$.

Let $\mathbb{Q}_\infty = \mathbb{Q} \cup \{-\infty, \infty\}$. An *atomic clock constraint* is a formula of the form $k \preceq x \preceq' l$ or $k \preceq x - y \preceq' l$ where $x, y \in \mathcal{C}$, $k, l \in \mathbb{Q}_{\geq 0}$ and $\preceq, \preceq' \in \{<, \leq\}$. A *guard* is a conjunction of atomic clock constraints. For $M, \eta \in \mathbb{Q}_{>0}$ such that $\frac{1}{\eta} \in \mathbb{N}$, we denote by $\Phi_{\mathcal{C}}(\eta, M)$ the set of guards on the clock set $\mathcal{C}$, whose constants are either $\pm\infty$ or less than or equal to $M$ in absolute value and are integer multiples of $\eta$. Let $\Phi_{\mathcal{C}}$ denote the set of all guards on clock set $\mathcal{C}$. A valuation $v$ satisfies $\varphi \in \Phi_{\mathcal{C}}$ if all atomic clock constraints of $\varphi$ are satisfied when each $x \in \mathcal{C}$ is replaced by $v(x)$. Let $[\![\varphi]\!]$ denote the set of valuations that satisfy $\varphi$. We define the *enlargement* of atomic clock constraints by $\Delta \in \mathbb{Q}$ as

$$\langle k \preceq x - y \preceq' l \rangle_\Delta = k - \Delta \preceq x - y \preceq' l + \Delta,$$
$$\text{and} \qquad \langle k \preceq x \preceq' l \rangle_\Delta = k - \Delta \preceq x \preceq' l + \Delta.$$

for $x, y \in \mathcal{C}$ and $k, l \in \mathbb{Q}_{>0}$. The enlargement of a guard $\varphi$, denoted by $\langle \varphi \rangle_\Delta$, is obtained by enlarging all its atomic clock constraints.

**Definition 3.** *A timed automaton $\mathcal{A}$ is a tuple $(\mathcal{L}, \mathcal{C}, \Sigma, l_0, E)$, consisting of a finite set $\mathcal{L}$ of locations, a finite set $\mathcal{C}$ of clocks, a finite alphabet $\Sigma$ of labels, a finite set $E \subseteq \mathcal{L} \times \Phi_{\mathcal{C}} \times \Sigma \times 2^{\mathcal{C}} \times \mathcal{L}$ of edges, and an initial location $l_0 \in \mathcal{L}$.*

*We write* $l \xrightarrow{\varphi,\sigma,R} l'$ *if* $e = (l, \varphi, \sigma, R, l') \in E$*, and call* $\varphi$ *the guard of* $e$*.* $\mathcal{A}$ *is an* integral *timed automaton if all constants that appear in its guards are integers.*

We call the inverses of positive integers *granularities*. The *granularity* of a timed automaton is the inverse of the least common denominator of the finite constants in its guards. For any timed automaton $\mathcal{A}$ and rational $\Delta \geq 0$, let $\mathcal{A}_\Delta$ denote the timed automaton obtained from $\mathcal{A}$ where each guard $\varphi$ is replaced with $\langle\varphi\rangle_\Delta$.

**Definition 4.** *The semantics of a timed automaton* $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \Sigma, l_0, E)$ *is a TTS over alphabet* $\Sigma$*, denoted* $[\![\mathcal{A}]\!]$*, whose state space is* $\mathcal{L} \times \mathbb{R}^{\mathcal{C}}_{\geq 0}$*. The initial state is* $(l_0, \mathbf{0})$*, where* $\mathbf{0}$ *denotes the valuation where all clocks have value* 0. Delay transitions *are defined as* $(l, v) \xrightarrow{\tau} (l, v+\tau)$ *for any state* $(l, v)$ *and* $\tau \in \mathbb{K}$. Action transitions *are defined as* $(l, v) \xrightarrow{\sigma} (l', v')$*, for any edge* $l \xrightarrow{g,\sigma,R} l'$ *in* $\mathcal{A}$ *such that* $v \models g$ *and* $v' = v[R \leftarrow 0]$.

*For any* $k \in \mathbb{N}_{>0}$*, we define the* sampled semantics *of* $\mathcal{A}$*, denoted by* $[\![\mathcal{A}]\!]^{\frac{1}{k}}$ *as the TTS defined similarly to* $[\![\mathcal{A}]\!]$ *by taking the time domain as* $\mathbb{K} = \frac{1}{k}\mathbb{N}$.

We write $[\![\mathcal{A}]\!] \sim_\varepsilon [\![\mathcal{B}]\!]$, $[\![\mathcal{A}]\!] \approx_\varepsilon [\![\mathcal{B}]\!]$ and $[\![\mathcal{A}]\!] \sqsubseteq^{\mathsf{Bad}} [\![\mathcal{B}]\!]$ when the initial states of timed automata $\mathcal{A}$ and $\mathcal{B}$ are related accordingly in the disjoint union of the transition systems, defined in the usual way.

We define the usual notion of region equivalence [3]. Let $M$ be the maximum (rational) constant that appears in the guards of $\mathcal{A}$, let $\eta$ be the granularity of $\mathcal{A}$. Multiplying any constant in $\mathcal{A}$ by $\frac{1}{\eta}$, we obtain an integral timed automaton.

Given valuations $u, v \in \mathbb{R}^{\mathcal{C}}_{\geq 0}$ and rationals $M, \eta$, define $v \simeq^M_\eta u$ to hold if, and only if, for all formulas $\varphi \in \Phi_{\mathcal{C}}(\eta, M)$, $u \models \varphi$ if and only if $v \models \varphi$. The equivalence class of a valuation $u$ for the relation $\simeq^M_\eta$ is denoted by $\mathsf{reg}(u)^M_\eta = \{v \mid u \simeq^M_\eta v\}$. Each such class is called an $(\eta, M)$-region. In the rest, when constant $M$ is (resp. $M$ and $\eta$ are) clear from context, we simply write $\mathsf{reg}(u)_\eta$ (resp. $\mathsf{reg}(u)$) and call these $\eta$-regions (resp. regions). We denote by $\overline{\mathsf{reg}(u)^M_\eta}$ the topological closure of $\mathsf{reg}(u)^M_\eta$. The number of $(\eta, M)$-regions is bounded by $O(2^{|\mathcal{C}|}|\mathcal{C}|!(M/\eta)^{|\mathcal{C}|})$ [3].

For a region $r$, we denote by $r[R \leftarrow 0]$, the region obtained by resetting clocks in $R$. We define $\mathsf{tsucc}^*(r)$ as the set of *time-successor regions of* $r$, that is, the set of $\eta$-regions $r'$ such that $u + d \in r'$ for some $u \in r$ and $d \in \mathbb{R}_{\geq 0}$.

We now associate with each $(\eta, M)$-region a guard that defines it. Assume we number the clocks with indices so that $\mathcal{C} = \{x_1, \ldots, x_m\}$, and fix any $(\eta, M)$-region $r$. Let us define $x_0 = 0$, and $\mathcal{C}_0 = \mathcal{C} \cup \{x_0\}$. Then, for each pair $i, j \in \mathcal{C}_0$, there exists a number $A_{i,j} \in \eta\mathbb{Z} \cap [-M, M] \cup \{\infty\}$ and $\prec_{i,j} \in \{<, \leq\}$ s.t. $\varphi_r$, defined as

$$\varphi_r = \bigwedge_{(x_i, x_j) \in \mathcal{C}_0} -A_{j,i} \prec_{j,i} x_i - x_j \prec_{i,j} A_{i,j},$$

is such that $[\![\varphi_r]\!] = r$. Moreover, we assume that for all $i, j, k \in \mathcal{C}_0$, $A_{i,i} = 0$ and $A_{i,j} \leq A_{i,k} + A_{k,j}$. Note that this is a standard definition: the matrix $(A_{i,j})_{i,j}$ is a *difference-bound matrix (DBM)* that defines region $r$, and the latter condition defines its *canonical form* [13]. Later we will refer to matrix $(A_{i,j})_{i,j}$ as the DBM that defines region $r$.

### 2.3 Quantitative Extension of Computation Tree Logic

In the style of [10, 16, 14] we present a quantitative extension of CTL, which measures (in a sense that we make clear below) how far a formula is from being satisfied in a given state.

**Definition 5.** *Let $\mathbb{I}$ be the set of closed nonempty intervals of $\mathbb{R}_{\geq 0}$, and $\Sigma$ be a finite alphabet. We define the set of state- and path-formulas as follows*[1]

$$\Psi ::= \top \mid \bot \mid \Psi_1 \wedge \Psi_2 \mid \Psi_1 \vee \Psi_2 \mid \mathsf{E}\Pi \mid \mathsf{A}\Pi$$
$$\Pi ::= \mathsf{X}_A^I \Psi \mid \bar{\mathsf{X}}_A^I \Psi \mid \Psi_1 \mathsf{R}^I \Psi_2 \mid \Psi_1 \mathsf{U}^I \Psi_2$$

*for $I \in \mathbb{I}$ and $A \subseteq \Sigma$. We write $\mathcal{L}_T(\Sigma)$ or simply $\mathcal{L}_T$ for the set of state formulae.*

To define the semantics of $\mathcal{L}_T$, we introduce the distance between a point and an interval: $|z, [x, y]| = 0$ when $z \in [x, y]$, and $|z, [x, y]| = \min\{|x - z|, |y - z|\}$ otherwise. Now, given a state $s$, the value of a state formula is defined inductively as follows: $(\!|\top|\!)(s) = 0$, $(\!|\bot|\!)(s) = \infty$, and

$$(\!|\psi_1 \vee \psi_2|\!)(s) = \inf\left\{(\!|\psi_1|\!)(s), (\!|\psi_2|\!)(s)\right\} \qquad (\!|\mathsf{E}\pi|\!)(s) = \inf\left\{(\!|\pi|\!)(\gamma) \mid \gamma \in P(s)\right\}$$
$$(\!|\psi_1 \wedge \psi_2|\!)(s) = \sup\left\{(\!|\psi_1|\!)(s), (\!|\psi_2|\!)(s)\right\} \qquad (\!|\mathsf{A}\pi|\!)(s) = \sup\left\{(\!|\pi|\!)(\gamma) \mid \gamma \in P(s)\right\}$$

For a path $\gamma$, it is defined as:

$$(\!|\mathsf{X}_A^I \psi|\!)(\gamma) = (\!|\bar{\mathsf{X}}_A^I \psi|\!)(\gamma) = \max\{|\gamma(0)_d, I|, (\!|\psi|\!)(\mathsf{state}_1(\gamma))\} \qquad \text{if } \gamma(0)_\sigma \in A$$
$$(\!|\mathsf{X}_A^I \psi|\!)(\gamma) = +\infty \quad \text{and} \quad (\!|\bar{\mathsf{X}}_A^I \psi|\!)(\gamma) = 0 \qquad \text{if } \gamma(0)_\sigma \notin A$$
$$(\!|\psi_1 \mathsf{U}^I \psi_2|\!)(\gamma) = \inf_k \left( \max\left\{ \max_{0 \leq j < k} |(\!|\psi_1|\!)(\mathsf{state}_j(\gamma)), I|,\ (\!|\psi_2|\!)(\mathsf{state}_k(\gamma))\right\}\right)$$
$$(\!|\psi_1 \mathsf{R}^I \psi_2|\!)(\gamma) = \sup_k \left( \min\left\{ \max_{0 \leq j < k} |(\!|\psi_1|\!)(\mathsf{state}_j(\gamma)), I|, (\!|\psi_2|\!)(\mathsf{state}_k(\gamma))\right\}\right)$$

For instance, $(\!|\mathsf{EX}_{\{a\}}^{[2,5]} \top|\!)(s)$ is the lower bound of the set

$$\{\left|d, [2, 5]\right| \mid \text{there is a transition } s \xrightarrow{d,a} s'\}.$$

Intuitively, this semantics measures the amount of point-wise modifications (in the timing constraints of the formula) that are needed for this formula to hold at a given state. Notice that untimed[2] formulas of $\mathcal{L}_T$ can only be evaluated to 0 or $+\infty$, and this value reflects the Boolean value of the underlying CTL formula.

It is shown in [22] that $\mathcal{L}_T$ characterizes $\varepsilon$-bisimilarity between the states of *weighted Kripke structures*. In the following proposition, we generalize one direction of this result to timed automata, showing that $\varepsilon$-bisimilar states have close satisfaction values for all formulas of $\mathcal{L}_T$, which implies that these properties (and their values) are preserved upto $\varepsilon$ by the constructions we give in Section 4.

[1] To establish the relationship to timed-action $\varepsilon$-bisimulation, the logic uses actions on transitions instead of the more usual atomic propositions on states. It is easy to encode the latter by adding extra transitions to sink states.

[2] *I.e.*, when all timing constraints are $[0, +\infty)$.

**Proposition 1.** *For any timed automaton $\mathcal{A}$ and states $s, s'$ of $[\![\mathcal{A}]\!]$, for all $\varepsilon \geq 0$, if $s \approx_{\varepsilon+} s'$ then for any $\psi \in \mathcal{L}_T$ either $(\!|\psi|\!)(s) = (\!|\psi|\!)(s') = \infty$ or $|(\!|\psi|\!)(s) - (\!|\psi|\!)(s')| \leq \varepsilon$.*

## 3 Implementability

As explained in the introduction, even the smallest enlargement of the guards may yield extra behaviour in timed automata. Similarly, any sampling of the time domain may remove behaviours. Here, we give several definitions of *robustness* and *samplability*, which distinguish timed automata whose enlargement (resp. whose sampled semantics) is $\varepsilon$-bisimilar to the original automaton, for some $\varepsilon$.

### 3.1 Robustness

Earlier work on robustness based on enlargement, such as [11, 6, 7] concentrated on deciding the existence of a positive $\Delta$ under which the enlarged automaton is correct w.r.t. a given property. Here, we consider a stronger notion of robustness, which requires systems to be $\varepsilon$-bisimilar for some $\varepsilon$.

**Definition 6.** *A timed automaton $\mathcal{A}$ is* $\varepsilon$-bisimulation-robust *(or simply $\varepsilon$-robust), where $\varepsilon \geq 0$, if there exists $\Delta > 0$ such that $[\![\mathcal{A}]\!] \approx_{\varepsilon} [\![\mathcal{A}_{\Delta}]\!]$.*

Note that not all timed automata are robust. In fact, in the automaton $\mathcal{A}$ of Fig. 2, location $\ell_3$ is not reachable in $[\![\mathcal{A}]\!]$, but it becomes reachable in $[\![\mathcal{A}_{\Delta}]\!]$ for any $\Delta > 0$ (see [11]).

We do not know whether a timed automaton that is robust for some $\Delta$ is still robust for any $\Delta' < \Delta$, that is, whether $[\![\mathcal{A}]\!] \approx_{\varepsilon} [\![\mathcal{A}_{\Delta}]\!]$ implies $[\![\mathcal{A}]\!] \approx_{\varepsilon} [\![\mathcal{A}_{\Delta'}]\!]$ for $\Delta' < \Delta$, in general. This is the so-called "faster-is-better" property [2, 12], which means that if a property holds in some platform, it also holds in a faster or more precise platform. This is known to be satisfied for simpler notions of robustness mentioned above.

In the next section, we will present our construction which, for any $\mathcal{A}$, produces an alternative automaton $\mathcal{A}'$ which is robust and satisfies $[\![\mathcal{A}]\!] \approx_{\varepsilon} [\![\mathcal{A}'_{\Delta}]\!]$ for all small enough $\Delta$.

Bisimulation is not always sufficient when one wants to preserve state-based safety properties proven for $\mathcal{A}$. For instance, removing edges leading to unsafe states in $\mathcal{A}$ may provide us with a trivially safe automaton under any enlargement. However, edges leading to such states are used to detect failures, so removing these will not necessarily remove the failure (since the states that immediately trigger a failure may still be reachable). Fig. 3 gives such an "incorrect" construction. To cope with this problem, we rely on ready-simulation and require $\mathcal{A}'$ to satisfy $[\![\mathcal{A}'_{\Delta}]\!] \sqsubseteq^{\mathsf{Bad}} [\![\mathcal{A}_{\Delta}]\!]$, where $\mathsf{Bad}$ are distinguished actions leading to unsafe states in $\mathcal{A}$. This means that any run of $[\![\mathcal{A}'_{\Delta}]\!]$ can be realized in $[\![\mathcal{A}_{\Delta}]\!]$, and that no $\mathsf{Bad}$-action is enabled in that run in the latter (and hence no unsafe state is reached). Thus, intuitively, no state reached in $[\![\mathcal{A}'_{\Delta}]\!]$ corresponds to an unsafe state in $[\![\mathcal{A}_{\Delta}]\!]$, and in particular, if $\mathcal{A}_{\Delta}$ has unsafe runs (leading to unsafe

states), then these cannot be realized in $\mathcal{A}'_\Delta$. Clearly, the automaton in Fig. 3 does not satisfy this. We formalize this idea here.

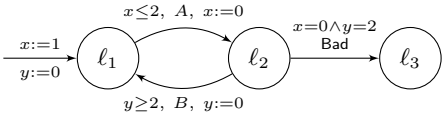



**Fig. 2.** A non-robust timed automaton [20].

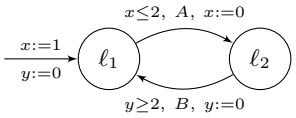



**Fig. 3.** A robust but unsafe alternative.

**Definition 7.** *A timed automaton $\mathcal{A}$ is* safe w.r.t. a set of actions $\mathsf{Bad} \subseteq \Sigma$, *if $d_\infty\big(\mathsf{Reach}(\llbracket\mathcal{A}\rrbracket), Pre(\mathsf{Bad})\big) > 0$, where $d_\infty$ is the standard supremum metric, and $Pre(\mathsf{Bad}) = \bigcup_{\sigma\in\mathsf{Bad},(l,\sigma,g,R,l')\in E} \{l\} \times \llbracket g\rrbracket$ is the precondition of* Bad*-actions*

Notice that $Pre(\mathsf{Bad})$ is the set of states from which a Bad action can be done, and that $d_\infty(\mathsf{Reach}\llbracket\mathcal{A}\rrbracket, Pre(\mathsf{Bad})) = 0$ does not imply that a state of $Pre(\mathsf{Bad})$ is reachable in $\mathcal{A}$. But we still consider such an automaton as unsafe, since, intuitively, any enlargement of the guards may lead to a state of $Pre(\mathsf{Bad})$. It can be seen that automaton of Fig. 2 is safe w.r.t. action Bad. Note that a closed timed automaton is safe w.r.t. Bad iff Bad is not reachable.

Recall the standard notion of robustness, used *e.g.* in [11]:

**Definition 8.** *A timed automaton $\mathcal{A}$ is* safety-robust (w.r.t. Bad) *if there exists $\Delta > 0$ such that $\llbracket\mathcal{A}_\Delta\rrbracket$ is safe w.r.t.* Bad.

In the rest, Bad will refer to a set of actions given with the timed automaton we consider. When we say that a timed automaton is safe, or safety-robust, these actions will be implicit.

We introduce the notion of *safety-robust implementation* (parameterized by a bisimilarity relation $\equiv$, which will range over $\{\sim_0, \sim_{0^+}, \approx_0, \approx_{0^+}\}$), where we only require the alternative automaton to preserve a given safety specification.

**Definition 9 (Safety-Robust Implementation).** *Let $\mathcal{A}$ be a timed automaton which is safe w.r.t. actions* Bad*, and $\equiv$ denote any bisimilarity relation. A* safety-robust implementation of $\mathcal{A}$ w.r.t $\equiv$ *is a timed automaton $\mathcal{A}'$ such that:*

(*i*) *$\mathcal{A}'$ is safety-robust;*
(*ii*) *$\llbracket\mathcal{A}'\rrbracket \equiv \llbracket\mathcal{A}\rrbracket$;*
(*iii*) *there exists $\Delta_0 > 0$ s.t. for all $0 < \Delta' < \Delta < \Delta_0$, $\llbracket\mathcal{A}'_{\Delta'}\rrbracket \sqsubseteq^{\mathsf{Bad}} \llbracket\mathcal{A}_\Delta\rrbracket$.*

Now we define the notion of *robust implementation*. We require such an implementation to be robust and equivalent to the original automaton, and to preserve safety specifications.

**Definition 10 (Robust Implementation).** *Let $\mathcal{A}$ be a timed automaton which is safe w.r.t.actions* Bad*, and $\equiv$ denote any bisimilarity. An* $\varepsilon$-robust implementation of $\mathcal{A}$ w.r.t. $\equiv$ *is a timed automaton $\mathcal{A}'$ such that:*

(*i*) $\mathcal{A}'$ *is* $\varepsilon$*-robust;*
(*ii*) $[\![\mathcal{A}']\!] \equiv [\![\mathcal{A}]\!]$*;*
(*iii*) *there exists* $\Delta_0 > 0$ *s.t. for all* $0 < \Delta' < \Delta < \Delta_0$, $[\![\mathcal{A}'_{\Delta'}]\!] \sqsubseteq^{\mathsf{Bad}} [\![\mathcal{A}_\Delta]\!]$.

### 3.2 Samplability

As we noted in the introduction, some desired behaviours of a given timed automaton may be removed in the sampled semantics. Preservation of the untimed language under some sampling rate was shown decidable in [1]. The proof is highly technical (it is based on the limitedness problem for a special kind of counter automata).We are interested in the stronger notion of *bisimulation-samplability*, which, in particular, implies the preservation of untimed language.

**Definition 11.** *A timed automaton is said to be* $\varepsilon$-bisimulation-samplable *(or simply* $\varepsilon$-samplable*) if there exists a granularity* $\eta$ *such that* $[\![\mathcal{A}]\!] \approx_\varepsilon [\![\mathcal{A}]\!]^\eta$.

Note that not all timed automata are bisimulation-samplable: [17] describes timed automata $\mathcal{A}$ which are not (time-abstract) bisimilar to their sampled semantics for any granularity $\eta$. We define a *sampled implementation* as follows.

**Definition 12 (Sampled Implementation).** *Let* $\mathcal{A}$ *be a timed automaton, and* $\equiv$ *denote any bisimilarity relation. A* $\varepsilon$-sampled implementation w.r.t. $\equiv$ *is a timed automaton* $\mathcal{A}'$ *such that*

(*i*) $\mathcal{A}'$ *is* $\varepsilon$*-samplable;*
(*ii*) $[\![\mathcal{A}']\!] \equiv [\![\mathcal{A}]\!]$.

Note that a similar phenomenon as in Fig. 2 does not occur in sampled semantics since sampling does not add extra behaviour, but may only remove some.

### 3.3 Main result of the paper

We will present two constructions which yield an implementation for any timed automaton. In our first construction, for any timed automaton given with a safety specification, we construct a safety robust implementation. Our second construction is stronger: Given any timed automaton $\mathcal{A}$ and any desired $\varepsilon > 0$, we construct a timed automaton $\mathcal{A}'$ which is both an $\varepsilon$-robust implementation and an $\varepsilon$-sampled implementation of $\mathcal{A}$ w.r.t. $\approx_{0^+}$ (we also give a variant w.r.t. $\sim_0$ for robustness).

Since, $\mathcal{A}$ and $\mathcal{A}'_\Delta$ are timed-action $\varepsilon$-bisimilar, the satisfaction values of the formulas in $\mathcal{L}_T$ are preserved up to $\varepsilon$ (Proposition 1). In particular, all standard untimed linear- and branching-time properties (e.g. expressible in LTL, resp. CTL) proven for the original automaton are preserved in the implementation. An example of such a property is deadlock-freedom, which is an important property of programs.

**Theorem 1.** *Let* $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \Sigma, l_0, E)$ *be an integral timed automaton which is safe w.r.t. some set* $\mathsf{Bad} \subseteq \Sigma$*. Let* $W$ *denote the number of regions of* $\mathcal{A}$*. Then,*

1. *There exists a safety robust implementation of $\mathcal{A}$ w.r.t $\sim_0$, with $|\mathcal{L}|$ locations, the same number of clocks and at most $|E| \cdot W$ edges.*
2. *For all $\varepsilon > 0$, there exists a timed automaton $\mathcal{A}'$ which is a $\varepsilon$-robust implementation w.r.t. $\sim_0$; and a timed automaton $\mathcal{A}''$ which is both a $\varepsilon$-sampled and $\varepsilon$-robust implementation w.r.t. $\approx_{0^+}$. Both timed automata have the same number of clocks as $\mathcal{A}$, and the number of their locations and edges is bounded by $O(|\mathcal{L}| \cdot W \cdot (\frac{1}{\varepsilon})^{|\mathcal{C}|})$.*

The rest of the paper is devoted to the proof of this theorem. The two constructions are presented in the next section, and proved thereafter.

## 4 Making Timed Automata Robust and Samplable

For any timed automaton $\mathcal{A}$ and any location $l$ of $\mathcal{A}$, let $\mathsf{Reach}(\llbracket\mathcal{A}\rrbracket)|_l$ denote the projection of the set of reachable states at location $l$ to $\mathbb{R}^{\mathcal{C}}_{\geq 0}$. For any $l$, there exist guards $\varphi^l_1, \ldots, \varphi^l_{n_l}$ such that $\bigcup_i \llbracket\varphi^l_i\rrbracket = \mathsf{Reach}(\llbracket\mathcal{A}\rrbracket)|_l$ (in fact, the set of reachable states at a given location is a union of regions but not necessarily convex). We use these formulas to construct a new automaton where we restrict all transitions to be activated only at reachable states.

**Definition 13.** *Let $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \Sigma, l_0, E)$ be any integral timed automaton. Define timed automaton $\mathsf{safe}(\mathcal{A})$ from $\mathcal{A}$ by replacing each edge $l \xrightarrow{\varphi,\sigma,R} l'$, by edges $l \xrightarrow{\varphi\wedge\varphi^l_i,\sigma,R} l'$ for all $i \in \{1, \ldots, n_l\}$.*

As stated in Theorem 1 the worst-case complexity of this construction is exponential. However, in practice, $\mathsf{Reach}(\llbracket\mathcal{A}\rrbracket)|_l$ may have a simple shape, which can be captured by few formulas $\varphi^l_i$.

Although the above construction will be enough to obtain a safety-robust timed automaton w.r.t. a given set $\mathsf{Bad}$, it may not be bisimulation-robust. The following construction ensures this.

**Definition 14.** *Let $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \Sigma, l_0, E)$ be an integral timed automaton. Let $M$ be the largest constant that appears in $\mathcal{A}$, and let $\eta$ be any granularity. We define $\mathsf{impl}_\eta(\mathcal{A})$ as a timed automaton over the set of locations $l_r$ where $l$ is a location of $\mathcal{A}$ and $r$ is an $(\eta, M)$-region, and over the same set of clocks. Edges are defined as follows. Whenever there is an edge $l \xrightarrow{\varphi,\sigma,R} l'$ in $\mathcal{A}$, we let $l_r \xrightarrow{\varphi\wedge\varphi_s,\sigma,R} l'_{s[R\leftarrow 0]}$, for all $(\eta, M)$-regions $r$ and $s \in \mathsf{tsucc}^*(r)$ such that $\llbracket\varphi_s\rrbracket \subseteq \llbracket\varphi\rrbracket$.*

*We define $\overline{\mathsf{impl}}_\eta(\mathcal{A})$ as the closed timed automaton obtained from $\mathsf{impl}_\eta(\mathcal{A})$ where each guard is replaced by its closed counterpart*[3]*.*

Throughout this paper, we always consider integral timed automata as input, and the only non-integer constants are those added by our construction. Observe that the size of $\mathsf{impl}_\eta(\mathcal{A})$ depends on $\eta$, since a smaller granularity yields a greater number of $(\eta, M)$-regions.

[3] that is, all $<$ are replaced by $\leq$, and $>$ by $\geq$.

The main theorem is a direct corollary of the following lemma, where we state our results in detail. The bounds on the size of the constructed implementations follow by construction.

**Lemma 1.** *Let $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \Sigma, l_0, E)$ be an integral timed automaton and fix any $\varepsilon > 0$. Assume that $\mathcal{A}$ is safe w.r.t. some set $\mathsf{Bad} \subseteq \Sigma$. Then,*

1. *$\mathsf{safe}(\mathcal{A})$ is safety-robust, $[\![\mathcal{A}]\!] \sim_0 [\![\mathsf{safe}(\mathcal{A})]\!]$ and for any $\Delta < \frac{1}{2|\mathcal{C}|}$, $[\![\mathsf{safe}(\mathcal{A})_\Delta]\!] \sqsubseteq^{\mathsf{Bad}} [\![\mathcal{A}_\Delta]\!]$.*
2. *For any granularity $\eta$ and $\Delta > 0$ such that $2(\eta + \Delta) < \varepsilon$, we have $[\![\mathcal{A}]\!] \approx_{0+} [\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!]$ and $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!] \approx_\varepsilon [\![\overline{\mathsf{impl}}_\eta(\mathcal{A})_\Delta]\!]$. Moreover, for any $0 < \Delta' < \Delta < \frac{1}{|\mathcal{C}|}$, $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})_{\Delta'}]\!] \sqsubseteq^{\mathsf{Bad}} [\![\mathcal{A}_\Delta]\!]$.*
3. *For any granularity $\eta$ and $\Delta > 0$ such that $2(\eta + \Delta) < \varepsilon$, we have $[\![\mathcal{A}]\!] \sim_0 [\![\mathsf{impl}_\eta(\mathcal{A})]\!]$ and $[\![\mathsf{impl}_\eta(\mathcal{A})]\!] \approx_\varepsilon [\![\mathsf{impl}_\eta(\mathcal{A})_\Delta]\!]$. Moreover, whenever $\Delta < \frac{1}{|\mathcal{C}|}$, $[\![\mathsf{impl}_\eta(\mathcal{A})_\Delta]\!] \sqsubseteq^{\mathsf{Bad}} [\![\mathcal{A}_\Delta]\!]$.*
4. *For any granularities $\eta$ and $\alpha$ such that $\eta = k\alpha$ for some $k \in \mathbb{N}_{>0}$ and $\eta < \varepsilon/2$, $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!] \approx_\varepsilon [\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!]^\alpha$.*

Note that both $\overline{\mathsf{impl}}_\eta(\mathcal{A})$ and $\mathsf{impl}_\eta(\mathcal{A})$ provide the relation $\approx_{\varepsilon^+}$ between the specification (that is, $[\![\mathcal{A}]\!]$) and the implementation (that is, $[\![\mathcal{A}'_\Delta]\!]$). However, the latter has a stronger relation with $[\![\mathcal{A}]\!]$, so we also study it separately.

*Trading precision against complexity.* The choice of the granularity in $\overline{\mathsf{impl}}_\eta(\mathcal{A})$ and $\mathsf{impl}_\eta(\mathcal{A})$ allows one to obtain an implementation of $\mathcal{A}$ with any desired precision. However, this comes with a cost since the size of $\overline{\mathsf{impl}}_\eta(\mathcal{A})$ is exponential in the granularity $\eta$. But it is also possible to give up on precision in order to reduce the size of the implementation. In fact, one could define $\mathsf{impl}_\equiv(\mathcal{A})$ where the regions are replaced by the equivalence classes of any finite time-abstract bisimulation $\equiv$. Then, we get $[\![\mathcal{A}]\!] \approx_0 [\![\mathsf{impl}_\equiv(\mathcal{A})]\!]$ and $[\![\mathsf{impl}_\equiv(\mathcal{A})]\!]$ is time-abstract bisimilar to $[\![\mathsf{impl}_\equiv(\mathcal{A})_\Delta]\!]$ for any $\Delta > 0$. In order to obtain, say $[\![\mathsf{impl}_\equiv(\mathcal{A})]\!] \approx_K [\![\mathsf{impl}_\equiv(\mathcal{A})_\Delta]\!]$, for some desired $K \geq 1$, one could, roughly, split these bisimulation classes to sets of delay-width at most $O(K)$, that is the maximal delay within a bounded bisimulation class (there is a subtlety with unbounded classes, where, moreover, all states must have arbitrarily large time-successors within the class). Note however that safety specifications are only guaranteed to be preserved for small enough $K$ (see Lemma 1).

## 5 Proof of Correctness

This section is devoted to the proof of Lemma 1. We start with general properties of regions, in subsection 5.1. In subsection 5.2, we prove the robustness of $\mathsf{impl}_\eta(\mathcal{A})$, $\overline{\mathsf{impl}}_\eta(\mathcal{A})$ and $\mathsf{safe}(\mathcal{A})$, as stated in points 1 through 3 of Lemma 1. In subsection 5.3, we prove that $\overline{\mathsf{impl}}_\eta(\mathcal{A})$ is bisimulation-samplable (point 4). Last, the ready simulation is proved for all the systems in subsection 5.4.

### 5.1 Properties of regions

We give several properties of the enlargement of regions. Fixing constants $\eta$ and $M$, we refer to any $(\eta, M)$-region simply as a region.

**Proposition 2.** *Let $u \in \mathbb{R}^{\mathcal{C}}_{\geq 0}$ such that $u \in [\![\langle \varphi_s \rangle_\Delta]\!]$ for some region $s$. Then for any subset of clocks $R \subseteq C$, $u[R \leftarrow 0] \in [\![\langle \varphi_{s[R \leftarrow 0]} \rangle_\Delta]\!]$.*

The following proposition shows, intuitively, that enlarged guards cannot distinguish the points of an "enlarged region". The proof is straightforward using difference bound matrices in canonical form. Note that the property does not hold if $\varphi_s$ is not in canonical form.

**Proposition 3.** *Let $s$ denote a region, and $\varphi$ a guard. If $[\![\varphi_s]\!] \subseteq [\![\varphi]\!]$, then $[\![\langle \varphi_s \rangle_\Delta]\!] \subseteq [\![\langle \varphi \rangle_\Delta]\!]$.*

**Proposition 4.** *Let $u \in \mathbb{R}^{\mathcal{C}}_{\geq 0}$ such that $u \in [\![\langle \varphi_s \rangle_\Delta]\!]$ for some region $s$. Then for all $s' \in \mathsf{tsucc}^*(s)$, there exists $d \geq 0$ such that $u + d \in [\![\langle \varphi_{s'} \rangle_\Delta]\!]$.*

The previous proposition is no longer valid if $\varphi_s$ is not canonical. As an example, take the region defined by $x = 1 \wedge y = 0$, whose immediate successor is $1 < x < 2 \wedge 0 < y < 1 \wedge x - y = 1$. The enlargement of the former formula is satisfied by valuation $(x = 1 - \Delta, y = \Delta)$ but this has no time-successor that satisfies the enlargement of the latter.

Last, we need the following proposition which provides a bound on the delay that it takes to go from a region to another.

**Proposition 5.** *Let $r$ be a region, and $s$ a time-successor region of $r$, and $\Delta \geq 0$. Suppose that $u \in [\![\varphi_r]\!]$ and $u + d \in [\![\varphi_s]\!]$ for some $d \geq 0$. Then for any $v \in [\![\langle \varphi_r \rangle_\Delta]\!]$, there exists $d' \geq 0$ such that $v + d' \in [\![\langle \varphi_s \rangle_\Delta]\!]$ and $|d' - d| \leq 2\eta + 2\Delta$.*

### 5.2 Proof of Robustness

We first prove that $\mathsf{impl}_\eta(\mathcal{A})$ and $\overline{\mathsf{impl}}_\eta(\mathcal{A})$ are bisimulation-robust, for an appropriate $\varepsilon$, that is $[\![\mathcal{A}']\!] \approx_\varepsilon [\![\mathcal{A}'_\Delta]\!]$ where $\mathcal{A}'$ denotes any of these (Lemma 2). Then we show "faithfulness" results: Lemma 3 shows that $[\![\mathcal{A}]\!] \sim_0 [\![\mathsf{impl}_\eta(\mathcal{A})]\!]$ and $[\![\mathsf{safe}(\mathcal{A})]\!] \sim_0 [\![\mathcal{A}]\!]$, and Lemma 4 shows that $[\![\mathcal{A}]\!] \approx_{0^+} [\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!]$.

**Lemma 2.** *For any timed automaton $\mathcal{A}$, any granularity $\eta$, and any $\Delta > 0$, we have $[\![\mathsf{impl}_\eta(\mathcal{A})]\!] \approx_{2\Delta+2\eta} [\![\mathsf{impl}_\eta(\mathcal{A})_\Delta]\!]$ and $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!] \approx_{2\Delta+2\eta} [\![\overline{\mathsf{impl}}_\eta(\mathcal{A})_\Delta]\!]$.*

*Proof (Sketch).* We fix any $\eta$ and $\Delta$. Let us consider $\mathsf{impl}_\eta(\mathcal{A})$. The case of $\overline{\mathsf{impl}}_\eta(\mathcal{A})$ is similar. We define relation $\mathcal{R} \subseteq (\mathcal{L} \times \mathbb{R}^C) \times (\mathcal{L} \times \mathbb{R}^C)$ between $[\![\mathsf{impl}_\eta(\mathcal{A})]\!]$ and $[\![\mathsf{impl}_\eta(\mathcal{A})_\Delta]\!]$ by $(l_r, u)\mathcal{R}(l'_{r'}, u')$ whenever $l_r = l'_{r'}$ and

$$\forall s \in \mathsf{tsucc}^*(r), \exists d \geq 0, u + d \in [\![\varphi_s]\!] \iff \exists d' \geq 0, u' + d' \in [\![\langle \varphi_s \rangle_\Delta]\!]. \quad (1)$$

Intuitively, relation $\mathcal{R}$ relates states which can reach, by a delay, the same set of regions: we require the first system to reach $[\![\varphi_s]\!]$, while it is sufficient that the

second one reaches $[\![\langle\varphi_s\rangle_\Delta]\!]$, since its guards are enlarged by $\Delta$. Then, Propositions 2, 3, and 4 ensure that this relation is maintained after each transition, proving that $\mathcal{R}$ is a timed-action bisimulation. The parameter $2\Delta + 2\eta$ is given by Proposition 5, applied on relation (1). □

The parameter which we provide for the timed-action bisimilarity is (almost) tight. In fact, consider the automaton in Figure 2, where the guard of the edge entering $\ell_1$ is changed to $x \leq 1$. Fix any $\eta$ and $\Delta$ and consider the following cycle in $\overline{\mathsf{impl}}_\eta(\mathcal{A})$: $(\ell_{1,r_1}) \to (\ell_{2,r_2}) \to (\ell_{1,r_1})$, where $r_1$ is the region $1-\eta < x < 1 \wedge y = 0$, and $r_2$ is the region $x = 0 \wedge 1 < y < 1+\eta$. Suppose $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})_\Delta]\!]$ first goes to location $(\ell_{1,r_1})$ with $x = 1+\Delta, y = 0$, and that this is matched in $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!]$ by $(\ell_{1,r_1'}, (x = 1-\alpha, y = 0))$ where necessarily $\alpha \geq 0$. It is shown in [11] that in any such cycle, the enlarged automaton can reach (by iterating the cycle) all states of the region $r_1$ at location $\ell_1$. In particular, $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})_\Delta]\!]$ can go to state $(\ell_{1,r_1}, (x = 1-\eta, y = 0))$. However, without enlargement, all states $(\ell_1, r_1', (v_x', v_y'))$ reached from a state $(\ell_{1,r_1}, (v_x, v_y))$ with $v_y = 0$ satisfy $v_x' \geq v_x$, that is, the value of the clock $x$ at location $\ell_1$ cannot decrease along any run ([11]). Thus, the state $(\ell_{1,r_1}, (x = 1-\eta, y = 0))$ of $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})_\Delta]\!]$ is matched in $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!]$ by some state $(\ell_{1,r_1''}, (v_x', 0))$ where $v_x' \geq 1-\alpha$. Now, from there, $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})_\Delta]\!]$ can delay $1+\Delta+\eta$ and go to $\ell_2$, whereas $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!]$ can delay at most $1+\alpha$ to take the same transition. The difference between the delays at the first and the last step is then at least $\max\big(\Delta+\alpha, 1+\Delta+\eta-(1+\alpha)\big) \geq \Delta+\eta/2$.

Next, we show that $\mathsf{safe}(\mathcal{A})$ and $\mathsf{impl}_\eta(\mathcal{A})$ are strongly 0-bisimilar to $\mathcal{A}$. The proof is omitted.

**Lemma 3.** *For any timed automaton $\mathcal{A}$, we have $[\![\mathsf{safe}(\mathcal{A})]\!] \sim_0 [\![\mathcal{A}]\!]$, and $[\![\mathcal{A}]\!] \sim_0 [\![\mathsf{impl}_\eta(\mathcal{A})]\!]$ for any granularity $\eta$.* □

The proof of $[\![\mathcal{A}]\!] \approx_{0^+} [\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!]$ is trickier. In fact, since all guards are closed in $\overline{\mathsf{impl}}_\eta(\mathcal{A})$, but not necessarily in $\mathcal{A}$, all time delays may not be matched exactly. The first part of the proof follows the lines of Proposition 16 of [19], who, by a similar construction, prove that the finite timed traces of $[\![\mathcal{A}]\!]$ are dense in those of $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!]$, for an appropriate topology. Their result has a similar flavor, but we consider $0^+$-bisimulation which cannot be interpreted in terms of density in an obvious way.

**Lemma 4.** *For any timed automaton $\mathcal{A}$ and granularity $\eta$, $[\![\mathcal{A}]\!] \approx_{0^+} [\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!]$.*

*Proof (Sketch).* We fix any $\eta$ and $\delta \in (0,1)$. We define $(l,v)\mathcal{R}(l_r, v')$ iff

$$r = \mathsf{reg}(v), v' \in \overline{\mathsf{reg}(v)} \text{ and } \exists v'' \in \mathsf{reg}(v), v = \delta v'' + (1-\delta)v'. \tag{2}$$

We show that $\mathcal{R}$ is a timed-action $0^+$-bisimulation. One direction of the bisimulation follows from convexity of regions, while the other direction is less obvious, and necessitates the following technical lemma. □

**Lemma 5.** *Let $v, v', v'' \in \mathbb{R}_{\geq 0}$ such that $v'' \in \mathsf{reg}(v)$ and $v' \in \overline{\mathsf{reg}(v)}$, and $v = \varepsilon v'' + (1-\varepsilon)v'$ for some $\varepsilon \in (0,1)$. Then for all $d \geq 0$, there exists $d', d'' \geq 0$ s.t. $v+d = \varepsilon(v''+d'') + (1-\varepsilon)(v'+d')$, $v''+d'' \in \mathsf{reg}(v+d)$ and $v'+d' \in \overline{\mathsf{reg}(v+d)}$.*

### 5.3 Proof of Samplability

We now show that $\overline{\mathsf{impl}}_\eta(\mathcal{A})$ is a sampled implementation for any timed automaton $\mathcal{A}$. This result follows from the following lemma and Lemma 4. The proof is similar to Lemma 2.

**Lemma 6.** *Let $\mathcal{A}$ be any integral timed automaton. For any granularities $\eta$ and $\alpha$ such that $\eta = k\alpha$ for some $k \in \mathbb{N}_{>0}$, we have $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!] \approx_{2\eta} [\![\overline{\mathsf{impl}}_\eta(\mathcal{A})]\!]^\alpha$.*

### 5.4 Proof of Safety Preservation (Ready Simulation)

**Lemma 7.** *We have $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})_{\Delta'}]\!] \sqsubseteq^{\mathsf{Bad}} [\![\mathcal{A}_\Delta]\!]$ and $[\![\mathsf{impl}_\eta(\mathcal{A})_\Delta]\!] \sqsubseteq^{\mathsf{Bad}} [\![\mathcal{A}_\Delta]\!]$ for any $0 < \Delta' < \Delta < \frac{1}{|\mathcal{C}|}$; and $[\![\mathsf{safe}(\mathcal{A})_\Delta]\!] \sqsubseteq^{\mathsf{Bad}} [\![\mathcal{A}_\Delta]\!]$ for any $\Delta < \frac{1}{2|\mathcal{C}|}$.*

*Proof (Sketch).* The simulation can be shown similarly to Lemma 3. We show that actions $\mathsf{Bad}$ are not enabled in any state of the simulating run, whenever $\mathcal{A}$ is safe w.r.t. $\mathsf{Bad}$. Let us consider the first statement. Informally, this is due to two facts: (1) the set of reachable states in $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})_\Delta]\!]$ have a small distance (at most $\Delta$) to the corresponding reachable states in $[\![\mathcal{A}]\!]$; (2) the states of $[\![\mathcal{A}]\!]$ have a positive distance to $\mathrm{Pre}_{\mathcal{A}}(\mathsf{Bad})$, which can be bounded from below by $\frac{1}{|\mathcal{C}|}$. Thus, choosing $\frac{1}{|\mathcal{C}|} - \Delta > 0$, we prove that $[\![\overline{\mathsf{impl}}_\eta(\mathcal{A})_{\Delta'}]\!]$ is also safe w.r.t. $\mathsf{Bad}$. □

## 6 Conclusion

We have presented a way to transform any timed automaton into robust and samplable ones, while preserving the original semantics with any desired precision. Such a transformation is interesting if the timed automaton under study is not robust (or not samplable), or cannot be certified as such. In this case, one can simply model-check the original automaton for desired properties, then apply our constructions, which will preserve the specification.

Our constructions also allow one to solve the *robust synthesis* problem. In the synthesis problem, the goal is to obtain automatically (i.e. to synthesize) a timed automaton which satisfies a given property. If one solves this problem for timed automata and obtain a synthesized system $\mathcal{A}$, then applying our constructions, we get that $\overline{\mathsf{impl}}_\eta(\mathcal{A})_\Delta$ and $\overline{\mathsf{impl}}_\eta(\mathcal{A})^\eta$ satisfy the same (say, untimed) properties.

As a future work, we will be interested in *robust controller synthesis*. In this problem, we are given a system $\mathcal{S}$ which we cannot change, and we are asked to synthesize a system $\mathcal{C}$, called controller, such that the parallel composition of the two satisfies a given property. The *robust* controller synthesis is the controller synthesis problem where the behaviour of the controller is $\mathcal{C}_\Delta$ (the controller has imprecise clocks), and we need to decide whether there is some $\Delta$ for which the parallel composition still satisfies the property.

*Acknowledgement.* We thank David N. Jansen for his detailed and insightful comments.

## References


1. P. Abdulla, P. Krčál, and W. Yi. Sampled semantics of timed automata. *Logical Methods in Computer Science*, 6(3:14), 2010.
2. K. Altisen and S. Tripakis. Implementation of timed automata: An issue of semantics or modeling? In FORMATS'05, LNCS 3829, p. 273–288. Springer, 2005.
3. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
4. E. Asarin, O. Maler, and A. Pnueli. On discretization of delays in timed automata and digital circuits. In CONCUR'98, LNCS 1466, p. 470–484. Springer, 1998.
5. C. Baier, N. Bertrand, P. Bouyer, Th. Brihaye, and M. Größer. Probabilistic and topological semantics for timed automata. In FSTTCS'07, LNCS 4855, p. 179–191. Springer, 2007.
6. P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of linear-time properties in timed automata. In LATIN'06, LNCS 3887, p. 238–249. Springer, 2006.
7. P. Bouyer, N. Markey, and P.-A. Reynier. Robust analysis of timed automata via channel machines. In FoSSaCS'08, LNCS 4962, p. 157–171. Springer, 2008.
8. M. Bozga, O. Maler, A. Pnueli, and S. Yovine. Some progress in the symbolic verification of timed automata. In CAV'97, LNCS 1254, p. 179–190. Springer, 1997.
9. F. Cassez, T. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In HSCC'02, LNCS 2289, p. 134–148. Springer, 2002.
10. L. de Alfaro, T. A. Henzinger, and R. Majumdar. Discounting the future in systems theory. In ICALP'03, LNCS, LNCS, p. 1022–1037. Springer, 2003.
11. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robust safety of timed automata. *Formal Methods in System Design*, 33(1-3):45–84, 2008.
12. M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: From timed models to timed implementations. *Formal Aspects of Computing*, 17(3):319–341, 2005.
13. D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In AVMFSS'89, LNCS 407, p. 197–212. Springer, 1990.
14. U. Fahrenberg, K. G. Larsen, and C. Thrane. A quantitative characterization of weighted Kripke structures in temporal logic. *Journal of Computing and Informatics*, 29(6+):1311–1324, 2010.
15. V. Gupta, Th. A. Henzinger, and R. Jagadeesan. Robust timed automata. In HART'97, LNCS 1201, p. 331–345. Springer, 1997.
16. T. A. Henzinger, R. Majumdar, and V. Prabhu. Quantifying similarities between timed systems. In FORMATS'05, LNCS 3829, p. 226–241. Springer, 2005.
17. P. Krčál and R. Pelánek. On sampled semantics of timed systems. In FSTTCS'05, LNCS 3821, p. 310–321. Springer, 2005.
18. K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. In POPL'89, p. 344–352, 1989.
19. J. Ouaknine and J. Worrell. Revisiting digitization, robustness, and decidability for timed automata. In LICS'03, p. 198–207. IEEE Computer Society, 2003.
20. A. Puri. Dynamical properties of timed systems. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.
21. O. Sankur, P. Bouyer, and N. Markey. Shrinking timed automata. Submitted, 2011.
22. C. Thrane, U. Fahrenberg, and K. G. Larsen. Quantitative analysis of weighted transition systems. *Journal Logic and Algebraic Programming*, 79(7):689–703, 2010.