JDEDS manuscript No.
(will be inserted by the editor)

Opacity for Linear Constraint Markov Chains

Béatrice Bérard · Olga Kouchnarenko · John Mullins · Mathieu Sassolas

Received: date / Accepted: date

Abstract On a partially observed system, a secret φ is *opaque* if an observer cannot ascertain that its trace belongs to φ. We consider specifications given as Constraint Markov Chains (CMC), which are underspecified Markov chains where probabilities on edges are required to belong to some set. The nondeterminism is resolved by a scheduler, and opacity on this model is defined as a worst case measure over all implementations obtained by scheduling. This measures the information obtained by a passive observer when the system is controlled by the smartest scheduler in coalition with the observer. When restricting to the subclass of Linear CMC, we compute (or approximate) this measure and prove that refinement of a specification can only improve opacity.

Keywords Opacity · Markov models · Specification · Refinement

Partially supported by a grant from Coopération France-Québec, Service Coopération et Action Culturelle 2012/26/SCAC (French Government), the NSERC Discovery Individual grant No. 13321 (Government of Canada), the FQRNT Team grant No. 167440 (Quebec's Government) and the CFQCU France-Quebec Cooperative grant No. 167671 (Quebec's Government). This research has been partially performed while the third author was visiting the LIP6, Université Pierre & Marie Curie.

Béatrice Bérard
Sorbonne Université, UPMC Paris 06, LIP6, CNRS UMR 7606, Paris, France
INRIA and LSV, CNRS and ENS Cachan, Université Paris-Saclay, France
E-mail: beatrice.berard@lip6.fr

Olga Kouchnarenko
Univ. Bourgogne Franche-Comté, FEMTO-ST, CNRS UMR 6174, Besançon, France
E-mail: olga.kouchnarenko@femto-st.fr

John Mullins
Department of Computer & Software Engineering, École Polytechnique de Montréal, Montreal (Quebec), Canada
E-mail: john.mullins@polymtl.ca

Mathieu Sassolas
Université Paris-Est, LACL, Créteil, France
E-mail: mathieu.sassolas@u-pec.fr

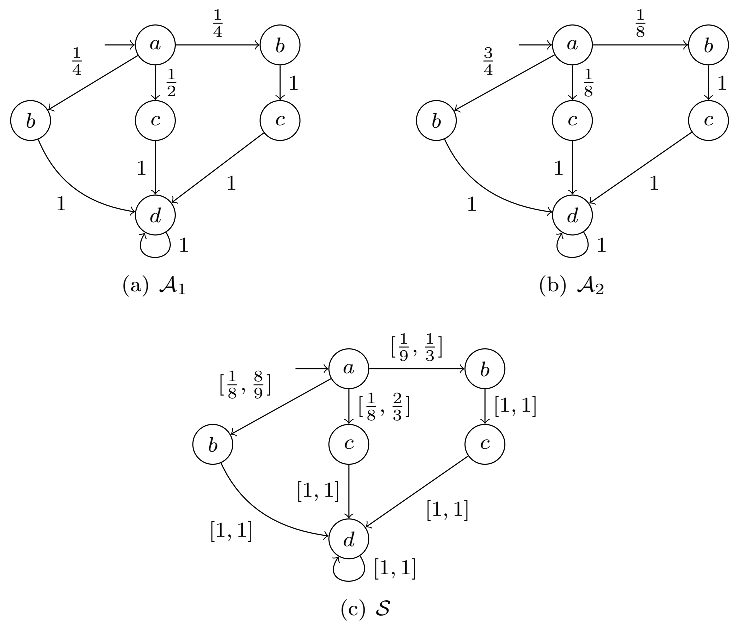
1 Introduction

Context and motivation. When modeling complex systems, a top-down approach allows gradually specifying various system requirements, while preserving some behavioral properties, like safety, reachability, and liveness under some conditions.

Security requirements, which are not behavioral ones [16], may not fare well under refinement, unless tailored specially to do so, as in [1]. Several known security properties such as noninference or anonymity can be encoded in the framework of *opacity* [19,12,1]. In this context, an external observer tries to discover whether a predicate (given as an ω-regular set) holds by partially observing the system through a projection of its actions. A system is opaque if the attacker fails to discover this information. In the possibilistic setting, a violation of opacity captures the existence of at least one perfect leak.

In probabilistic models like Discrete Time Markov Chains (DTMCs), naturally random events such as faults or message transmission failure, can be taken into account. Opacity was extended in this setting [22,8,5] to provide various measures of what is disclosed by observation. In the present work, we focus on the particular measure giving the probability of the set of runs violating opacity (hence violation occurs in a non probabilistic way).

We illustrate this measure on the two systems depicted in Fig. 1(a)-(b), which are DTMCs with the addition of labels on states (indicated inside).



(a) $\mathcal{A}_1$ (b) $\mathcal{A}_2$ (c) $\mathcal{S}$

Fig. 1 Probabilistic systems $\mathcal{A}_1$ or $\mathcal{A}_2$ implementing underspecified system $\mathcal{S}$.

We assume that the occurrence of b must be kept secret and that all labels except b are observable. In this case, the only runs *disclosing* the secret are those observed by ad^ω, since every such run betrays the occurrence of b. The probability of disclosure is $1/4$ in $\mathcal{A}_1$ while it is $3/4$ in $\mathcal{A}_2$, hence $\mathcal{A}_1$ is more secure than $\mathcal{A}_2$. Our aim is to establish sufficient conditions on systems like $\mathcal{A}_1$ and $\mathcal{A}_2$, that can be compared, for one of them to be more secure than the other.

In the process of system modeling, it is common practice to use *underspecified* models as first steps of specification. A first approach is to consider *substochastic* models where transition probabilities need not sum up to 1. In this framework, the notions of satisfaction and simulation were extensively studied in [3]. The second approach is to introduce non-determinism in the model to describe environment choices [18,23,14,13,4,5,17]. These models have also been studied in relation to the refinement process [18], with the particular case of Interval Markov Chains (IMCs) where the transitions are equipped with probability bounds in the form of intervals, as done in Fig. 1(c). For example, both systems of Fig. 1(a)-(b) could have been derived from the single underspecified system $\mathcal{S}$ of Fig. 1(c), with the same structure but imprecise probabilities.

Unfortunately, while closure under conjunction is a nice feature for specification formalisms, IMCs do not have this property. This is shown in [13], where Constraint Markov Chains (CMCs), first introduced in [18], are considered for the specification of finite state processes, and proved to provide a more robust model. In a CMC, the family of intervals associated with a state is replaced by a given (possibly infinite) set of distributions.

Scheduling is an effective way to obtain implementations of a CMC: at each step, a scheduler provides a distribution belonging to the given set, thus producing a (possibly infinite) DTMC on-the-fly. In the case of opacity, a scheduler represents a strategy of an agent inside the system, trying to disclose as much information as possible to a passive observer. Several works used schedulers to evaluate disclosure: [5] in the context of (fully specified) Markov Decision Processes and [6] for IMCs. A simpler way to define implementations of CMCs, which corresponds to considering only memoryless schedulers, would be to choose *a priori* a distribution for each state. This was done for IMC model checking in [14,4], where it is called the Uncertain Markov Chain semantics.

Contributions. We extend the work in [6], investigating opacity for CMCs. As before, disclosure is defined in the worst case scenario, as the supremum of the disclosure for all scheduled implementations. This measures the information obtained by a passive observer when the system is controlled by the smartest scheduler in coalition with the observer.

However, without an explicit description of the given probability sets, algorithmic questions cannot be solved on CMCs. Therefore, we consider here a subclass called Linear Constraint Markov Chains (LCMCs), where the set of distributions associated with a state is defined by linear inequalities. This class is closed under conjunction and contains IMCs.

We first show how to compute the disclosure for a subclass of LCMCs, where no transition can be completely blocked by the scheduler. In the general case, we give an overapproximation for bounded memory schedulers. We then establish monotonicity of the disclosure for LCMCs: refining an LCMC can only improve the opacity of all implementations obtained by scheduling.

Organization of the paper. After short preliminaries on probabilistic transition systems and opacity (Section 2), we recall in Section 3 the background on Constraint Markov Chains, with the associated refinement relations, and we define probabilistic disclosure in this contex. We show how to compute this measure for a restricted case of LCMCs in Section 4, with an approximation scheme for the general case. Finally, we prove monotonicity of opacity under refinement in Section 5.

2 Probabilistic transition systems and opacity

The set of natural numbers is denoted by $\mathbb{N}$ and the set of rational numbers by $\mathbb{Q}$. The composition of relations $\mathcal{R}_2$ and $\mathcal{R}_1$ is defined by $\mathcal{R}_2 \circ \mathcal{R}_1 = \{(x,z) \mid \exists y, (x,y) \in \mathcal{R}_1 \wedge (y,z) \in \mathcal{R}_2\}$. Given a finite alphabet Σ, we denote by Σ^* (resp. Σ^ω) the set of finite (resp. infinite) words over Σ, with $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$ and ε the empty word. We denote by $|w|$ the length of word w in $\mathbb{N} \cup \{+\infty\}$ with the same notation $|E|$ for the cardinality of a set E.

Given a countable set Z, a discrete distribution is a mapping $\mu : Z \to [0,1]$ such that $\sum_{z \in Z} \mu(z) = 1$. The support of μ is $supp(\mu) = \{z \in Z \mid \mu(z) > 0\}$. The set of all discrete distributions on Z is denoted by $\mathcal{D}ist(Z)$. When dealing with a *joint* distribution μ on domain $Z_1 \times Z_2$, we write $\mu(Y_1, Y_2) = \sum_{y_1 \in Y_1, y_2 \in Y_2} \mu(y_1, y_2)$ for $Y_1 \subseteq Z_1$ and $Y_2 \subseteq Z_2$, and we use as shorthands $\mu(y_1, Y_2) = \mu(\{y_1\}, Y_2)$ and $\mu(Y_1, y_2) = \mu(Y_1, \{y_2\})$.

2.1 Probabilistic Labeled Transition Systems and their languages

The secret to be protected from disclosure must be a measurable set, and a general way to describe it is to use an ω-regular set, since such sets are measurable [25] with robust closure properties. Among the various finite automata accepting those sets of infinite words, Deterministic Parity Automata (DPA) [20] are well suited for the combination with probabilistic models: while deterministic, they can describe any ω-regular language (which is not the case with Büchi automata for instance) and they can be complemented at no cost, which is useful in the disclosure computation process.

Definition 1 (Deterministic Parity Automaton) A deterministic parity automaton (DPA) over finite alphabet Σ is a tuple $\mathcal{A} = (Q, q_0, \theta, F)$, where Q is a finite set of states, $q_0 \in Q$ is an initial state, $\theta : Q \times \Sigma \to Q$ is a transition function, and F is a mapping from Q to a finite set of colors $\{1, \ldots, k\}$.

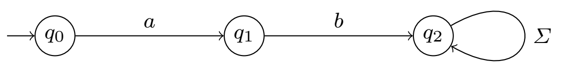


Fig. 2 A DPA for $\varphi_b = ab\Sigma^\omega$ with $F(q_0) = F(q_2) = 1$ and $F(q_2) = 2$.

An example used in the sequel can be seen in Fig. 2. A run of $\mathcal{A}$ on a word $w = a_1a_2 \cdots \in \Sigma^\omega$ is an infinite sequence $\rho = q_0q_1 \cdots \in Q^\omega$ such that for all $i \geq 0$, $q_{i+1} = \theta(q_i, a_{i+1})$. For such a run ρ, we define $\mathit{Inf}(\rho)$ as the set of states appearing infinitely often in the sequence. The run is accepting if $\mathit{min}\{F(q) \mid q \in \mathit{Inf}(\rho)\}$ is even. In this case, the corresponding word is accepted by $\mathcal{A}$ and $\mathcal{L}(\mathcal{A})$ is the subset of Σ^ω of words accepted by $\mathcal{A}$. A subset K of Σ^ω is ω-regular if there is an automaton $\mathcal{A}$ such that $K = \mathcal{L}(\mathcal{A})$.

Implementations are given by Probabilistic Transition Systems (PTSs). They are classical Discrete Time Markov Chains, with the addition of state labeling [3], restricting the processes of [18] with a countable set of states.

Definition 2 (Probabilistic Transition Systems) A *probabilistic transition system* (PTS) over alphabet Σ is a tuple $\mathcal{A} = (Q, q_{init}, \Delta, L)$ where Q is a countable set of states, with $q_{init} \in Q$ the initial state, $\Delta : Q \to \mathcal{D}ist(Q)$ is a mapping associating with any state $q \in Q$ a distribution $\Delta(q)$ over Q, with finite support, and $L : Q \to \Sigma$ is the labeling function on states.

PTSs can be seen as Discrete Time Markov Chains where the states are equipped with labels in Σ. While our presentation uses an alphabet Σ of labels for simplicity, it is always possible as done in [18,13] to consider the case where $\Sigma = 2^A$ for a set A of atomic propositions.

A (finite or infinite) run of $\mathcal{A}$ starting from state $q \in Q$ is a sequence of states $\rho = q_0q_1q_2 \ldots$ such that $q_0 = q$ and for each i, $0 \leq i < |\rho|$, $\Delta(q_i)(q_{i+1}) > 0$. When the run is finite $\rho = q_0q_1 \ldots q_n$, we note $q_n = \mathrm{lst}(\rho)$. We denote by $\mathit{Runs}_q(\mathcal{A})$ the set of infinite runs starting from q and we set $\mathit{Runs}(\mathcal{A}) = \mathit{Runs}_{q_{init}}(\mathcal{A})$, and similarly for finite runs $\mathit{FRuns}_q(\mathcal{A})$ the set of finite runs starting from q and $\mathit{FRuns}(\mathcal{A}) = \mathit{FRuns}_{q_{init}}(\mathcal{A})$. The mapping tr associates with any run ρ its *trace* $\mathrm{tr}(\rho) = L(q_0)L(q_1) \ldots \in \Sigma^\infty$ and we define $\mathit{Tr}(\mathcal{A}) = \{\mathrm{tr}(\rho) \mid \rho \in \mathit{Runs}(\mathcal{A})\}$ and $\mathit{FTr}(\mathcal{A}) = \{\mathrm{tr}(\rho) \mid \rho \in \mathit{FRuns}(\mathcal{A})\}$ respectively the set of infinite and finite traces of $\mathcal{A}$.

Recall [10] that a probability measure $\mathbf{P}_\mathcal{A}$ can be defined on $\mathit{Runs}(\mathcal{A})$: measurable sets are generated by *cones*, where the cone C_ρ associated with a finite run $\rho = q_0q_1 \ldots q_n$ is the subset of infinite runs in $\mathit{Runs}(\mathcal{A})$ having ρ as prefix. The probability of C_ρ is $\mathbf{P}_\mathcal{A}(C_\rho) = \prod_{i=0}^{n-1} \Delta(q_i)(q_{i+1})$. The cone of a word $w \in \Sigma^*$ is defined by $C_w = \bigcup_{\rho \in \mathrm{tr}^{-1}(w)} C_\rho$.

2.2 Probabilistic Opacity

The original definition of opacity was given in [12] for (non probabilistic) transition systems, w.r.t. some observation function $\mathcal{O}$ and some predicate φ (the secret) on the runs of the system.

We consider an *observation function* defined as a morphism $\mathcal{O} : \Sigma^\infty \to \Sigma_o^\infty$, based on a mapping $\pi : \Sigma \to \Sigma_o \cup \{\varepsilon\}$ for a finite alphabet Σ_o. For instance, if $\Sigma = 2^A$ for a set A of atomic propositions, we can define a subset A_o of observable propositions. Then, by setting $A_u = A \setminus A_o$, $\Sigma_o = 2^{A_o}$, π can be defined by $\pi(\sigma) = \varepsilon$ for $\sigma \in 2^{A_u}$ and $\pi(\sigma) = \sigma \cap A_o$ otherwise. For any subset L of Σ^∞, we say that a run ρ of PTS $\mathcal{A}$ satisfies L if its trace belongs to L and we may use L as a shorthand for $\mathrm{tr}^{-1}(L)$.

The secret is given as an ω-regular set $\varphi \subseteq \Sigma^\omega$. It is *opaque* with respect to $\mathcal{A}$ and $\mathcal{O}$ if each time a run satisfies φ, another run with the same observation does not. More precisely, the set of runs violating opacity is defined by $\mathcal{V}(\mathcal{A}, \mathcal{O}, \varphi) = (Tr(\mathcal{A}) \cap \varphi) \setminus (\mathcal{O}^{-1}(\mathcal{O}(Tr(\mathcal{A}) \setminus \varphi)))$ and φ is opaque if $\mathcal{V}(\mathcal{A}, \mathcal{O}, \varphi) = \emptyset$.

This set is used in [7,22] to extend the boolean property by defining various notions of probabilistic opacity. For instance, in [22], one of the measures corresponds to the particular case where φ is the set of traces of runs reaching a fixed set of secret states. We define the probabilistic disclosure as the probability of this set, which is measurable since φ is ω-regular:

Definition 3 (Probabilistic Disclosure) Let $\mathcal{A}$ be a PTS, $\mathcal{O}$ an observation function and φ an ω-regular predicate. The *probabilistic disclosure* of φ in $\mathcal{A}$ for $\mathcal{O}$ is $Disc(\mathcal{A}, \mathcal{O}, \varphi) = \mathbf{P}_{\mathcal{A}}(\mathcal{V}(\mathcal{A}, \mathcal{O}, \varphi))$.

For instance, recall systems $\mathcal{A}_1$ and $\mathcal{A}_2$ of Fig. 1. The secret predicate in this case is the set $\varphi_b = ab\Sigma^\omega$, accepted by the DPA in Fig. 2, and the observation function is the projection π onto $\{a, c, d\}^\omega$. This predicate is not opaque since the run abd^ω discloses the occurrence of b. This is measured by the disclosure: $Disc(\mathcal{A}_1, \pi, \varphi_b) = \mathbf{P}_{\mathcal{A}_1}(abd^\omega) = \frac{1}{4}$ and $Disc(\mathcal{A}_2, \pi, \varphi_b) = \mathbf{P}_{\mathcal{A}_2}(abd^\omega) = \frac{3}{4}$.

Remark that disclosure only measures probabilities of the observer being *sure* that the run is in the secret. For example, one can model anonymity of an agent α initiating some protocol by defining φ_α as the set of all runs initiated by α. Anonymity of α is then equivalent to opacity of φ_α. In the case where anonymity is not guaranteed, disclosure provides a measure of the threat. In the case where anonymity holds, this measure will be 0 and does not give any insight on the "strength" of anonymity. Other notions measuring this strength were proposed in [15,9] and quantitative opacity for partial disclosure of the secret have also been defined in [8], although they are not linear hence do not fare well under standard optimization techniques.

For two PTSs $\mathcal{A}_1$ and $\mathcal{A}_2$ over the same alphabet Σ, predicate φ and observation function $\mathcal{O}$, we say that $\mathcal{A}_1$ is more opaque than $\mathcal{A}_2$ if $Disc(\mathcal{A}_1, \mathcal{O}, \varphi) \leq Disc(\mathcal{A}_2, \mathcal{O}, \varphi)$.

3 Constraint Markov Chains, Opacity and Refinement

3.1 Constraint Markov Chains and Opacity

Constraint Markov chains (CMCs) were first introduced in [18] as a specification formalism and further studied in [13]. They generalize Interval Markov

Chains (IMCs), that were also investigated in [14,4,6] and extended with parameters in [17] with a focus on the consistency problem, *i.e.*, the problem of existence of an implementation satisfying a given specification.

Definition 4 (Constraint Markov Chains) A Constraint Markov Chain (CMC) over alphabet Σ is a tuple $\mathcal{S} = (S, s_{init}, T, \lambda)$ where S is a finite set of states, with $s_{init} \in S$ the initial state, $T : S \to 2^{Dist(S)}$ associates with any state $s \in S$ a set $T(s)$ of distributions over S, and $\lambda : S \to 2^{\Sigma}$ is the labeling function.

The class of CMCs is very general and benefits from nice closure properties as shown in [13]. However, without an explicit description of the sets $T(s)$ for $s \in S$, algorithmic questions cannot be solved. For our purpose, we consider the subclass of Linear CMCs, which is interesting for several reasons (detailed in the sequel): (1) the sets of probabilities $T(s)$, for $s \in S$, can be seen as convex polytopes, with good algorithmic properties, (2) the class (strictly) contains IMCs, hence all results will also apply to IMCs, and (3) it is closed by intersection, which can prove useful in the specification process. The larger class of polynomial CMCs remains to be studied w.r.t. opacity properties.

In a Linear CMC, each set $T(s)$ for $s \in S$ is defined by a conjunction of linear inequalities, which implies the closure under intersection. Moreover, the label $\lambda(s)$ is a singleton. In this case, with a slight abuse of notation, we note $\lambda(s) = a$ instead of $\lambda(s) = \{a\}$, as done in Fig. 1(c).

More precisely, a *linear constraint* over S is of the form $\sum_{s \in S} \alpha_s x_s \bowtie \beta$, with all α_s and β in $\mathbb{Q}$, $\bowtie \in \{<, \leq, =, \geq, >\}$, and each x_s is a variable for state s. A *linear probability set* on S is a subset of $Dist(S)$ where the distributions $\mu = (\mu(s))_{s \in S}$ are the solutions of a system of linear constraints over S. Remark that the conditions for μ to be a distribution are also described by linear constraints: $\sum_{s \in S} x_s = 1$ and, for all s, $0 \leq x_s \leq 1$; these constraints are implicit in the sequel. Thus a linear probability set is a linear set in $\mathbb{R}^{|S|}$ in the usual sense, hence a convex polytope. We denote by $\mathbb{L}(S)$ the set of linear probability sets on S.

Definition 5 (Linear CMC) A Linear CMC (or shortly LCMC) is a CMC $\mathcal{S} = (S, s_{init}, T, \lambda)$ where for each $s \in S$, $T(s) \in \mathbb{L}(S)$ and $|\lambda(s)| = 1$.

In an IMC, each set $T(s)$ is defined by a family $(I(s, s'))_{s' \in S}$ of intervals in $[0, 1]$ and contains all distributions μ such that for each $s' \in S$, $\mu(s') \in I(s, s')$. Since this can be expressed as a system of linear inequalities by introducing the lower and upper bounds of the intervals, an IMC is an LCMC. Note that any PTS can be seen as an IMC (hence as an LCMC), where each interval is reduced to a point. For convenience, we keep the interval notation when applicable.

For example, consider the LCMC $\mathcal{S}$ of Fig. 3, representing a simple system. For graphical depiction, an edge from s to s' is labeled by the variable for $x_{s'}$ in the system of linear constraints defining $T(s)$; as usual, absence of edge (s, s') means $x_{s'} = 0$ is a constraint of $T(s)$. The aim of this system is to

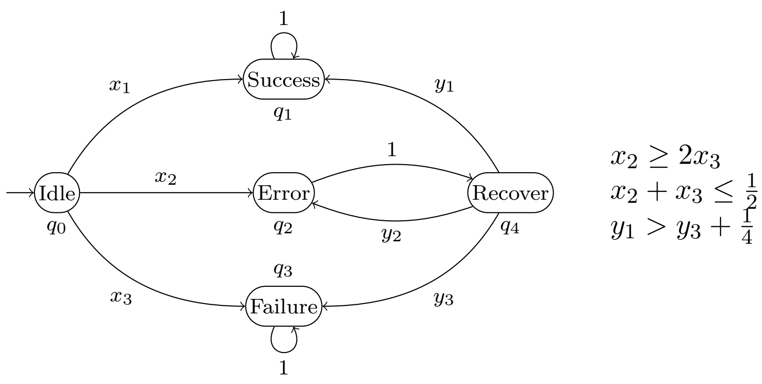


Fig. 3 An LCMC $\mathcal{S}$ which is not a conjunction of IMCs.

achieve *Success*, but there can be errors (which may be recovered) and failures (which cannot). The probabilities underlying the behavior of the system are not fixed, although a certain number of constraints are known. For example, the probability of a recoverable error is at least twice the one of an unrecoverable one, as expressed by the equation $x_2 \geq 2x_3$, and when trying to recover from an error, there is a probability of success exceeding the probability of definite failure by $\frac{1}{4}$.

Note that $\mathcal{S}$ is not an IMC. Indeed, assume that the constraints on $T(q_0)$ can be expressed by intervals. Any values for x_1, x_2, and x_3 that satisfy the linear constraints should be in the intervals. In particular, it would be the case for the three tuples of values for (x_1, x_2, x_3): $(1, 0, 0)$, $(\frac{1}{2}, \frac{1}{2}, 0)$, and $(\frac{1}{2}, \frac{1}{3}, \frac{1}{6})$. Hence $[\frac{1}{2}, 1] \subseteq I(q_0, q_1)$, $[0, \frac{1}{2}] \subseteq I(q_0, q_2)$, and $[0, \frac{1}{6}] \subseteq I(q_0, q_3)$ and the distribution defined by $x_1 = \frac{5}{6}$, $x_2 = 0$, $x_3 = \frac{1}{6}$ is within the bounds of the intervals, although it does not satisfy the constraint $x_2 \geq 2x_3$, which is a contradiction.

In addition, $\mathcal{S}$ is neither the result of conjunction of several IMCs, as those only yield constraints where coefficients are positive, hence constraint $x_2 \geq 2x_3$ (which is actually $x_2 - 2x_3 \geq 0$) cannot be expressed.

Several semantics can be given to CMCs with respect to the set of PTSs they specify. The simplest one corresponds to first choosing for each state s a distribution belonging to $T(s)$, thus producing a PTS (the implementation), with the same structure as the CMC (the specification). This was done for IMCs in [14,4] and called the Uncertain Markov Chain semantics. A richer semantics consists in introducing a scheduler, choosing the distribution at each step to obtain an implementation, as in a Markov Decision Process (MDP). This was also defined in [14] for IMCs, seen as a special case of Interval Markov Decision Process.

Finally, the most general semantics corresponds to the satisfaction relation from [18,13], restricted in [13] to finite state processes. This relation can be obtained from the refinement defined below in Section 3.2.

We consider here the MDP semantics. A run of $\mathcal{S}$ starting from a state s is a sequence $s \xrightarrow{\mu_1} s_1 \xrightarrow{\mu_2} \ldots$ where $s_i \in S$ and each μ_i is a distribution over S such that $\forall s \in S, \mu_i \in T(s_{i-1})$. As before, we denote by $Runs_s(\mathcal{S})$ the set of runs starting from s, we set $Runs(\mathcal{S}) = Runs_{s_{init}}(\mathcal{S})$, $FRuns(\mathcal{S})$ is the set of finite runs of $\mathcal{S}$ starting from s_{init}, and for a run $\rho = s \xrightarrow{\mu_1} s_1 \xrightarrow{\mu_2} \ldots s_{n-1} \xrightarrow{\mu_n} s_n$ in $FRuns(\mathcal{S})$ we define $\mathrm{lst}(\rho) = s_n$.

A scheduler resolves the non determinism by choosing a distribution at each step. More precisely:

Definition 6 (Scheduler) A scheduler A for a CMC specification $\mathcal{S} = (S, s_{init}, T, \lambda)$, is a mapping $A : FRuns(\mathcal{S}) \to \mathcal{D}ist(S)$ such that for each run ρ with $s = \mathrm{lst}(\rho)$, $A(\rho) \in T(s)$.

Note that the Uncertain Markov Chains semantics corresponds to the particular case of memoryless schedulers, where the distribution only depends on the last state of the run. We denote by $Sched(\mathcal{S})$ the set of schedulers for $\mathcal{S}$. Like for Markov Decision Processes, scheduling $\mathcal{S}$ with A produces a PTS denoted by $\mathcal{S}(A)$, where states are finite runs of $\mathcal{S}$ and the labelings must be consistent: $\mathcal{S}(A) = \langle Q, q_{init}, \Delta, L\rangle$ with $Q \subseteq FRuns(\mathcal{S})$, the initial state is $q_{init} = s_{init}$, the run containing only the initial state of $\mathcal{S}$, for $\rho \in Q$, $L(\rho) \in \lambda(\mathrm{lst}(\rho))$ and $\Delta(\rho)(\rho') = A(\rho)(s')$ for $\rho' = \rho \xrightarrow{A(\rho)} s'$, and 0 otherwise. We note:

$$\underline{sat}(\mathcal{S}) = \{\mathcal{S}(A) \mid A \in Sched(\mathcal{S})\}.$$

We now lift the notion of disclosure to the set of scheduled implementations of a specification $\mathcal{S}$ by:

$$Disc(\mathcal{S}, \mathcal{O}, \varphi) = \sup_{A \in Sched(\mathcal{S})} Disc(\mathcal{S}(A), \mathcal{O}, \varphi).$$

3.2 Refinement

The notion of strong refinement between probabilistic specifications was introduced in [18] through simulation. A weaker refinement was also proposed in [13]. These notions are adapted to our setting in Definitions 7 and 8 below.

Definition 7 (Strong refinement relation) For two CMC specifications $\mathcal{S}_1 = (S_1, s_{1,init}, T_1, \lambda_1)$ and $\mathcal{S}_2 = (S_2, s_{2,init}, T_2, \lambda_2)$ over alphabet Σ, $\mathcal{S}_1$ strongly refines $\mathcal{S}_2$, written $\mathcal{S}_1 \preceq_s \mathcal{S}_2$, if there exists a relation $\mathcal{R} \subseteq S_1 \times S_2$ such that $s_{1,init} \mathcal{R}\ s_{2,init}$ and if $s_1 \mathcal{R} s_2$ then:

(1) $\lambda_1(s_1) \subseteq \lambda_2(s_2)$,
(2) there exists a function $\delta : S_1 \to \mathcal{D}ist(S_2)$ such that for all $\mu \in T_1(s_1)$ $\sum_{s_1' \in S_1} \mu(s_1') \cdot \delta(s_1') \in T_2(s_2)$,
(3) $s_1' \mathcal{R} s_2'$ whenever $\delta(s_1')(s_2') > 0$.

For weak refinement, the mapping δ depends on the chosen distribution μ (as well as on s_1 and s_2) instead of being uniform:

Definition 8 (Weak refinement relation) For two CMC specifications $\mathcal{S}_1 = (S_1, s_{1,init}, T_1, \lambda_1)$ and $\mathcal{S}_2 = (S_2, s_{2,init}, T_2, \lambda_2)$ over alphabet Σ, $\mathcal{S}_1$ weakly refines $\mathcal{S}_2$, written $\mathcal{S}_1 \preceq_w \mathcal{S}_2$, if there exists a relation $\mathcal{R} \subseteq S_1 \times S_2$ such that $s_{1,init} \mathcal{R}\ s_{2,init}$ and if $s_1 \mathcal{R} s_2$ then:

(1) $\lambda_1(s_1) \subseteq \lambda_2(s_2)$,
(2) for each $\mu \in T_1(s_1)$ there exists a function $\delta : S_1 \to \mathcal{D}ist(S_2)$ such that $\sum_{s'_1 \in S_1} \mu(s'_1) \cdot \delta(s'_1) \in T_2(s_2)$,
(3) $s'_1 \mathcal{R} s'_2$ whenever $\delta(s'_1)(s'_2) > 0$.

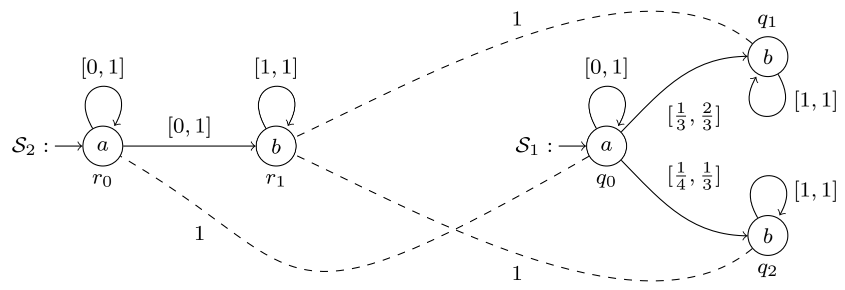


Fig. 4 A refinement of $\mathcal{S}_2$ by $\mathcal{S}_1$.

Fig. 4 illustrates the strong refinement relation $\mathcal{R}$ of $\mathcal{S}_2$ by $\mathcal{S}_1$. For Condition (2) above, we may uniformly use the function: $\delta_0(q_i)(r_j) = 1$ if $(q_i, r_j) \in \mathcal{R}$ and 0 otherwise, represented with dashed lines in Fig. 4.

Note that there is no weak refinement relation of $\mathcal{S}_1$ by $\mathcal{S}_2$. Indeed, let $\mu \in T_2(r_0)$ defined by $\mu(r_0) = \frac{4}{5}$ and $\mu(r_1) = \frac{1}{5}$. Given any function $\delta : S_2 \to \mathcal{D}ist(S_1)$ as in Definition 8, it must be the case that $\delta(r_0)(q_1) = 0$ since labels do not match. However, $\mu(r_1) \cdot \delta(r_1)(q_1) \leq \mu(r_1) = \frac{1}{5} \notin [\frac{1}{3}, \frac{2}{3}]$. So $\mu(r_1) \cdot \delta(r_1)$ is not a distribution in $T_1(q_0)$, which violates Condition (2).

When a PTS refines a specification, both notions coincide and correspond to the satisfaction relation, which defines the most general semantics of CMCs (from [18]): A PTS $\mathcal{A} = (Q, q_{init}, \Delta, L)$ implements a CMC $\mathcal{S} = (S, s_{init}, T, \lambda)$, written $\mathcal{A} \models \mathcal{S}$ for the associated relation $\models \subseteq Q \times P$, if $\mathcal{A}$ refines $\mathcal{S}$, where $\mathcal{A}$ is seen as a CMC with point distributions. Specification $\mathcal{S}$ is said *consistent* if it admits at least one implementation.

Refinement also applies to two PTSs, and since each probability set reduces to a single distribution, Condition (2) becomes (2′):

$$(2') \quad \text{For all } s'_2 \in S_2, \sum_{s'_1 \in S_1} \Delta_1(s_1)(s'_1) \cdot \delta(s'_1)(s'_2) = \Delta_2(s_2)(s'_2).$$

It is proved in [13] that $\mathcal{S}_1 \preceq_s \mathcal{S}_2$ implies $\mathcal{S}_1 \preceq_w \mathcal{S}_2$, which in turn implies that for any PTS $\mathcal{A}$, if $\mathcal{A} \models \mathcal{S}_1$ then $\mathcal{A} \models \mathcal{S}_2$, all implications being strict.

Finally, it can be seen that scheduling a CMC specification is a particular case of implementation:

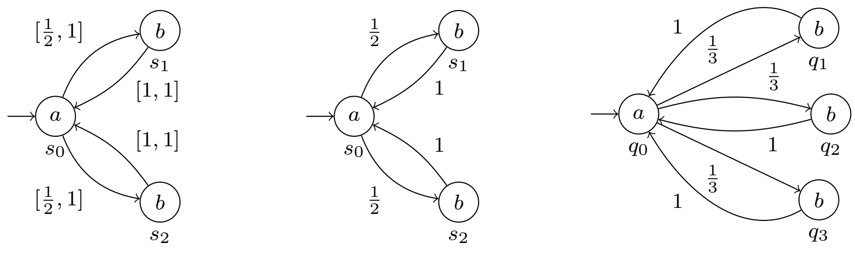
Proposition 1 *Let $\mathcal{S}$ be a CMC specification. For each scheduler A of $\mathcal{S}$, $\mathcal{S}(A) \models \mathcal{S}$.*

Proof The relation $\mathcal{R} \subseteq Q \times S$ is defined by $\mathcal{R} = \{(\rho, s) \mid \mathrm{lst}(\rho) = s\}$. We prove that the relation $\mathcal{R}$ is a refinement relation by defining $\delta_{(\rho,s)} : Q \to \mathcal{D}ist(S)$ as follows:

$$\text{For } \rho' \in Q, s' \in S,\ \delta_{(\rho,s)}(\rho')(s') = \begin{cases} 1 & \text{if } \rho' = \rho \xrightarrow{A(\rho)} s', \\ 0 & \text{otherwise.} \end{cases}$$

The first condition of refinement results from the definition of the labeling and the third one is direct from the definition of $\delta_{(\rho,s)}$. For Condition (2), the only distribution μ is $\Delta(\rho)$ with $\Delta(\rho)(\rho') = A(\rho)(s')$ if $\rho' = \rho \xrightarrow{A(\rho)} s'$ and 0 otherwise. But then $\sum_{\rho' \in Q} \Delta(\rho)(\rho')\delta_{(\rho,s)}(\rho')$ is simply equal to $A(\rho)$ and belongs to $T(s)$ by definition of a scheduler. $\square$

For any scheduler A, $\mathcal{S}(A)$ is a kind of *unfolding* of $\mathcal{S}$, which restricts the structure of $\mathcal{S}(A)$: at each step, the scheduler chooses a valid distribution over successor states. Hence not every implementation can be mapped to a scheduler. Said otherwise, not all implementations can be put in relation with $\mathcal{S}$ through a satisfaction relation where the joint distributions are diagonal.



(a) A specification $\mathcal{S}_0$ (b) $\mathcal{A}_1$, the only scheduling of $\mathcal{S}_0$ (c) $\mathcal{A}_2$, an implementation (not a scheduling) of $\mathcal{S}_0$

Fig. 5 A specification with an implementation that is not the result of scheduling.

For example, consider the specification $\mathcal{S}_0$ of Fig. 5(a). There is a single possible scheduler for this specification: the one that picks in s_0 probability $\frac{1}{2}$ to go to either s_1 or s_2 ($\mathcal{A}_1$ in Fig. 5(b)). However, the PTS $\mathcal{A}_2$ of Fig. 5(c) is also an implementation of this specification ($\mathcal{A}_2 \models \mathcal{S}_0$) where q_2 is split between s_1 and s_2. The mapping $\delta_{(q_0,s_0)}$ can be represented by the following matrix, giving on column i the distribution of state q_i on the states of $\mathcal{S}_0$:

$$\delta_{(q_0,s_0)} = \begin{array}{c|cccc} & q_0 & q_1 & q_2 & q_3 \\ \hline s_0 & 1 & 0 & 0 & 0 \\ s_1 & 0 & 1 & \frac{1}{2} & 0 \\ s_2 & 0 & 0 & \frac{1}{2} & 1 \end{array}$$

3.3 Motivating example

Consider a simple access control database mechanism to a medical database (inspired from [11]) as illustrated in Fig. 6(a). In $\mathcal{S}_2$, a user is first requested to input a username (a). If the user name is not on the list of authorized medical staff (d), the request is rejected (e) and otherwise, the user is requested to input a password (b). If the password is correct, access to the database records is provided (c) and is refused (e), otherwise. The transition probabilities take in account many factors *e.g.*, the number of legitimate users of the database, the robustness of their respective passwords, or the attacker's knowledge.

System $\mathcal{S}_1$ depicted in Fig. 6(b) refines the password verification process. If the password is correct, it is accepted. Otherwise, a lookup of a black list of common password is performed. If the password is in the list, access is refused, considering it is a malicious attacker and the user is allowed to try again, otherwise. The refinement restricts intervals and splits the state q_1 onto r_1 and r_2 in $\mathcal{S}_1$. In order to express security requirements that any implementation of these specifications should assume, consider the observation function $\mathcal{O}$ defined by $\mathcal{O}(x) = x$ for all $x \neq b$ and $\mathcal{O}(b) = \varepsilon$. It reflects the fact that the password input by the user should be kept secret. This could be realized by mean of some cryptographic infrastructure later on along the refinement process but it is not useful to specify this any further here.

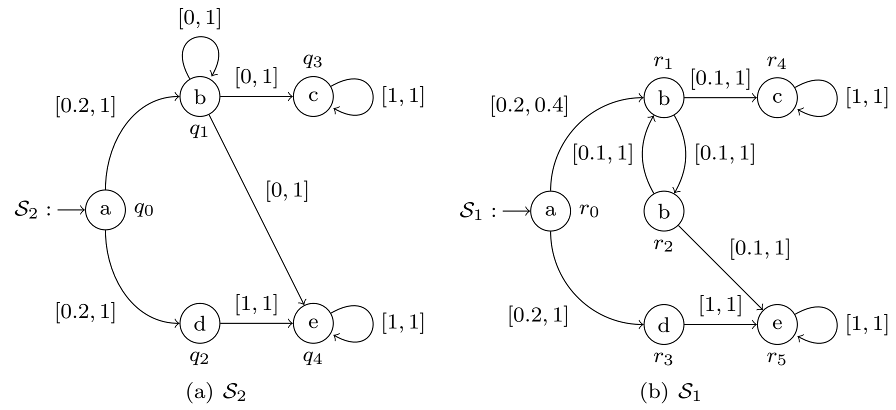


Fig. 6 An access control mechanism to a medical database $\mathcal{S}_2$ and its refinement $\mathcal{S}_1$.

It is well known that this kind of mechanism is not robust enough to provide an adequate protection against covert channels allowing an attacker to access any private information from patient medical record whose access should be strictly limited to medical professionals, the legitimate users of the database. As an example, imagine an observer who could discriminate between abc^ω, the execution where a legitimate user is accepted at the first password try and any execution in abb^+c^ω where he is only accepted after several tries. This could serve as a channel between this user and an attacker on behalf of whom he is acting for in order to leak any private information of any patient record. Requesting as best opacity as possible of the predicate $\varphi = abc^\omega$, for all implementations of the specification, tears down the possibility of such information flow.

Looking at the opacity of φ, we first establish that $Disc(\mathcal{S}_2, \mathcal{O}, \varphi) > 0$. Since the interval labeling the loop on q_1 contains 0, there is a distribution $\mu \in T_2(q_1)$ such that $\mu(q_1) = 0$. This loop *may* or *may not* be scheduled by an attacker. Hence any implementation $\mathcal{S}_2(A)$ of $\mathcal{S}_2$ blocking the loop on q_1 but not edges (q_1, q_3) and (q_1, q_4), discloses φ. Indeed, in this case,

$$\begin{aligned} L_2 = Tr(\mathcal{S}_2(A)) &= abc^\omega + abe^\omega + ade^\omega, \\ L_2 \setminus \varphi &= abe^\omega + ade^\omega, \\ \mathcal{O}(\varphi) &= ac^\omega \text{ and} \\ \mathcal{O}(L_2 \setminus \varphi) &= ae^\omega + ade^\omega \end{aligned}$$

proving that $\mathcal{O}(\varphi) \not\subseteq \mathcal{O}(L_2 \setminus \varphi)$.

This does not happen in $\mathcal{S}_1$, because for any scheduler A all edges have positive probabilities, in particular from r_1 there is positive probability to go to r_2 and come back to r_1. Hence,

$$\begin{aligned} L_1 &= Tr(\mathcal{S}_1(A)) = abc^\omega + ab(bb)^+c^\omega + a(bb)^+e^\omega + ade^\omega \\ &\text{and } \mathcal{O}(L_1 \setminus \varphi) = ac^\omega + ae^\omega + ade^\omega, \end{aligned}$$

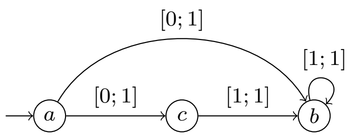
proving that φ is opaque for the infinitely many implementations $\mathcal{S}_1(A)$ of $\mathcal{S}_1$ that is, $Disc(\mathcal{S}_1, \mathcal{O}, \varphi) = 0$.

In Section 4 we give an algorithm to compute the exact disclosure of a specification when edges cannot be blocked (as in $\mathcal{S}_1$) and a scheme to approximate the general case (like for $\mathcal{S}_2$). Finally note that $Disc(\mathcal{S}_1, \mathcal{O}, \varphi) < Disc(\mathcal{S}_2, \mathcal{O}, \varphi)$. In Section 5 we prove that amazingly, despite the fact that, in general, information flow properties are not preserved under refinement, when $\mathcal{S}_1$ refines $\mathcal{S}_2$, it is always the case that $Disc(\mathcal{S}_1, \mathcal{O}, \varphi) \leq Disc(\mathcal{S}_2, \mathcal{O}, \varphi)$, for any secret φ and observation function $\mathcal{O}$.

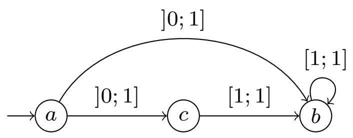
4 Computing the probabilistic disclosure

4.1 Modal edges

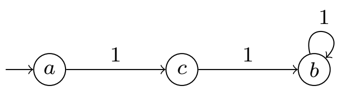
When the probability of an edge can be equal to 0, the corresponding action can be completely blocked by a scheduler. From a modeling point of view,



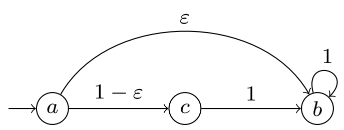
(a) A modal IMC $\mathcal{S}_\mathrm{m}$.



(b) A non-modal IMC $\mathcal{S}_\mathrm{nm}$.



(c) A disclosing implementation of $\mathcal{S}_\mathrm{m}$ (but not of $\mathcal{S}_\mathrm{nm}$).



(d) A non-disclosing implementation of $\mathcal{S}_\mathrm{nm}$ (and $\mathcal{S}_\mathrm{m}$), $\varepsilon > 0$.

Fig. 7 The influence of modal transitions on disclosure.

modal edges add a lot of flexibility for refinement. This however means that the range of potential implementations is larger and so it will be harder to obtain meaningful properties. Therefore such edges are desirable in an early modeling phase but less so in the latest refinements. As in [18], we call these edges *modal* edges, and CMCs that contain such edges are called *modal CMCs*.

Definition 9 (Modal edge) An edge (s, s') in CMC $\mathcal{S}$ is *modal* if there exists a scheduler A such that $\Delta(\rho)(\rho \xrightarrow{A(\rho)} s') = 0$ for any run ρ of $\mathcal{S}(A)$ with $\mathrm{lst}(\rho) = s$.

In the context of opacity, removing an edge drastically changes the disclosure, since it can remove ambiguities. A first example was seen in the previous section, where system $\mathcal{S}_2$ of Fig. 6 is a modal IMC, with a loop around q_1 that can be blocked by a scheduler.

For another example, consider the modal IMC $\mathcal{S}_\mathrm{m}$ of Fig. 7(a), where a and b are observed and the secret is the presence of c. An implementation of $\mathcal{S}_\mathrm{m}$ that blocks the direct edge from a to b (Fig. 7(c)) has a disclosure of 1, since the secret is guaranteed to be part of the only possible run. On the other hand, in the non-modal version of the IMC (Fig. 7(b)), such implementations are banned and only implementations that retain a small probability to avoid c are allowed. In these implementations, the disclosure is 0, since every run is observed as ab^ω and it is possible that c did not occur.

This example shows how our measure differs from the similar one in [5] for Markov Decision Processes. In [5], the set of runs of the disclosure is defined on the (unscheduled) MDP, and its probability is optimized afterwards. Hence viewing $\mathcal{S}_\mathrm{m}$ as an MDP with two actions μ_1 and μ_2, both with probability 1 from the initial state, the disclosure computed in [5] would be 0. The notion presented here is finer since the scheduling is considered as known by the attacker and the set of runs measured by the disclosure depends on the scheduled implementation.

The detection of modal edges is the first step toward computation of the disclosure.

Proposition 2 *The set of modal edges can be computed in time polynomial in the number of edges.*

Proof The procedure for each edge (s, s') is as follows. Assume $T(s)$ is defined by the conjunction $\bigwedge_{i=1}^{k} C_i$ where each C_i is a linear constraint. The edge (s, s') is modal if and only if $x_{s'} = 0 \wedge \bigwedge_{i=1}^{k} C_i$ admits a solution. This can be checked in polynomial time [21]. □

Note that it is not always possible to block several modal edges concomitantly. For example, in $\mathcal{S}_{\mathrm{m}}$ of Fig. 7(a), blocking both edges from state a would yield an *inconsistent* specification. Testing consistency can be done in polynomial time [13], although testing all possible combinations of modal edges *a priori* would require to consider them all, which induces an exponential blowup.

4.2 Computation in non-modal LCMCs

In the case of non-modal LCMCs, the disclosure can be computed:

Theorem 1 *Computing the value of disclosure for an LCMC $\mathcal{S}$ without modal edges can be done in EXPTIME.*

Proof The proof relies on constructing an (exponentially larger) MDP on which an optimization problem is solved. In the spirit of [24], the construction of the MDP relies on basic feasible solutions (BFSs), otherwise called corner points.

Let $\mathcal{B}$ be an arbitrary scheduling of $\mathcal{S}$. The probability information on the edges of $\mathcal{B}$ is of no importance because any scheduling of $\mathcal{S}$ has the same structure $\mathcal{B}$, since $\mathcal{S}$ does not have modal edges. In the rest of the proof, we are interested only in the structure of $\mathcal{B}$, which can be seen an a nondeterministic automaton.

Starting from a DPA $\mathcal{A}_\varphi$ for φ, a DPA $\mathcal{A}_\mathcal{V}$ for $\mathcal{V}(\mathcal{B}, \mathcal{O}, \varphi)$ can be built, with size exponential in the size of $\mathcal{S}$ and $\mathcal{A}_\varphi$ (and with a number k of colors polynomial in the size of $\mathcal{B}$ and $\mathcal{A}_\varphi$). This construction relies on intersections and complementations of DPA, with a determinization step that brings the exponential blowup [20]. Synchronizing the DPA $\mathcal{A}_\mathcal{V}$ with the original LCMC yields an LCMC $\mathcal{S}_\mathcal{V} = \mathcal{A}_\mathcal{V} \times \mathcal{S}$. Finding the optimal scheduler for $\mathcal{S}_\mathcal{V}$ to accept yields the optimal value of $Disc(\mathcal{S}(A), \mathcal{O}, \varphi)$ for any scheduler A. This optimization is done by translating $\mathcal{S}_\mathcal{V}$ into an MDP $\mathcal{M}_\mathcal{V}$ as follows.

For each state s of the $\mathcal{S}$ component of $\mathcal{S}_\mathcal{V}$, we compute the set of BFSs of the polytope defined by $T(s)$. Then we build the corresponding state in the MDP $\mathcal{M}_\mathcal{V}$ by adding a transition (*i.e.*, a probability distribution) per BFS. As a property of BFSs, any distribution in $T(s)$ in $\mathcal{S}_\mathcal{V}$ can be expressed as a

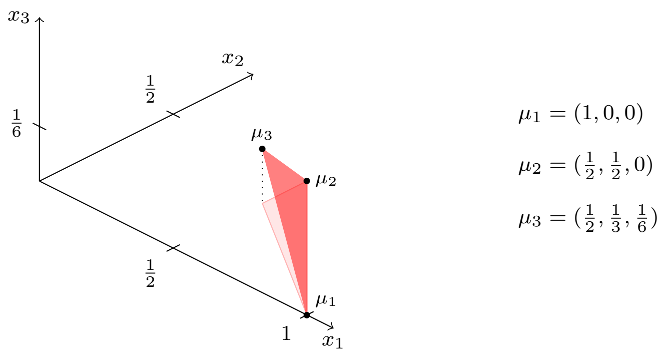


Fig. 8 Basic Feasible Solutions for state q_0.

barycentric combination of BFSs. Hence a scheduler on $\mathcal{M}_\mathcal{V}$ corresponds exactly to a scheduler on $\mathcal{S}_\mathcal{V}$. As a result maximizing the probability of $\mathcal{V}(\mathcal{B}, \mathcal{O}, \varphi)$ in $\mathcal{M}_\mathcal{V}$ is exactly the same as computing said optimum in $\mathcal{S}_\mathcal{V}$. This yields a memoryless scheduler, which in turn can be translated into a finite memory scheduler in $\mathcal{S}$.

Note that this construction annihilates the difference between strict and large inequalities. This is of no consequence on the computation of the value of the disclosure. Indeed, if the optimal value is reached on a facet of the polytope defined through a strict inequality, one can build a scheduler, possibly with infinite memory, reaching the optimal value. In the case where the strict constraint is of the form $x_s > 0$, *i.e.*, the edge should disappear in the limit, this actually does not introduce a modal edge in the sense that the set $\mathcal{V}(\mathcal{B}, \mathcal{O}, \varphi)$ is not changed.

The number of BFSs of a system of rank r in dimension n is bounded by $\binom{n}{r}$ (the number of subsets of cardinality r in a set of n elements), hence for each state there is an exponential number of BFSs to consider. However, the set of BFSs depends only on the (polynomially many) states of $\mathcal{S}$ and not on the (exponentially many) states of $\mathcal{A}_\mathcal{V}$. As a result the overall complexity of the procedure is in EXPTIME. $\square$

An example of the transformation described in the above proof is illustrated for state q_0 of the LCMC in Fig. 3. The BFSs are depicted in Fig. 8: all possible distributions are points in the red triangle with corners μ_1, μ_2, and μ_3 being the three BFSs. Note that these distributions only consider values for x_1, x_2, x_3, since all other values are null. The transformation of the LCMC (for state q_0) into an MDP is illustrated in Fig. 9.

4.3 Approximation in the general case

When a scheduler is faced with the choice to include or exclude a modal edge, it can produce several versions of PTSs, say $\mathcal{A}_1$ and $\mathcal{A}_2$, with $\mathit{Tr}(\mathcal{A}_1) \neq \mathit{Tr}(\mathcal{A}_2)$,

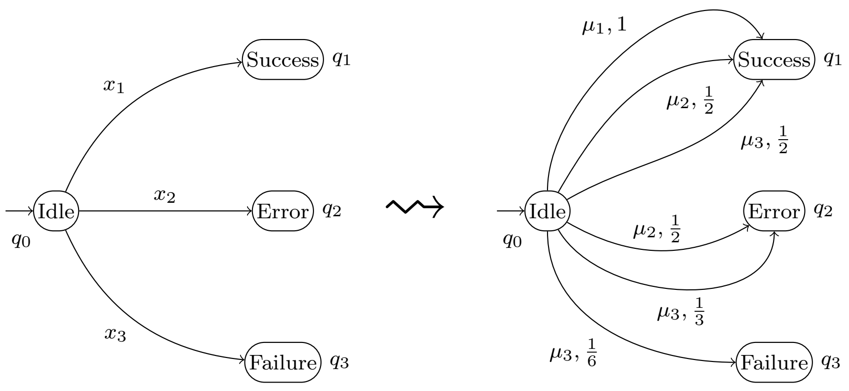


Fig. 9 Transforming an LCMC into an MDP using BFSs.

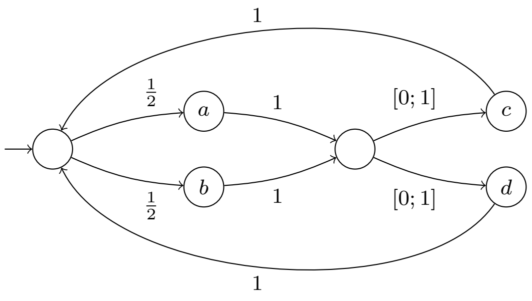


Fig. 10 IMC where the choice on modal edge requires history.

hence $\mathcal{V}(\mathcal{A}_1, \mathcal{O}, \varphi) \neq \mathcal{V}(\mathcal{A}_2, \mathcal{O}, \varphi)$. In addition, these choices may be history dependent, as in the example of Fig. 10, with $\varphi = a\Sigma^\omega$ and only letters c and d being observed. Intuitively, a way for the scheduler to always disclose the presence of an initial a is to always follow an a by the same letter, say a c. However, this choice must be made after the first letter has been seen. Moreover, leaving the possibility of a run $ad\cdots$ to occur means that run $ac\cdots$ does not disclose φ. As a result, the scheduler should also take into account φ and the observation function before committing to a choice with respect to modal edges. Note that the simpler model of UMC does not allow the scheduler to change according to history.

So far, the general case of modal LCMCs remains open and we conjecture that the computation is not possible in general. However, we now propose an approximation scheme using finite memory schedulers.

In the case of modal LCMCs, disclosure can be approximated by computing only what can be disclosed by an adversary with bounded memory. Increasing the allotted memory provides a better approximation of the disclosure, although there is no guarantee that the disclosure can be achieved with finite memory. A finite memory adversary can be defined as follows:

Definition 10 (n-memory scheduler) Let $[n]$ denote the set $\{1, 2, \dots, n\}$. An n-memory scheduler A for an LCMC specification $\mathcal{S} = (S, s_{init}, T, \lambda)$, is a tuple $A = ([n], i_{init}, \theta, \gamma)$ where $[n]$ is the set of modes, i_{init} is the starting mode, $\theta : [n] \times S \to [n]$ is a mode transition function and $\gamma : [n] \times S \to \mathcal{D}ist(S)$, is the choice function with $\gamma(i, s) \in T(s)$ for all $i \in [n]$ and all $s \in S$.

Scheduling $\mathcal{S}$ with A produces a PTS $\mathcal{S}(A)$ where the set of states is $[n] \times S$, the initial state is (i_{init}, s_{init}), for $(i, s) \in [n] \times S$, $L(i, s) \in \lambda(s)$ and $\theta(i, s)(i', s') = \gamma(i, s)(s')$ for $i' = \theta(i, s)$. We denote by $Sched_n(\mathcal{S})$ the set of n-memory schedulers for $\mathcal{S}$, with $\underline{sat}_n(\mathcal{S}) = \{\mathcal{S}(A) \mid A \in Sched_n(\mathcal{S})\}$ and

$$Disc_n(\mathcal{S}, \mathcal{O}, \varphi) = sup_{A \in Sched_n(\mathcal{S})} Disc(\mathcal{S}(A), \mathcal{O}, \varphi).$$

Memoryless schedulers are those in $Sched_1(\mathcal{S})$ and the set of finite memory schedulers is $\bigcup_n Sched_n(\mathcal{S})$ [2].

The computation of the disclosure of an LCMC $\mathcal{S}$ under bounded memory adversaries relies on:

- the computations of the set of modal edges of $\mathcal{S}$ (Proposition 2);
- the value of disclosure for LCMCs without modal edges (Theorem 1);
- the *unwinding* of $\mathcal{S}$ with respect to a memory bound and the removal of modal edges (described below).

The unwinding construction. Given $\mathcal{S} = (S, s_{init}, T, \lambda)$ and a finite transition system $\mathcal{A}_{n,\theta} = ([n], i_{init}, \theta)$, we construct the LCMC $\mathcal{A}_{n,\theta} \times \mathcal{S} = ([n] \times S, (i_{init}, s_{init}), \hat{T}, \hat{\lambda})$ s.t.

- $\hat{\mu} \in \hat{T}(i, s)$ iff $\exists \mu \in T(s)$ such that for $i' \in [n]$,

$$\hat{\mu}(i', s') = \begin{cases} \mu(s') \text{ if } i' = \theta(i, s) \\ 0 \text{ otherwise} \end{cases}$$

- $\hat{\lambda}(i, s) = \lambda(s)$

The unwinding of $\mathcal{S}$ by an n-state automaton $\mathcal{A}$ produces an LCMC $\mathcal{S}_{\mathcal{A}}$ formed with n copies of $\mathcal{S}$ (one for each state of $\mathcal{A}$), communicating with each other according to the mode transition function of $\mathcal{A}$. Memoryless schedulers on $\mathcal{S}_{\mathcal{A}}$ then correspond to n-memory schedulers on $\mathcal{S}$. We note $Sched_{\mathcal{A}_{n,\theta}}(\mathcal{S})$ the set of schedulers for $\mathcal{S}$ of the form $A = (\mathcal{A}_{n,\theta}, \gamma)$ for $\gamma : [n] \times S \to \mathcal{D}ist(S)$ where for $s \in S$ and for all $i \in [n]$, $\gamma(i, s) \in T(s)$. Hence, any implementation of $\underline{sat}_n(\mathcal{S})$ can be obtained by a memoryless scheduler on some $\mathcal{S}_{\mathcal{A}}$. Remark that there is only a finite number of such automata. This construction entails:

Lemma 1 *For any LCMC $\mathcal{S}$,*

$$\underline{sat}_1(\mathcal{A}_{n,\theta} \times \mathcal{S}) = \{\mathcal{S}(A) \mid A = (\mathcal{A}_{n,\theta}, \gamma) \textit{ for some } \gamma\}.$$

The modal edge removing construction. Given $\mathcal{S} = (S, s_{init}, T, \lambda)$, an LCMC, let $\mathcal{M}(\mathcal{S})$ be the set of modal edges of $\mathcal{S}$. For a subset $\xi \subseteq \mathcal{M}(\mathcal{S})$, we define $\mathcal{S} \setminus \xi$ as the modal edge free LCMC obtained from $\mathcal{S}$ by removing edges in ξ and "*unmodalizing*" modal edges of $\mathcal{S}$ not in ξ: this means that if $(s, s') \in \xi$, we add the constraint $x_{s'} = 0$ to $T(s)$ and if $(s, s') \notin \xi$, we add the constraint $x_{s'} > 0$ to $T(s)$. Note that removing an arbitrary subset ξ of $\mathcal{M}(\mathcal{S})$ may result in an inconsistent specification. These will be ignored by our algorithm (recall that consistency checks are polynomial for LCMCs [13]).

The algorithm. Algorithm 1 enumerates all possible finite transition systems with n states and unwinds $\mathcal{S}$ over them. For each such unwinding, the algorithm explores the set of memoryless schedulers. This exploration is done by first selecting the set of modal edges to remove and for each set, compute the maximal disclosure using the procedure of Theorem 1. Since this procedure may use arbitrary schedulers, the output of the algorithm is an overapproximation of $Disc_n(\mathcal{S}, \mathcal{O}, \varphi)$, the disclosure of $\mathcal{S}$ restricted to n-memory schedulers. On the other hand, it is an underapproximation of the actual disclosure.

ALGORITHM 1: Overapproximating $Disc_n(\mathcal{S}, \mathcal{O}, \varphi)$

```
   Input: $n \in \mathbb{N}, \mathcal{S}, \mathcal{O}$ and $\varphi$
   Output: $Disc(n, \mathcal{S}, \mathcal{O}, \varphi)$ such that $Disc_n(\mathcal{S}, \mathcal{O}, \varphi) \leq Disc(n, \mathcal{S}, \mathcal{O}, \varphi) \leq Disc(\mathcal{S}, \mathcal{O}, \varphi)$
1  $Disc(n, \mathcal{S}, \mathcal{O}, \varphi) = 0$;
2  foreach $\theta : [n] \times S \to [n]$ do
3      construct $\mathcal{A}_{n,\theta} \times \mathcal{S}$;
4      compute the set $\mathcal{M}(\mathcal{A}_{n,\theta} \times \mathcal{S})$;
       // By the procedure described in the proof of Prop. 2
5      foreach $\xi \in 2^{\mathcal{M}(\mathcal{A}_{n,\theta} \times \mathcal{S})}$ do
6          construct $(\mathcal{A}_{n,\theta} \times \mathcal{S}) \setminus \xi$;
7          if $(\mathcal{A}_{n,\theta} \times \mathcal{S}) \setminus \xi$ is consistent then
8              compute $Disc((\mathcal{A}_{n,\theta} \times \mathcal{S}) \setminus \xi, \mathcal{O}, \varphi)$;
               // By the procedure described in the proof of Thm. 1
9              $Disc(n, \mathcal{S}, \mathcal{O}, \varphi) = \max\{Disc(n, \mathcal{S}, \mathcal{O}, \varphi), Disc((\mathcal{A}_{n,\theta} \times \mathcal{S}) \setminus \xi, \mathcal{O}, \varphi)\}$;
10         end
11     end
12 end
13 return $Disc(n, \mathcal{S}, \mathcal{O}, \varphi)$;
```

Correctness. The partial correctness of Algorithm 1 relies on Lemma 1 above as well as the following observation, indicating that the choice of some set of modal edges to remove exactly corresponds to the scheduler choice:

Lemma 2 *For any LCMC $\mathcal{S}$ and for any memoryless scheduler $A = (\mathcal{A}, \gamma)$ of $\mathcal{S}$, there exists a maximal subset ξ of $\mathcal{M}(\mathcal{S})$ such that for any $(s, s') \in \mathcal{M}(\mathcal{S}), \gamma(1, s)(s') = 0$ iff $(s, s') \in \xi$.*

Proof Since the scheduler is memoryless, the choice of scheduling a modal edge to 0 must be uniformly taken along any run, hence this edge never appears in the PTS scheduled by A. This is equivalent to remove this edge from $\mathcal{S}$ so the set ξ is the collection of all such edges. □

Theorem 2 (Correctness of Algorithm 1) *Given an LCMC $\mathcal{S}$, an observation $\mathcal{O}$ and a secret φ for $\mathcal{S}$, the disclosure of φ in $\mathcal{S}$ for $\mathcal{O}$ by a polynomial size adversary (in the size of $\mathcal{S}$) can be over-approximated in EXPTIME.*

Proof We prove that the output of Algorithm 1:

$$Disc(n, \mathcal{S}, \mathcal{O}, \varphi) = \max_{\theta} \max_{\xi} Disc(\mathcal{A}_{n,\theta} \times \mathcal{S} \setminus \xi, \mathcal{O}, \varphi)$$

satisfies the post-condition $Disc_n(\mathcal{S}, \mathcal{O}, \varphi) \leq Disc(n, \mathcal{S}, \mathcal{O}, \varphi) \leq Disc(\mathcal{S}, \mathcal{O}, \varphi)$. For the first inequality, let $A = (\mathcal{A}, \gamma)$ be an n-memory scheduler for $\mathcal{S}$ and let $\mathcal{S}_{\mathcal{A}}$ be the corresponding unwinding. This unwinding is computed in the loop of Algorithm 1 at line 3. By Lemma 1 the n-memory schedulers on $\mathcal{S}$ are exactly the memoryless schedulers on $\mathcal{S}_{\mathcal{A}}$. Hence the choice function γ is memoryless on $\mathcal{A} \times \mathcal{S}$. Therefore Lemma 2 provides a set ξ of modal edges to be removed, such that $\mathcal{S}_{\mathcal{A}}$ and $\mathcal{S}_{\mathcal{A}} \setminus \xi$ coincide when scheduled by γ. The removal of this set ξ is performed in the inner loop at line 6. Hence the disclosure of $\mathcal{S}(A)$ is less than or equal to $Disc(\mathcal{S}_{\mathcal{A}} \setminus \xi, \mathcal{O}, \varphi)$ computed at line 8 (equality is not ensured since the computation of Theorem 1 does not restrict memory). This value is itself smaller than the output of the algorithm, hence $Disc_n(\mathcal{S}, \mathcal{O}, \varphi) \leq Disc(n, \mathcal{S}, \mathcal{O}, \varphi)$.

The second inequality results from the fact that all values computed by Algorithm 1 are of the form $Disc(\mathcal{S}_{\mathcal{A}} \setminus \xi, \mathcal{O}, \varphi)$. Since the maximum is obtained on a subset of schedulers, $Disc(n, \mathcal{S}, \mathcal{O}, \varphi) \leq Disc(\mathcal{S}, \mathcal{O}, \varphi)$ holds.

This algorithm makes $2^{O(n)}$ calls to the EXPTIME procedure of Theorem 1. Thus the complexity of Algorithm 1 remains in EXPTIME as long as n is polynomial in the size of $\mathcal{S}$. □

4.4 Example: the access control system

We illustrate this construction on the access control mechanism $\mathcal{S}_2$ of Fig. 6(a) with $n = 1$. Recall that the secret is $\varphi = abc^\omega$ and that only letter b is hidden.

The only modal edges are transitions stemming from state q_1 which can be blocked at will (at most two at a time). Note that if edge from q_1 to q_3 is blocked by a scheduler A, the predicate $\varphi = abc^\omega$ has probability 0 hence $Disc(\mathcal{S}_2(A), \mathcal{O}, \varphi) = 0$. In addition, if loop $q_1 \rightarrow q_1$ is not blocked, then runs abb^+c^ω, which are observed as ac^ω like abc^ω have non null probability in any scheduling. Hence φ would be opaque and $Disc(\mathcal{S}_2(A), \mathcal{O}, \varphi) = 0$. Thus there remains the case of transition $q_1 \rightarrow q_4$.

Whether it is blocked or not, computing the optimal disclosure in those non-modal LCMCs is done by synchronizing $\mathcal{S}_2$ with the DPA accepting disclosing runs, *i.e.* the single run abc^ω. In this case, the product is equivalent to having priority 0 to state q_3 and 1 to all other states. So the goal of the scheduler is to maximize the probability to reach q_3.

Computing the optimal scheduler is done by building the MDP with Basic Feasible Solutions. For example in state q_0, the two BFSs are $(q_0 \xrightarrow{0.2} q_1, q_0 \xrightarrow{0.8} q_2)$ and $(q_0 \xrightarrow{0.8} q_1, q_0 \xrightarrow{0.2} q_2)$. In the case where $q_1 \to q_4$ is not blocked, the two BFSs for state q_1 are $(q_1 \xrightarrow{1} q_3, q_1 \xrightarrow{0} q_4)$ and $(q_1 \xrightarrow{0} q_3, q_1 \xrightarrow{1} q_4)$, even though the intervals are open on 0: in this case, the solutions can be reached in the limit by a family of schedulers. Optimization on this MDP shows that the maximal probability to reach q_3 is obtained by choosing BFSs $(q_0 \xrightarrow{0.8} q_1, q_0 \xrightarrow{0.1} q_2)$ and $(q_1 \xrightarrow{1} q_3, q_1 \xrightarrow{0} q_4)$, reaching state q_3 with probability 0.8, hence $Disc(\mathcal{S}_2, \mathcal{O}, \varphi) \geq 0.8$.

Note that even though this computation was done by restricting the unwinding to a single memory state (no actual unwinding), in this example it provides the actual optimal value, since the only meaningful unwinding can be done at state q_1 but the best way for the scheduler to disclose the secret is always to block loop $q_1 \to q_1$ and maximize the probability to reach q_3. It follows that $Disc(\mathcal{S}_2, \mathcal{O}, \varphi) = 0.8$.

4.5 Uncertain Markov Chain semantics

The problem is simpler in the case of the Uncertain Markov Chain semantics. Since no memory is allowed in this semantics, no unwinding is necessary. Nonetheless, one must consider all possible sets of modal edges as in Algorithm 1. Once the set of edges is set, the optimal disclosure can be computed in a similar way as in Theorem 1. However, we add the constraints that all probabilities coming from the same edge of $\mathcal{S}$ must be equal, which is not the case in the scheduler semantics.

Corollary 1 (UMC semantics) *Given an arbitrary LCMC $\mathcal{S}$ with UMC semantics, an observation $\mathcal{O}$ and a secret φ for $\mathcal{S}$, the disclosure of φ in $\mathcal{S}$ for $\mathcal{O}$ can be computed in EXPTIME.*

5 Monotonicity of disclosure

This section is devoted to the proof of the following result, establishing monotonicity with respect to weak refinement for the disclosure over scheduled implementations:

Theorem 3 *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be LCMC specifications such that $\mathcal{S}_1 \preceq_w \mathcal{S}_2$. Then $Disc(\mathcal{S}_1, \mathcal{O}, \varphi) \leq Disc(\mathcal{S}_2, \mathcal{O}, \varphi)$.*

Since scheduling is a restrictive way to derive implementations from a specification, it is not the case in general that $\underline{sat}(\mathcal{S}_1) \subseteq \underline{sat}(\mathcal{S}_2)$: although any scheduling $\mathcal{S}_1(A_1)$ of $\mathcal{S}_1$ with A_1 is an implementation of $\mathcal{S}_2$, this implementation may not be a scheduling.

Instead, the proof builds a scheduler A_2 for $\mathcal{S}_2$, producing an implementation $\mathcal{S}_2(A_2)$ that is *refined* by $\mathcal{S}_1(A_1)$ (Theorem 4, illustrated in Fig. 11). Then,

$$\begin{tikzcd}
\mathcal{S}_1 \arrow[r, "\mathcal{R}"] & \mathcal{S}_2 \\
\mathcal{S}_1(A_1) \arrow[u, "sat_1"] \arrow[r, dashed, "\mathcal{R}'"'] \arrow[ur, dotted, "\mathcal{R} \circ sat_1"'] & \mathcal{S}_2(A_2) \arrow[u, dashed, "sat_2"']
\end{tikzcd}$$

Fig. 11 Result of Theorem 4. Relation $\mathcal{R} \circ sat_1$ always exists but might not be a scheduling.

this refinement is shown to ensure that the probabilities of (cones of) finite words coincide (Propositions 3 and 4). The disclosure set being a measurable event, coincidence of probabilities on cones ensures coincidence of probabilities for the disclosure.

Notations. Given two specifications $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $\mathcal{S}_1$ weakly refines $\mathcal{S}_2$ through relation $\mathcal{R}$, we define the relation $\sim$ on $FRuns(\mathcal{S}_1) \times FRuns(\mathcal{S}_2)$ by: $\rho_1 \sim \rho_2$ if $|\rho_1| = |\rho_2|$ and at any intermediate step i, the corresponding states satisfy $s_{1,i} \mathcal{R}\, s_{2,i}$.

For $\rho_2 \in FRuns(\mathcal{S}_2)$, we set $sim(\rho_2) = \{\rho_1 \in FRuns(\mathcal{S}_1) \mid \rho_1 \sim \rho_2\}$. We now define a measure μ_{ρ_2} over $sim(\rho_2)$ by $\mu_{\rho_2}(\rho_1) = \frac{\mathbf{P}_{\mathcal{A}_1}(\rho_1)}{\mathbf{P}_{\mathcal{A}_1}(sim(\rho_2))}$ (where the probability of finite run ρ is abusively written instead of the probability of its cone C_ρ).

We first show how to build a scheduler A_2 for $\mathcal{S}_2$ such that $\mathcal{S}_1(A_1)$ refines $\mathcal{S}_2(A_2)$. Recall from Section 3.2 that for PTSs, weak and strong refinement coincide and are thus simply called *refinement* in the remainder of this section.

Theorem 4 *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be LCMC specifications such that $\mathcal{S}_1$ weakly refines $\mathcal{S}_2$. Then for any $A_1 \in Sched(\mathcal{S}_1)$ there exists $A_2 \in Sched(\mathcal{S}_2)$ such that $\mathcal{S}_1(A_1)$ refines $\mathcal{S}_2(A_2)$.*

Proof Let $\mathcal{S}_1 = (S_1, s_{init,1}, T_1, \lambda_1)$ and $\mathcal{S}_2 = (S_2, s_{init,2}, T_2, \lambda_2)$ be LCMCs such that $\mathcal{S}_1$ refines $\mathcal{S}_2$ with $\mathcal{R}$. Let sat_1 be the satisfaction relation related to A_1 and $\mathcal{A}_1 = \mathcal{S}_1(A_1) = (Q_1, q_{init,1}, \Delta_1, L_1)$. Then we show that there exists $A_2 \in Sched(\mathcal{S}_2)$ and a refinement relation $\mathcal{R}'$ such that $\mathcal{R} \circ sat_1 = sat_2 \circ \mathcal{R}'$ where sat_2 is the satisfaction relation related to A_2.

Let $\rho_2 \in FRuns(\mathcal{S}_2)$ with $\mathrm{lst}(\rho_2) = s_2$. Then, for any $\rho_1 \in sim(\rho_2)$, $A_1(\rho_1) \in T_1(\mathrm{lst}(\rho_1))$ and $\mathrm{lst}(\rho_1) \mathcal{R}\, s_2$. Since $\mathcal{S}_1$ weakly refines $\mathcal{S}_2$, there exists $\delta_{\rho_1} : S_1 \to Dist(S_2)$ such that $\sum_{s'_1 \in S_1} A_1(\rho_1)(s'_1)\delta_{\rho_1}(s'_1) \in T_2(s_2)$.

We define A_2 on $FRuns(\mathcal{S}_2)$ by:

$$A_2(\rho_2) = \sum_{\rho_1 \in sim(\rho_2)} \mu_{\rho_2}(\rho_1) \sum_{s'_1 \in S_1} A_1(\rho_1)(s'_1) \cdot \delta_{\rho_1}(s'_1).$$

From the definition of μ_{ρ_2} and the convexity of $T_2(s_2)$, we can conclude that $A_2(\rho_2)$ also belongs to $T_2(s_2)$, hence A_2 is a scheduler of $\mathcal{S}_2$.

Writing now $\mathcal{A}_2 = \mathcal{S}_2(A_2) = (Q_2, q_{init,2}, \Delta_2, L_2)$, the relation $\mathcal{R}'$ can be defined as $\sim$ (relating runs that are similar "step by step", as defined above). To see that the conditions are satisfied, let ρ_1 and ρ_2 be runs in Q_1 and Q_2 respectively. Then the mapping $\delta' : Q_1 \to \mathcal{D}ist(Q_2)$ is obtained by:

$$\delta'(\rho_1)(\rho_2) = \begin{cases} \mu_{\rho_2}(\rho_1)\delta_{\rho_1}(\mathrm{lst}(\rho_1))(\mathrm{lst}(\rho_2)) & \text{if } \rho_1 \sim \rho_2, \\ 0 & \text{otherwise.} \end{cases}$$

Since $\mathcal{A}_1$ and $\mathcal{A}_2$ are PTSs, we just need to show that Equation (3.2) holds. Writing $\rho_2' = \rho_2 \xrightarrow{A_2(\rho_2)} s_2'$, we have:

$$\begin{aligned} \Delta_2(\rho_2)(\rho_2') &= A_2(\rho_2)(s_2') \\ &= \sum_{\rho_1 \in sim(\rho_2)} \mu_{\rho_2}(\rho_1) \sum_{s_1' \in S_1} A_1(\rho_1)(s_1') \cdot \delta_{\rho_1}(s_1')(s_2') \\ &= \sum_{\rho_1 \in sim(\rho_2)} \sum_{s_1' \in S_1} A_1(\rho_1)(s_1') \cdot \delta'(\rho_1 \xrightarrow{A(\rho_1)} s_1')(\rho_2') \\ &= \sum_{\rho_1' \in Q_1} A_1(\rho_1)(s_1') \cdot \delta'(\rho_1')(\rho_2') \end{aligned}$$

by defining $\rho_1' = \rho_1 \xrightarrow{A_1(\rho_1)} s_1'$ for each ρ_1 and remarking that $\delta' = 0$ if its arguments are not similar runs. Hence:

$$\Delta_2(\rho_2)(\rho_2') = \sum_{\rho_1' \in Q_1} \Delta_1(\rho_1)(\rho_1') \cdot \delta'(\rho_1')(\rho_2'). \qquad \square$$

Consider for example the (memoryless) scheduler for the refined access control mechanism $\mathcal{S}_1$ of Fig. 6(b) that assigns the following probabilities:

$$r_0 \xrightarrow{\frac{1}{3}} r_1 \qquad r_0 \xrightarrow{\frac{2}{3}} r_3 \qquad r_1 \xrightarrow{\frac{4}{5}} r_4 \qquad r_1 \xrightarrow{\frac{1}{5}} r_2 \qquad r_2 \xrightarrow{\frac{7}{8}} r_1 \qquad r_2 \xrightarrow{\frac{1}{8}} r_5$$

and all other probabilities being 1. The scheduled system is a PTS that is a refinement of the LCMC $\mathcal{S}_2$ (Fig. 6(a)) scheduled by the scheduler with two modes $\top$ (initial) and $\bot$ as follows:

In $\top$(or $\bot$) and q_0 : schedule($q_0 \xrightarrow{\frac{1}{3}} q_1$, $q_0 \xrightarrow{\frac{2}{3}} q_2$); go to $\top$

In $\top$ and q_1 : schedule($q_1 \xrightarrow{\frac{4}{5}} q_3$, $q_1 \xrightarrow{\frac{1}{5}} q_1$, $q_1 \xrightarrow{0} q_4$); go to $\bot$

In $\bot$ and q_1 : schedule($q_1 \xrightarrow{0} q_3$, $q_1 \xrightarrow{\frac{7}{8}} q_1$, $q_1 \xrightarrow{\frac{1}{8}} q_4$); go to $\top$

and all other probabilities being 1 and the mode is left unchanged.

Now we show that refinement between two PTSs is sufficient to compare their disclosure. Namely, we show that the probabilities of cones of words are equal in both systems. Note that although this property is well known to hold for paths, it needs to be lifted to words in order to compare disclosure.

We start by considering the sets of traces globally; although it is folklore that refinement implies trace inclusion, we provide a proof for completeness sake.

Proposition 3 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be PTSs such that $\mathcal{A}_1$ refines $\mathcal{A}_2$. Then $Tr(\mathcal{A}_1) = Tr(\mathcal{A}_2)$.*

Proof We prove the proposition by induction on a strengthened statement. Namely, we claim that for every finite run in $\mathcal{A}_1$ there exists a similar run in $\mathcal{A}_2$. Since an infinite run is the limit of the sequence of its finite prefixes, this claim is sufficient to prove the proposition. Assume by induction that the proposition holds for every word of length n. Let $w \in FTr(\mathcal{A}_1)$ of length $n+1$. We write $w = w_0 a$ for some $a \in \Sigma$. Consider a run of $\mathcal{A}_1$ that produces w. It is of the form $\rho_0 s_1'$ where $\lambda(s_1') = a$; let $s_1 = \mathrm{lst}(\rho_0)$. Let ρ_0' be a run in $\mathcal{A}_2$, similar to ρ_0, and $s_2 = \mathrm{lst}(\rho_0')$. By definition of refinement, there exists a function δ such that for any state s_2' of $\mathcal{A}_2$,

$$\Delta_2(s_2)(s_2') = \sum_{\sigma_1 \in S_1} \Delta_1(s_1)(\sigma_1) \cdot \delta(\sigma_1)(s_2').$$

Moreover, whenever $\delta(\sigma_1)(s_2') > 0$, $\lambda(s_1') = \lambda(s_2')$. Since $\delta(s_1')$ is a distribution over S_2, $\delta(s_1')(s_2') > 0$ for at least one state s_2'. Hence $\rho_0' s_2'$ is similar to ρ, which shows in particular that $w \in FTr(\mathcal{A}_2)$. □

We additionally show that probabilities coincide:

Proposition 4 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be PTSs such that $\mathcal{A}_1$ refines $\mathcal{A}_2$. Then for all $w \in \Sigma^*$, $\mathbf{P}_{\mathcal{A}_1}(C_w) = \mathbf{P}_{\mathcal{A}_2}(C_w)$.*

Since a given word may be produced by several paths, their probabilities should be considered altogether. Hence the proof of the above proposition is not immediate; it is quite technical and can be found in Appendix A.

Existing properties about refinement for PTSs can be retrieved as consequences of the above result. They were for example obtained as a particular case of sub-stochastic refinement in [3]. Although not necessary to prove the main theorem, these results illustrate how constraining refinement between PTSs is.

Recall that a probabilistic bisimulation [18] is a bisimulation that preserves transition probabilities, *i.e.*, a bisimulation relation $\mathcal{R}$ on states such that for any equivalence class R of $\mathcal{R}$, and any two related states $s\mathcal{R}s'$, $\Delta(s)(R) = \Delta(s')(R)$.

Corollary 2 ([3]) *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be PTSs such that $\mathcal{A}_1$ refines $\mathcal{A}_2$. Then there exists a probabilistic bisimulation over the union of both PTSs.*

Corollary 3 ([3]) *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be PTSs such that $\mathcal{A}_1$ refines $\mathcal{A}_2$. Then $\mathcal{A}_2$ also refines $\mathcal{A}_1$.*

We are now ready to prove Theorem 3:

Proof Let $\mathcal{A}_1 \in \underline{sat}(\mathcal{S}_1)$. By Theorem 4 there exists $\mathcal{A}_2 \in \underline{sat}(\mathcal{S}_2)$ that is refined by $\mathcal{A}_1$. By Proposition 4, $\mathbf{P}_{\mathcal{A}_1}(C_w) = \mathbf{P}_{\mathcal{A}_2}(C_w)$ for every word $w \in \mathit{FTr}(\mathcal{A}_1)$. Hence, for any ω-regular (hence measurable) language $\mathcal{L}$, one has $\mathbf{P}_{\mathcal{A}_1}(\mathcal{L}) = \mathbf{P}_{\mathcal{A}_2}(\mathcal{L})$. It is in particular the case for $\mathcal{V}(\mathcal{A}_1, \mathcal{O}, \varphi) = \mathcal{V}(\mathcal{A}_2, \mathcal{O}, \varphi)$. Therefore, $\mathit{Disc}(\mathcal{A}_1, \mathcal{O}, \varphi) = \mathit{Disc}(\mathcal{A}_2, \mathcal{O}, \varphi)$. Consequently, the theorem holds. □

The result above can now be combined with compositional results on refinement obtained in [13]. In particular, since the conjunction of two CMC specifications is the greatest lower bound with respect to weak refinement, and LCMCs are closed under conjunction, we have:

Proposition 5 *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be LCMC specifications. Then $\mathit{Disc}(\mathcal{S}_1 \wedge \mathcal{S}_2) \leq \min(\mathit{Disc}(\mathcal{S}_1), \mathit{Disc}(\mathcal{S}_2))$.*

6 Conclusion

This work investigates how refinement between probabilistic models impacts the security, modeled as opacity, showing that disclosure is monotonic with respect to refinement when implementations are produced by an adversary through scheduling. We provide EXPTIME procedures to compute (1) the worst-case disclosure for a subclass of LCMCs and (2) lower bounds on disclosure for all LCMCs when restricted to polynomial size memory schedulers (in the size of the LCMC).

Unfortunately, we conjecture that the disclosure cannot be computed in general for LCMCs with modal edges.

While we considered here only the *worst case scenario*, it would be interesting to handle also the best case, thus providing bounds on the disclosure of all possible implementations. We also plan to extend our results to larger sub-classes of CMCs or Parametric IMCs from [13,17]. In particular, a question that was left aside is the parallel composition of LCMC, which generates polynomial and possibly non-convex constraints on distributions. Studying the effect of disclosure with respect to this operation would be an interesting (but difficult) task.

References

1. Alur, R., Černý, P., Zdancewic, S.: Preserving secrecy under refinement. In: Proc. ICALP'06, *LNCS*, vol. 4052, pp. 107–118. Springer (2006)
2. Baier, C., Katoen, J.P.: Principles of Model Checking (Representation and Mind Series). The MIT Press (2008)
3. Baier, C., Katoen, J.P., Hermanns, H., Wolf, V.: Comparative branching-time semantics for Markov chains. Information and Computation **200**, 149–214 (2005)
4. Benedikt, M., Lenhardt, R., Worrell, J.: LTL model checking of interval Markov chains. In: Proc. TACAS'13, *LNCS*, vol. 7795, pp. 32–46. Springer (2013)

5. Bérard, B., Chatterjee, K., Sznajder, N.: Probabilistic Opacity for Markov decision processes. Information Processing Letters **115**(1), 52–59 (2015)
6. Bérard, B., Kouchnarenko, O., Mullins, J., Sassolas, M.: Preserving opacity on interval Markov chains under simulation. In: C.G. Cassandras, A. Giua, Z. Li (eds.) Proc. of 13th International Workshop on Discrete Event Systems, WODES'16, pp. 319–324. IEEE (2016)
7. Bérard, B., Mullins, J., Sassolas, M.: Quantifying opacity. In: G. Ciardo, R. Segala (eds.) Proc. QEST'10, pp. 263–272. IEEE Computer Society (2010)
8. Bérard, B., Mullins, J., Sassolas, M.: Quantifying opacity. Mathematical Structures in Computer Science **25**(2), 361–403 (2015)
9. Bhargava, M., Palamidessi, C.: Probabilistic anonymity. In: M. Abadi, L. de Alfaro (eds.) Proc. CONCUR'05, *LNCS*, vol. 3653, pp. 171–185 (2005)
10. Billingsley, P.: Probability and Measure, 3rd edn. Wiley (1995)
11. Biondi, F., Legay, A., Nielsen, B.F., Wąsowski, A.: Maximizing entropy over Markov processes. Journal of Logical and Algebraic Methods in Programming **83**(5–6), 384–399 (2014)
12. Bryans, J.W., Koutny, M., Mazaré, L., Ryan, P.Y.A.: Opacity generalised to transition systems. Intl. Journal of Information Security **7**(6), 421–435 (2008)
13. Caillaud, B., Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wasowski, A.: Constraint Markov chains. Theoretical Computer Science **412**(34), 4373–4404 (2011)
14. Chatterjee, K., Henzinger, T., Sen, K.: Model-checking omega-regular properties of interval Markov chains. In: R.M. Amadio (ed.) Proc. FoSSaCS'08, pp. 302–317 (2008)
15. Chaum, D.: The dining cryptographers problem: unconditional sender and recipient untraceability. Journal of Cryptology **1**, 65–75 (1988)
16. Clarkson, M.R., Schneider, F.B.: Hyperproperties. Journal of Computer Security **18**(6), 1157–1210 (2010)
17. Delahaye, B.: Consistency for parametric interval Markov chains. In: É. André, G. Frehse (eds.) Proc. SynCoP'15, *OASICS*, vol. 44, pp. 17–32. Schloss Dagstuhl - LZI (2015)
18. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: Proc. LICS'91, pp. 266–277. IEEE Computer Society (1991)
19. Mazaré, L.: Decidability of opacity with non-atomic keys. In: Proc. FAST'04, *Intl. Federation for Information Processing*, vol. 173, pp. 71–84. Springer (2005)
20. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. Logical Methods in Computer Science **3**(3) (2007)
21. Roos, C., Terlaky, T., Vial, J.P.: Theory and Algorithms for Linear Optimization. An Interior Point Approach. Wiley-Interscience, John Wiley & Sons Ltd (1997)
22. Saboori, A., Hadjicostis, C.N.: Current-state opacity formulations in probabilistic finite automata. IEEE Transactions on Automatic Control **59**(1), 120–133 (2014)
23. Segala, R.: Modeling and Verification of Randomized Distributed Real-Time Systems. Ph.D. thesis, MIT, Department of Electrical Engineering and Computer Science (1995)
24. Sen, K., Viswanathan, M., Agha, G.: Model-checking Markov chains in the presence of uncertainties. In: H. Hermanns, J. Palsberg (eds.) Proc. of 12th Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'06, *LNCS*, vol. 3920, pp. 394–410. Springer (2006)
25. Vardi, M.Y.: Automatic verification of probabilistic concurrent finite-state programs. In: Proc. 26th Annual Symposium on Foundations of Computer Science (FOCS'85), pp. 327–338. IEEE Computer Society (1985)

A Proof of Proposition 4

Assume by induction that the proposition holds for every word of length n. Let $w \in FTr(\mathcal{A}_1) = FTr(\mathcal{A}_2)$ (recall Proposition 3) of length $n+1$ with $w = w_0 a$ for some $a \in \Sigma$. A run ρ' of $\mathcal{A}_2$ that produces w can be assumed to be of the form $\rho' = \rho'_0 s'_2$ with $\mathrm{tr}(\rho'_0) = w_0$ and $\lambda(s'_2) = a$. Then $\mathbf{P}_{\mathcal{A}_2}(C_{\rho'}) = \mathbf{P}_{\mathcal{A}_2}(C_{\rho'_0})\Delta_2(s_2)(s'_2)$ where $s_2 = \mathrm{lst}(\rho'_0)$ and hence

$$\begin{aligned}\mathbf{P}_{\mathcal{A}_2}(C_w) &= \sum_{\substack{\rho' \in FRuns(\mathcal{A}_2)\\ \mathrm{tr}(\rho')=w}} \mathbf{P}_{\mathcal{A}_2}(C_{\rho'})\\ &= \sum_{s_2, s'_2 \in S_2} \sum_{\substack{\rho'_0 \in FRuns(\mathcal{A}_2)\\ \mathrm{tr}(\rho'_0)=w_0,\\ \mathrm{lst}(\rho'_0)=s_2, \mathrm{lst}(\rho')=s'_2}} \mathbf{P}_{\mathcal{A}_2}(C_{\rho'_0}) \cdot \Delta_2(s_2)(s'_2)\end{aligned}$$

Now let $\mathcal{A}_1$ s.t. $\mathcal{A}_2$ simulates $\mathcal{A}_1$ then, as

$$\sum_{\substack{\rho_0 \in FRuns(\mathcal{A}_1)\\ \rho_0 \sim \rho'_0\\ \mathrm{lst}(\rho_0)=s_1}} \mu_{\rho'_0}(\rho_0) = \sum_{s_1 \in S_1} \sum_{\substack{\rho_0 \in FRuns(\mathcal{A}_1)\\ \rho_0 \sim \rho'_0\\ \mathrm{lst}(\rho_0)=s_1}} \mu_{\rho'_0}(\rho_0) = 1$$

we get

$$\begin{aligned}\mathbf{P}_{\mathcal{A}_2}(C_w) &= \sum_{s_2, s'_2 \in S_2} \sum_{\substack{\rho'_0 \in FRuns(\mathcal{A}_2)\\ \mathrm{tr}(\rho'_0)=w_0,\\ \mathrm{lst}(\rho'_0)=s_2, \mathrm{lst}(\rho')=s'_2}} \mathbf{P}_{\mathcal{A}_2}(C_{\rho'_0}) \cdot \Delta_2(s_2)(s'_2) \cdot \sum_{s_1 \in S_1} \sum_{\substack{\rho_0 \in FRuns(\mathcal{A}_1)\\ \rho_0 \sim \rho'_0\\ \mathrm{lst}(\rho_0)=s_1}} \mu_{\rho'_0}(\rho_0)\\ &= \sum_{\substack{s_1 \in S_1\\ s_2, s'_2 \in S_2}} \sum_{\substack{\rho'_0 \in FRuns(\mathcal{A}_2)\\ \mathrm{tr}(\rho'_0)=w_0,\\ \mathrm{lst}(\rho'_0)=s_2, \mathrm{lst}(\rho')=s'_2}} \mathbf{P}_{\mathcal{A}_2}(C_{\rho'_0}) \cdot \Delta_2(s_2)(s'_2) \sum_{\substack{\rho_0 \in FRuns(\mathcal{A}_1)\\ \rho_0 \sim \rho'_0\\ \mathrm{lst}(\rho_0)=s_1}} \mu_{\rho'_0}(\rho_0)\end{aligned}$$

As the terms are null if it is not the case that $s_1 \mathcal{R} s_2$, we have:

$$\begin{aligned}&= \sum_{\substack{s_1 \in S_1\\ s_2, s'_2 \in S_2}} \sum_{\substack{\rho'_0 \in FRuns(\mathcal{A}_2)\\ \mathrm{tr}(\rho'_0)=w_0,\\ \mathrm{lst}(\rho'_0)=s_2, \mathrm{lst}(\rho')=s'_2}} \mathbf{P}_{\mathcal{A}_2}(C_{\rho'_0}) \cdot\\ &\qquad \left(\sum_{s'_1 \in S_1} \Delta_1(s_1)(s'_1) \cdot \delta_{s_1, s_2}(s'_1)(s'_2) \sum_{\substack{\rho_0 \in FRuns(\mathcal{A}_1)\\ \rho_0 \sim \rho'_0\\ \mathrm{lst}(\rho_0)=s_1}} \mu_{\rho'_0}(\rho_0) \right)\end{aligned}$$

And as the terms are null if $\lambda(s'_1) \neq \lambda(s'_2) = \lambda(\mathrm{lst}(\rho))$, we have:

$$\begin{aligned}&= \sum_{\substack{s_1 \in S_1\\ s_2, s'_2 \in S_2}} \sum_{\substack{\rho'_0 \in FRuns(\mathcal{A}_2)\\ \mathrm{tr}(\rho'_0)=w_0,\\ \mathrm{lst}(\rho'_0)=s_2, \mathrm{lst}(\rho')=s'_2}} \mathbf{P}_{\mathcal{A}_2}(C_{\rho'_0}) \sum_{\substack{s'_1 \in S_1\\ \lambda(s_1)=\lambda(\mathrm{lst}(\rho))}} \Delta_1(s_1)(s'_1) \cdot\\ &\qquad \left(\delta_{s_1, s_2}(s'_1)(s'_2) \sum_{\substack{\rho_0 \in FRuns(\mathcal{A}_1)\\ \rho_0 \sim \rho'_0\\ \mathrm{lst}(\rho_0)=s_1}} \mu_{\rho'_0}(\rho_0) \right)\end{aligned}$$

$$\mathbf{P}_{\mathcal{A}_2}(C_w) = \sum_{s_1\in S_1} \sum_{\substack{s'_1\in S_1\\ \lambda(s_1)=\lambda(\mathrm{lst}(\rho))}} \sum_{\substack{s_2\in S_2\\ s'_2\in S_2}} \delta_{s_1,s_2}(s'_1)(s'_2)\cdot$$

$$\left(\sum_{\substack{\rho'_0\in \mathit{FRuns}(\mathcal{A}_2)\\ \mathrm{tr}(\rho'_0)=w_0,\\ \mathrm{lst}(\rho'_0)=s_2,\mathrm{lst}(\rho')=s'_2}} \mathbf{P}_{\mathcal{A}_2}(C_{\rho'_0})\Delta_1(s_1)(s'_1)\cdot \sum_{\substack{\rho_0\in \mathit{FRuns}(\mathcal{A}_1)\\ \rho_0\sim\rho'_0\\ \mathrm{lst}(\rho_0)=s_1}} \mu_{\rho'_0}(\rho_0)\right)$$

And since $\sum_{s'_2\in S_2}\delta_{s_1,s_2}(s'_1)(s'_2)=1$:

$$= \sum_{s_1\in S_1} \sum_{\substack{s'_1\in S_1\\ \lambda(s_1)=\lambda(\mathrm{lst}(\rho))}} \sum_{s_2\in S_2} \sum_{\substack{\rho'_0\in \mathit{FRuns}(\mathcal{A}_2)\\ \mathrm{tr}(\rho'_0)=w_0,\\ \mathrm{lst}(\rho'_0)=s_2}} \mathbf{P}_{\mathcal{A}_2}(C_{\rho'_0})\cdot\Delta_1(s_1)(s'_1) \sum_{\substack{\rho_0\in \mathit{FRuns}(\mathcal{A}_1)\\ \rho_0\sim\rho'_0\\ \mathrm{lst}(\rho_0)=s_1}} \mu_{\rho'_0}(\rho_0)$$

$$= \sum_{s_1\in S_1} \sum_{\substack{s'_1\in S_1\\ \lambda(s_1)=\lambda(\mathrm{lst}(\rho))}} \sum_{\substack{\rho'_0\in \mathit{FRuns}(\mathcal{A}_2)\\ \mathrm{tr}(\rho'_0)=w_0}} \mathbf{P}_{\mathcal{A}_2}(C_{\rho'_0})\cdot\Delta_1(s_1)(s') \sum_{\substack{\rho_0\in \mathit{FRuns}(\mathcal{A}_1)\\ \rho_0\sim\rho'_0\\ \mathrm{lst}(\rho_0)=s_1}} \mu_{\rho'_0}(\rho_0)$$

$$= \sum_{s_1\in S_1} \sum_{\substack{s'_1\in S_1\\ \lambda(s_1)=\lambda(\mathrm{lst}(\rho))}} \Delta_1(s_1)(s'_1)\cdot \sum_{\substack{\rho'_0\in \mathit{FRuns}(\mathcal{A}_2)\\ \mathrm{tr}(\rho'_0)=w_0}} \sum_{\substack{\rho_0\in \mathit{FRuns}(\mathcal{A}_1)\\ \rho_0\sim\rho'_0\\ \mathrm{lst}(\rho_0)=s_1}} \mathbf{P}_{\mathcal{A}_2}(C_{\rho'_0})\mu_{\rho'_0}(\rho_0)$$

$$= \sum_{s_1\in S_1} \sum_{\substack{s'_1\in S_1\\ \lambda(s_1)=\lambda(\mathrm{lst}(\rho))}} \Delta_1(s_1)(s'_1) \sum_{\substack{\rho'_0\in \mathit{FRuns}(\mathcal{A}_2)\\ \mathrm{tr}(\rho'_0)=w_0}} \cdot \sum_{\substack{\rho_0\in \mathit{FRuns}(\mathcal{A}_1)\\ \rho_0\sim\rho'_0\\ \mathrm{lst}(\rho_0)=s_1}} \mathbf{P}_{\mathcal{A}_2}(C_{\rho'_0})\frac{\mathbf{P}_{\mathcal{A}_1}(C_{\rho_0})}{\mathbf{P}_{\mathcal{A}_1}(sim(\rho'_0))}$$

$$= \sum_{s_1\in S_1} \sum_{\substack{s'_1\in S_1\\ \lambda(s_1)=\lambda(\mathrm{lst}(\rho))}} \Delta_1(s_1)(s'_1) \sum_{\substack{\rho_0\in \mathit{FRuns}(\mathcal{A}_1)\\ \mathrm{tr}(\rho_0)=w_0\\ \mathrm{lst}(\rho_0)=s_1}} \frac{\mathbf{P}_{\mathcal{A}_1}(C_{\rho_0})}{\mathbf{P}_{\mathcal{A}_1}(sim(\rho'_0))} \sum_{\substack{\rho'_0\in \mathit{FRuns}(\mathcal{A}_2)\\ \mathrm{tr}(\rho'_0)=w_0}} \mathbf{P}_{\mathcal{A}_2}(C_{\rho'_0})$$

$$= \sum_{s_1\in S_1} \sum_{\substack{s'_1\in S_1\\ \lambda(s_1)=\lambda(\mathrm{lst}(\rho))}} \Delta_1(s_1)(s'_1)\cdot \sum_{\substack{\rho_0\in \mathit{FRuns}(\mathcal{A}_1)\\ \mathrm{tr}(\rho_0)=w_0\\ \mathrm{lst}(\rho_0)=s_1}} \frac{\mathbf{P}_{\mathcal{A}_1}(C_{\rho_0})}{\mathbf{P}_{\mathcal{A}_1}(sim(\rho'_0))}\cdot\mathbf{P}_{\mathcal{A}_2}(C_{w_0})$$

and by induction hypothesis, $\mathbf{P}_{\mathcal{A}_2}(C_{w_0})=\mathbf{P}_{\mathcal{A}_1}(C_{w_0})$, hence:

$$= \sum_{s_1\in S_1} \sum_{\substack{s'_1\in S_1\\ \lambda(s_1)=\lambda(\mathrm{lst}(\rho))}} \Delta_1(s_1)(s'_1)\cdot \sum_{\substack{\rho_0\in \mathit{FRuns}(\mathcal{A}_1)\\ \mathrm{tr}(\rho_0)=w_0\\ \mathrm{lst}(\rho_0)=s_1}} \frac{\mathbf{P}_{\mathcal{A}_1}(C_{\rho_0})}{\mathbf{P}_{\mathcal{A}_1}(sim(\rho'_0))}\cdot\mathbf{P}_{\mathcal{A}_1}(C_{w_0})$$

and since $\mathbf{P}_{\mathcal{A}_1}(sim(\rho'_0))=\mathbf{P}_{\mathcal{A}_1}(C_{w_0})$:

$$= \sum_{s_1\in S_1} \sum_{\substack{s'_1\in S_1\\ \lambda(s_1)=\lambda(\mathrm{lst}(\rho))}} \Delta_1(s_1)(s'_1)\cdot \sum_{\substack{\rho_0\in \mathit{FRuns}(\mathcal{A}_1)\\ \mathrm{tr}(\rho_0)=w_0\\ \mathrm{lst}(\rho_0)=s_1}} \mathbf{P}_{\mathcal{A}_1}(C_{\rho_0})$$

$$= \sum_{s_1\in S_1} \sum_{\substack{\rho_0\in \mathit{FRuns}(\mathcal{A}_1)\\ \mathrm{tr}(\rho_0)=w_0\\ \mathrm{lst}(\rho_0)=s_1}} \mathbf{P}_{\mathcal{A}_1}(C_{\rho_0})\cdot \sum_{\substack{s'_1\in S_1\\ \lambda(s_1)=\lambda(\mathrm{lst}(\rho))}} \Delta_1(s_1)(s'_1)$$

$$= \sum_{\substack{\rho_0\in \mathit{FRuns}(\mathcal{A}_1)\\ \mathrm{tr}(\rho_0)=w_0}} \mathbf{P}_{\mathcal{A}_1}(C_{\rho_0})\Delta_1(\mathrm{lst}(\rho_0))(\mathrm{lst}(\rho)) \;=\; \mathbf{P}_{\mathcal{A}_1}(C_w)$$