The Bright Side of Timed Opacity

Étienne André, Sarah Dépernet, Engel Lefaucheux

The Bright Side of Timed Opacity*

Étienne André[1,2], Sarah Dépernet[3,4], and Engel Lefaucheux[4]

[1] Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030, Villetaneuse, France
[2] Institut universitaire de France (IUF)
[3] Université de Bordeaux, France
[4] Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract. In 2009, Franck Cassez showed that the timed opacity problem, where an attacker can observe some actions with their timestamps and attempts to deduce information, is undecidable for timed automata (TAs). Moreover, he showed that the undecidability holds even for subclasses such as event-recording automata. In this article, we consider the same definition of opacity for several other subclasses of TAs: with restrictions on the number of clocks, of actions, on the nature of time, or on a new subclass called observable event-recording automata. We show that opacity can mostly be retrieved, except for one-action TAs and for one-clock TAs with ϵ-transitions, for which undecidability remains. We then exhibit a new decidable subclass in which the number of observations made by the attacker is limited.

Keywords: timed automata · opacity · timing attacks

1 Introduction

The notion of *opacity* [24,18] concerns information leaks from a system to an attacker; that is, it expresses the power of the attacker to deduce some secret information based on some publicly observable behaviors. If an attacker observing a subset of the actions cannot deduce whether a given sequence of actions has been performed, then the system is opaque. Time particularly influences the deductive capabilities of the attacker. It has been shown in [22] that it is possible for models that are opaque when timing constraints are omitted, to become non-opaque when those constraints are added to the models.

Timed automata (TAs) [2] are an extension of finite automata that can measure and react to the passage of time, extending traditional finite automata with the ability to handle real-time constraints. They are equipped with a finite set of clocks that can be reset and compared with integer constants, enabling the modeling and verification of real-time systems.

* This work is partially supported by ANR BisoUS (ANR-22-CE48-0012). This is the author (and extended) version of the manuscript of the same name published in the proceedings of the 25th International Conference on Formal Engineering Methods (ICFEM 2024). The final version is available at springer.com.

1.1 Related work

There are several ways to define opacity problems in TAs, depending on the power of the attacker. The common idea is to ensure that the attacker cannot deduce from the observation of a run whether it was a private or a public run. The attacker in [19] is able to observe a subset $\Sigma_o \subseteq \Sigma$ of actions with their timestamps. In this context, a timed word w is said to be opaque if there exists a public run that produces the projection of w following Σ_o as an observed timed word. In this configuration, one can consider the opacity problem consisting of determining, knowing a TA $\mathcal{A}$ and a set of timed words, whether all words in this set are opaque in $\mathcal{A}$. This problem has been shown to be undecidable for TAs [19]. This notably relates to the undecidability of timed language inclusion for TAs [2]. However, the undecidability holds in [19] even for the restricted class of event-recording automata (ERAs) [3] (a subclass of TAs), for which language inclusion is decidable. The aforementioned negative results leave hope only if the definition or the setting is changed, which was done in four main lines of work.

First, in [27,28], the input model is simplified to *real-time automata* [20], a restricted formalism compared to TAs. In this setting, (initial-state) opacity becomes decidable [27,28]. In [29], Zhang studies labeled real-timed automata (a subclass of labeled TAs); in this setting, state-based (at the initial time, the current time, etc.) opacity is proved to be decidable by extending the observer (that is, the classical powerset construction) from finite automata to labeled real-timed automata.

Second, in [5], the authors consider a time-bounded notion of the opacity of [19], where the attacker has to disclose the secret before an upper bound, using a partial observability. This can be seen as a secrecy with an *expiration date*. In addition, the analysis is carried over a time-bounded horizon. The authors prove that this problem is decidable for TAs.

Third, in [12,11], the authors present an alternative definition to Cassez's opacity by studying *execution-time opacity*: the attacker has only access to the execution time of the system, as opposed to Cassez' partial observations with some observable events (with their timestamps). In that case, most problems become decidable (see [10] for a survey). Untimed control in this setting was considered in [7], while [12,11] consider also *parametric* versions of the opacity problems, in which timing parameters [4] can be used in order to make the system execution-time opaque. Timed control in this setting was considered in [8].

Finally, a very recent paper (and written concurrently) [6] addresses opacity in the one-clock setting, with additional variants regarding current-location timed opacity and initial-location timed opacity. Our result regarding decidability over discrete time (Theorem 7) matches their result (see Remark 4)—we also provide exact complexity. Furthermore, our respective seemingly contradictory results on one-clock TAs without ε-transitions (we prove decidability, while undecidability is proved in [6]) are in fact not contradictory due to the presence of unobservable actions in [6] (see Remark 3).

Regarding non-interference for TAs, some decidability results are proved in [14,15,9], while control was considered in [16]. General security problems for TAs are surveyed in [13].

1.2 Contributions

Considering the negative results from [19] there are mainly two directions: one can consider more restrictive classes of automata, or one can limit the capabilities of the attacker—we address both directions in this work.

We address here ∃-opacity ("there exists a pair of runs, one visiting and one not visiting the private locations set, that cannot be distinguished"), weak opacity ("for any run visiting the private locations set, there is another run not visiting it and the two cannot be distinguished") and full opacity (weak opacity, but with the other direction holding as well).

Our attacker model is as follows: the attacker knows the TA modeling the system and can observe (some) actions, but never gain access to the values of the clocks, nor knows which locations are visited. Their goal is to deduce from these observations whether a private location was visited.

Our set of contributions is threefold.

Inter-reducibility Our first contribution is to prove that weak opacity and full opacity are inter-reducible. This result, interesting *per se*, also allows us to consider only one of both cases in the remainder of the paper.

Opacity in subclasses of TAs Throughout the second part of this paper (Section 5), we consider the same attacker settings as in [19] but for natural subclasses of TAs: first we deal with one-action TAs, then with one-clock TAs (both with and without ε-transitions—a mostly technical consideration which makes a difference in decidability), TAs over discrete time, and a new subclass which we call observable ERAs. Precisely, we show that:

1. The problem of ∃-opacity is decidable for general TAs and thus for all subclasses of TAs we consider as well (Section 5.1).
2. The problems of weak and full opacity are both undecidable for TAs with only one action (Section 5.2) or two clocks (Section 5.3).
3. These two problems are also undecidable for TAs with a single clock, unless we forbid ε-transitions, in which case the problems become decidable (Section 5.3).
4. These two problems are decidable for unrestricted TAs over discrete time (Section 5.4), as well as for observable ERAs (Section 5.5).

These results overall build on existing results from the literature. They however allow us to draw a clear border between decidability and undecidability. Moreover, we provide the exact complexity for most of the decidable results, which in some cases, complexify the proofs.

As a proof ingredient for Section 5.4, we also show that language inclusion is decidable for TAs over discrete time (a rather unsurprising—yet interesting—result, of which we could not find a proof in the literature).

Reducing the attacker power Then, in the third part (Section 6), we introduce a new approach in which we reduce the visibility of the attacker to a *finite* number of actions occurring at the beginning of the run, on an unrestricted TA. This models the case of an attacker with a limited attack budget, while considering the maximal class of TAs. This more elaborate result allows us to retrieve decidability.

1.3 Outline

Section 2 recalls necessary preliminaries. Section 3 defines the problems of interest. Section 4 proves inter-reducibility of weak and full opacity. Section 5 addresses opacity for subclasses of TAs, while Section 6 reduces the power of the attacker to a finite set of observations. Section 7 concludes.

2 Preliminaries

We denote by $\mathbb{N}, \mathbb{Z}, \mathbb{Q}_{\geq 0}, \mathbb{R}_{\geq 0}$ the sets of non-negative integers, integers, non-negative rationals and non-negative reals, respectively. If a and b are two integers with $a \leq b$, the set $\{a, a+1, \dots, b-1, b\}$ is denoted by $[\![a; b]\!]$.

We let $\mathbb{T}$ be the domain of the time, which will be either non-negative reals $\mathbb{R}_{\geq 0}$ (continuous-time semantics) or naturals $\mathbb{N}$ (discrete-time semantics). Unless otherwise specified, we assume $\mathbb{T} = \mathbb{R}_{\geq 0}$.

Clocks are real-valued variables that all evolve over time at the same rate. Throughout this paper, we assume a set $\mathbb{X} = \{x_1, \dots, x_H\}$ of *clocks*. A *clock valuation* is a function $\mu : \mathbb{X} \to \mathbb{T}$, assigning a non-negative value to each clock. We write $\mathbf{0}$ for the clock valuation assigning 0 to all clocks. Given a constant $d \in \mathbb{T}$, $\mu + d$ denotes the valuation s.t. $(\mu + d)(x) = \mu(x) + d$, for all $x \in \mathbb{X}$. If R is a subset of $\mathbb{X}$ and μ a clock valuation, we call *reset* of the clocks of R and denote by $[\mu]_R$ the valuation s.t. for all clock $x \in \mathbb{X}$, $[\mu]_R(x) = 0$ if $x \in R$ and $[\mu]_R(x) = \mu(x)$ otherwise.

We assume $\bowtie \in \{<, \leq, =, \geq, >\}$. A constraint C is a conjunction of inequalities over $\mathbb{X}$ of the form $x \bowtie d$, with $d \in \mathbb{Z}$. Given C, we write $\mu \models C$ if the expression obtained by replacing each x with $\mu(x)$ in C evaluates to true.

2.1 Timed Automata

A TA is a finite automaton extended with a finite set of real-valued clocks. We also add to the standard definition of TAs a special private locations set, which is then used to define the subsequent opacity concepts.

Definition 1 (TA [2]). *A TA $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$, where: 1) Σ is a finite set of actions, 2) L is a finite set of locations, $\ell_0 \in L$ is the initial location, 3) $L_{priv} \subseteq L$ is a set of private locations, $L_f \subseteq L$ is a set of final locations, 4) $\mathbb{X}$ is a finite set of clocks, 5) I is the invariant, assigning to every $\ell \in L$ a constraint $I(\ell)$ over $\mathbb{X}$ (called* invariant*), 6) E is a finite set*

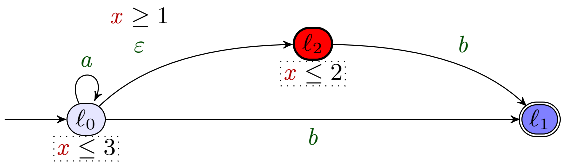


Fig. 1: A TA example

of edges $e = (\ell, g, a, R, \ell')$ where $\ell, \ell' \in L$ are the source and target locations, $a \in \Sigma \cup \{\varepsilon\}$ (where ε denotes an unobservable action), $R \subseteq \mathbb{X}$ is a set of clocks to be reset, and g is a constraint over $\mathbb{X}$ (called guard*).*

Example 1. In Fig. 1, we give an example of a TA with three locations ℓ_0, ℓ_1 and ℓ_2, three edges, two observable actions $\{a, b\}$, and one clock x. ℓ_0 is the initial location, ℓ_2 is the (unique) private location, and ℓ_1 is the (unique) final location. ℓ_0 has an invariant "$x \leq 3$" and the edge from ℓ_0 to ℓ_2 is labelled by the unobservable action ε and has a guard "$x \geq 1$".

Definition 2 (Semantics of a TA). *Given a TA $\mathcal{A} = (\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$, the semantics of $\mathcal{A}$ is given by the timed transition system $\mathfrak{T}_{\mathcal{A}} = (\mathfrak{S}, \mathfrak{s}_0, \Sigma \cup \{\varepsilon\} \cup \mathbb{R}_{\geq 0}, \rightarrow)$, with*

1. $\mathfrak{S} = \left\{(\ell, \mu) \in L \times \mathbb{R}_{\geq 0}^{\mathbb{X}} \mid \mu \models I(\ell)\right\}$, $\mathfrak{s}_0 = (\ell_0, \mathbf{0})$,
2. $\rightarrow \subseteq \mathfrak{S} \times E \times \mathfrak{S} \cup \mathfrak{S} \times \mathbb{R}_{\geq 0} \times \mathfrak{S}$ *consists of the discrete and (continuous) delay transition relations:*
   (a) *discrete transitions:* $((\ell, \mu), e, (\ell', \mu')) \in \rightarrow$, *and we write* $(\ell, \mu) \overset{e}{\mapsto} (\ell', \mu')$, *if* $(\ell, \mu), (\ell', \mu') \in \mathfrak{S}$, $e = (\ell, g, a, R, \ell') \in E$, $\mu' = [\mu]_R$, *and* $\mu \models g$.
   (b) *delay transitions:* $((\ell, \mu), d, (\ell, \mu + d)) \in \rightarrow$, *and we write* $(\ell, \mu) \overset{d}{\mapsto} (\ell, \mu + d)$, *if* $d \in \mathbb{R}_{\geq 0}$ *and* $\forall d' \in [0, d], (\ell, \mu + d') \in \mathfrak{S}$.

Moreover we write $(\ell, \mu) \xrightarrow{(d,e)} (\ell', \mu')$ for a combination of a delay and discrete transition if $\exists \mu'' : (\ell, \mu) \overset{d}{\mapsto} (\ell, \mu'') \overset{e}{\mapsto} (\ell', \mu')$.

Given a TA $\mathcal{A}$ with semantics $(\mathfrak{S}, \mathfrak{s}_0, \Sigma \cup \{\varepsilon\} \cup \mathbb{R}_{\geq 0}, \rightarrow)$, we refer to the elements of $\mathfrak{S}$ as the *configurations* of $\mathcal{A}$. A (finite) *run* of $\mathcal{A}$ is an alternating sequence of configurations of $\mathcal{A}$ and pairs of delays and edges starting from the initial configuration $\mathfrak{s}_0$ and ending in a final configuration (i.e., whose location is final), of the form $(\ell_0, \mu_0), (d_0, e_0), (\ell_1, \mu_1), \ldots (\ell_n, \mu_n)$ for some $n \in \mathbb{N}$, with $\ell_n \in L_f$ and for $i = 0, 1, \ldots n-1$, $\ell_i \notin L_f$, $e_i \in E$, $d_i \in \mathbb{R}_{\geq 0}$, and $(\ell_i, \mu_i) \xrightarrow{(d_i, e_i)} (\ell_{i+1}, \mu_{i+1})$. A *path* of $\mathcal{A}$ is a prefix of a run ending with a configuration.

2.2 Region Automaton

We recall that the region automaton is obtained by quotienting the set of clock valuations out by an equivalence relation $\simeq$ recalled below.

Given a TA $\mathcal{A}$ and its set of clocks $\mathbb{X}$, we define $M : \mathbb{X} \to \mathbb{N}$ the map that associates to a clock x the greatest value to which the interpretations of x are compared within the guards and invariants; if x appears in no constraint, we set $M(x) = 0$.

Given $\alpha \in \mathbb{R}$, we write $\lfloor \alpha \rfloor$ and $\text{frac}(\alpha)$ respectively for the integral and fractional parts of α.

Definition 3 (Equivalence relation $\simeq$ for valuations [2]). *Let μ, μ' be two clock valuations (with values in $\mathbb{R}_{\geq 0}$). We say that μ and μ' are equivalent, denoted by $\mu \simeq \mu'$ when, for each $x \in \mathbb{X}$, either $\mu(x) > M(x)$ and $\mu'(x) > M(x)$ or the three following conditions hold:*

1. $\lfloor \mu(x) \rfloor = \lfloor \mu'(x) \rfloor$;
2. *$frac(\mu(x)) = 0$ if and only if $frac(\mu'(x)) = 0$;*
3. *for each $y \in \mathbb{X}$, $frac(\mu(x)) \leq frac(\mu(y))$ if and only if $frac(\mu'(x)) \leq frac(\mu'(y))$.*

The equivalence relation is extended to the configurations of $\mathcal{A}$: let $\mathfrak{s} = (\ell, \mu)$ and $\mathfrak{s}' = (\ell', \mu')$ be two configurations in $\mathcal{A}$, then $\mathfrak{s} \simeq \mathfrak{s}'$ if and only if $\ell = \ell'$ and $\mu \simeq \mu'$.

The equivalence class of a valuation μ is denoted $[\mu]$ and is called a *clock region*, and the equivalence class of a configuration $\mathfrak{s} = (\ell, \mu)$ is denoted $[\mathfrak{s}]$ and called a *region* of $\mathcal{A}$. Clock regions are denoted by the enumeration of the constraints defining the equivalence class. Thus, values of a clock x that go beyond $M(x)$ are merged and described in the regions by "$x > M(x)$".

The set of regions of $\mathcal{A}$ is denoted by $\mathcal{R}_{\mathcal{A}}$. These regions are of finite number: this allows us to construct a finite "untimed" regular automaton, the region automaton $\mathcal{RA}_{\mathcal{A}}$. Locations of $\mathcal{RA}_{\mathcal{A}}$ are regions of $\mathcal{A}$, and the transitions of $\mathcal{RA}_{\mathcal{A}}$ convey the reachable valuations associated with each configuration in $\mathcal{A}$.

To formalize the construction, we need to transform discrete and time-elapsing transitions of $\mathcal{A}$ into transitions between the regions of $\mathcal{A}$. To do that, we define a "time-successor" relation that corresponds to time-elapsing transitions.

Definition 4 (Time-successor relation [11]). *Let $r = (\ell, [\mu]), r' = (\ell', [\mu']) \in \mathcal{R}_{\mathcal{A}}$. We say that r' is a time-successor of r when $r \neq r'$, $\ell = \ell'$ and for each configuration (ℓ, μ) in r, there exists $d \in \mathbb{R}_{\geq 0}$ such that $(\ell, \mu + d)$ is in r' and for all $d' < d, (\ell, \mu + d') \in r \cup r'$.*

A region $r = (\ell, [\mu])$ is *unbounded* when, for all x in $\mathbb{X}$ and all $\mu' \in [\mu]$, $\mu'(x) > M(x)$.

Definition 5 (Region automaton [2]). *Given a TA $\mathcal{A} = (\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$, the region automaton is the tuple $\mathcal{RA}_{\mathcal{A}} = (\Sigma_{\mathcal{R}}, \mathcal{R}, r_0, \mathcal{R}_f, E_{\mathcal{R}})$ where 1) $\Sigma_{\mathcal{R}} = \Sigma \cup \{\varepsilon\}$; 2) $\mathcal{R} = \mathcal{R}_{\mathcal{A}}$; 3) $r_0 = [\mathfrak{s}_0]$; 4) $\mathcal{R}_f$ is the set of regions whose first component is a final location $\ell_f \in L_f$; 5) i)* (discrete transitions) *For every $r = (\ell, [\mu])$ with $\ell \notin L_f$, $r' = (\ell', [\mu']) \in \mathcal{R}_{\mathcal{A}}$ and $a \in \Sigma \cup \{\varepsilon\}$:*

$$(r, a, r') \in E_{\mathcal{R}} \text{ if } \exists \mu'' \in [\mu], \exists \mu''' \in [\mu'], (\ell, \mu'') \overset{e}{\mapsto} (\ell', \mu''')$$

with $e = (\ell, g, a, R, \ell') \in E$.
ii) (delay transitions) *For every* $r = (\ell, [\mu])$ *with* $\ell \notin L_f$, $r' \in \mathcal{R}_{\mathcal{A}}$*:*

$$(r, \varepsilon, r') \in E_{\mathcal{R}} \text{ if } r' \text{ is a time-successor of } r \text{ or if } r = r' \text{ is unbounded.}$$

As in TAs, a *run* of $\mathcal{RA}_{\mathcal{A}}$ is an alternating sequence of regions of $\mathcal{RA}_{\mathcal{A}}$ and actions starting from the initial region r_0 and ending in a final region, of the form $r_0, a_0, r_1, a_1, \dots r_{n-1}, a_{n-1}, r_n$ for some $n \in \mathbb{N}$, with $r_n \in R_f$ and for $i \in [\![0; n-1]\!]$, $r_i \notin R_f$, and $(r_i, a_i, r_{i+1}) \in E_{\mathcal{R}}$. A *path* of $\mathcal{RA}_{\mathcal{A}}$ is a prefix of a run ending with a region and the trace of a path of $\mathcal{RA}_{\mathcal{A}}$ is the sequence of actions (ε excluded) contained in this path.

3 Opacity Problems in Timed Automata

3.1 Timed Words, Private and Public Runs

Given a TA $\mathcal{A}$ and a run $\rho = (\ell_0, \mu_0), (d_0, e_0), (\ell_1, \mu_1), \dots, (\ell_n, \mu_n)$ on $\mathcal{A}$, we say that L_{priv} is *visited in* ρ if there exists $m \in \mathbb{N}$ such that $\ell_m \in L_{priv}$. We denote by $Visit^{priv}(\mathcal{A})$ the set of runs visiting L_{priv}, and refer to them as *private* runs. Conversely, we say that L_{priv} is *avoided in* ρ if the run ρ does not visit L_{priv}. We denote the set of runs avoiding L_{priv} by $Visit^{\overline{priv}}(\mathcal{A})$, referring to them as *public* runs.

A timed word is a sequence of pairs made up of an action and a timestamp in $\mathbb{R}_{\geq 0}$, with the timestamps being non-decreasing over the sequence. We denote by $TW^*(\Sigma)$ the set of all finite timed words over the alphabet Σ. A run ρ on a TA $\mathcal{A}$ defines a timed word: if ρ is of the form $(\ell_0, \mu_0), (d_0, e_0), (\ell_1, \mu_1), \dots, (\ell_n, \mu_n)$ where for each $i \in [\![0; n-1]\!]$, $e_i = (\ell_i, g_i, a_i, R_i, \ell_{i+1})$ and $a_i \in \Sigma \cup \{\varepsilon\}$, then it generates the timed word $(a_{j_0}, \sum_{i=0}^{j_0} d_i)(a_{j_1}, \sum_{i=0}^{j_1} d_i) \cdots (a_{j_m}, \sum_{i=0}^{j_m} d_i)$, where $j_0 < j_1 < \cdots < j_m$ and $\{j_k \mid k \in [\![0; m]\!]\} = \{i \in [\![0; n-1]\!] \mid a_i \neq \varepsilon\}$. We denote by $Tr(\rho)$ and call *trace* of ρ the timed word generated by the run ρ and, by extension, given a set of runs Ω, we denote by $Tr(\Omega)$ the set of the traces of runs in Ω.

The set of timed words recognized by a TA $\mathcal{A}$ is the set of traces generated by its runs, $Tr(Visit^{priv}(\mathcal{A}) \cup Visit^{\overline{priv}}(\mathcal{A}))$ (thus a subset of $(\Sigma \times \mathbb{R}_{\geq 0})^*$). To shorten these notations, we use $Tr(\mathcal{A})$ for the set of timed words recognized by $\mathcal{A}$, also called *language* of $\mathcal{A}$. Similarly, we use $Tr^{priv}(\mathcal{A}) = Tr(Visit^{priv}(\mathcal{A}))$ to denote the set of traces of private runs, and $Tr^{\overline{priv}}(\mathcal{A}) = Tr(Visit^{\overline{priv}}(\mathcal{A}))$ for the set of traces of public runs.

In Cassez's original definition [19], actions were partitioned into two sets, depending on whether an attacker could observe them or not. For simplicity, here we replaced all unobservable transition in $\mathcal{A}$ by ε-transitions. Projecting the sequence of actions in a run onto the observable actions, as done by Cassez, is equivalent to replacing these actions by ε and taking the trace of the run. Therefore, with respect to opacity, our model is equivalent to [19].

3.2 Defining Timed Opacity

In this section, a definition of timed opacity based on the one from [19] is introduced, with three variants inspired by [10]: existential, full and weak opacity. If the attacker observes a set of runs of the system (i.e., observes their associated traces), we do not want them to deduce whether L_{priv} was visited or not during these observed runs. Opacity holds when the traces can be produced by both private and public runs.

We are thus first interested in the existence of an opaque trace produced by the TA, that is, a trace that cannot allow the attacker to decide whether it was generated by a private or a public run. ∃-opacity, which can be seen as the weakest form of opacity, is useful to check if there is at least one opaque trace; if not, the system cannot be made opaque by restraining the behaviors.

Definition 6 (∃-opacity). *A TA $\mathcal{A}$ is* ∃-opaque *if* $Tr^{priv}(\mathcal{A}) \cap Tr^{\overline{priv}}(\mathcal{A}) \neq \emptyset$.

> **∃-opacity decision problem:**
> INPUT: A TA $\mathcal{A}$
> PROBLEM: Is $\mathcal{A}$ ∃-opaque?

Ideally and for a stronger security of the system, one can ask the system to be opaque *for all* possible traces of the system: a TA $\mathcal{A}$ is fully opaque whenever for any trace in $Tr(\mathcal{A})$, it is not possible to deduce whether the run that generated this trace visited L_{priv} or not. Sometimes, a weaker notion is sufficient to ensure the required security in the system, i.e., when the compromising information solely comes from the identification of the private runs.

Definition 7 (Full and weak opacity). *A TA $\mathcal{A}$ is* fully opaque *if* $Tr^{priv}(\mathcal{A}) = Tr^{\overline{priv}}(\mathcal{A})$*. A TA $\mathcal{A}$ is* weakly opaque *if* $Tr^{priv}(\mathcal{A}) \subseteq Tr^{\overline{priv}}(\mathcal{A})$.

> **Full (resp. weak) opacity decision problem:**
> INPUT: A TA $\mathcal{A}$
> PROBLEM: Is $\mathcal{A}$ fully (resp. weakly) opaque?

Example 2. The TA $\mathcal{A}$ depicted in Fig. 1 is ∃-opaque and weakly opaque but not fully opaque. Indeed,

$$Tr^{priv}(\mathcal{A}) = \big\{(a,\tau_1)\cdots(a,\tau_n)(b,\tau_{n+1}) \mid n \in \mathbb{N} \wedge \forall i \in [\![1,n]\!], \tau_i \leq \tau_{i+1} \leq 2 \wedge \tau_{n+1} \geq 1\big\}$$

$$Tr^{\overline{priv}}(\mathcal{A}) = \big\{(a,\tau_1)\cdots(a,\tau_n)(b,\tau_{n+1}) \mid n \in \mathbb{N} \wedge \forall i \in [\![1,n]\!], \tau_i \leq \tau_{i+1} \leq 3\big\}$$

This TA verifies $Tr^{priv}(\mathcal{A}) \subseteq Tr^{\overline{priv}}(\mathcal{A})$ and $Tr^{priv}(\mathcal{A}) \cap Tr^{\overline{priv}}(\mathcal{A}) \neq \emptyset$ since $(b, 1.5) \in Tr^{priv}(\mathcal{A})$.

4 Inter-reducibility of Weak and Full Opacity

In this section, we prove a new result relating weak and full opacity (Section 4.2). To this end, we first introduce in Section 4.1 a construction—that will also be useful to prove our subsequent results in Sections 5 and 6.

4.1 $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$

First, we need a construction of two TAs $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$ that recognize timed words produced respectively by private and public runs of a given TA $\mathcal{A}$.

The public runs TA $\mathcal{A}_{pub}$ is the easiest to build: it suffices to remove the private locations from $\mathcal{A}$ to eliminate every private run in the system. (See formal definition in Definition 11 in Appendix A.)

The private runs TA $\mathcal{A}_{priv}$ is obtained by duplicating all locations and transitions of $\mathcal{A}$: one copy $\mathcal{A}_S$ corresponds to the paths that already visited the private locations set, and the other copy $\mathcal{A}_{\bar{S}}$ corresponds to the paths that did not (this is a usual way to encode a Boolean, here "L_{priv} was visited", in the locations of a TA). For each private location ℓ_{priv} in $\mathcal{A}$ we copy all transitions leading to the copy of ℓ_{priv} in $\mathcal{A}_{\bar{S}}$ and redirect them to the copy of ℓ_{priv} in $\mathcal{A}_S$. The initial location is the one from $\mathcal{A}_{\bar{S}}$ and the final locations are the ones from $\mathcal{A}_S$. Hence all runs need to go from $\mathcal{A}_{\bar{S}}$ to $\mathcal{A}_S$ before reaching a final location, which requires visiting a private location.

Definition 8 (Private runs TA $\mathcal{A}_{priv}$).
Let $\mathcal{A} = (\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$ be a TA. The private runs TA $\mathcal{A}_{priv} = (\Sigma, L_S \uplus L_{\bar{S}}, \ell_0^{\bar{S}}, L_{priv}^S, L_f^S, \mathbb{X}, I', E')$ is defined as follows:

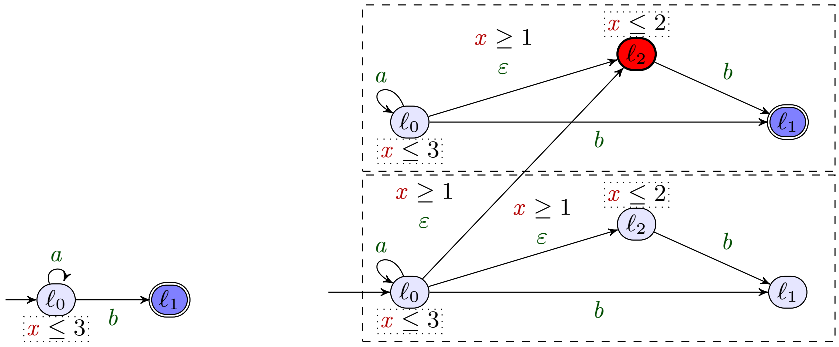
1. *$L_S = \{\ell^S \mid \ell \in L\}$ and $L_{\bar{S}} = \{\ell^{\bar{S}} \mid \ell \in L\}$.*
2. *$L_f^S = \{\ell_f^S \mid \ell_f \in L_f\}$ is the set of final locations, and $L_{priv}^S = \{\ell_{priv}^S \mid \ell_{priv} \in L_{priv}\}$ is the set of private locations;*
3. *I' is defined such as $I'(\ell^S) = I'(\ell^{\bar{S}}) = I(\ell)$*
4. *$E' = E_S \uplus E_{\bar{S}} \uplus E_{\bar{S}\to S}$ where E_S and $E_{\bar{S}}$ are the two disjoint copies of E respectively associated with the sets of locations L_S and $L_{\bar{S}}$, and $E_{\bar{S}\to S}$ is a copy of the set of all transitions that go toward $L_{priv}^{\bar{S}}$ where the target location $\ell_{priv}^{\bar{S}}$ has been changed into ℓ_{priv}^S. More formally:*

$$E_S = \big\{(\ell^S, g, a, R, \ell'^S) \mid (\ell, g, a, R, \ell') \in E\big\}$$
$$E_{\bar{S}} = \big\{(\ell^{\bar{S}}, g, a, R, \ell'^{\bar{S}}) \mid (\ell, g, a, R, \ell') \in E\big\}$$
$$E_{\bar{S}\to S} = \big\{(\ell^{\bar{S}}, g, a, R, \ell_{priv}^S) \mid (\ell, g, a, R, \ell_{priv}) \in E\big\}.$$

Example 3. We illustrate these constructions in Fig. 2 with $\mathcal{A}$ from Fig. 1.

The languages of $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$ are respectively $Tr^{priv}(\mathcal{A})$ and $Tr^{\overline{priv}}(\mathcal{A})$.

Remark 1. By a minor modification on $\mathcal{A}_{priv}$, one can build a TA $\mathcal{A}_{memo}$ that recognizes exactly the same language as $\mathcal{A}$ and that stores in each location whether the private locations set has been visited. To do so, we add the set $\{\ell_f^{\bar{S}} \mid \ell_f \in L_f\}$ to the set of final locations in $\mathcal{A}_{priv}$ and we remove each $\ell_{priv}^{\bar{S}} \in L_{priv}^{\bar{S}}$ from $L_{\bar{S}}$ in the same way as we did in $\mathcal{A}_{pub}$: the private locations of $\mathcal{A}_{memo}$ are exactly those of $\mathcal{A}_{priv}$. Notably, $\mathcal{A}$ is weakly (resp. fully) opaque if and only if $\mathcal{A}_{memo}$ is weakly (resp. fully) opaque.



(a) $\mathcal{A}_{pub}$ (b) $\mathcal{A}_{priv}$

Fig. 2: $\mathcal{A}_{pub}$ and $\mathcal{A}_{priv}$ with the example from Fig. 1

4.2 Inter-reducibility Proof

While the distinction between weak and full notions of opacity can lead to meaningful changes [10], within our framework both associated problems are inter-reducible.

Theorem 1. *The weak opacity decision problem and the full opacity decision problem are inter-reducible.*

Proof. Let us first show that the full opacity decision problem reduces to the weak opacity decision problem. Let $\mathcal{A}$ be a TA. In order to test whether $\mathcal{A}$ is fully opaque, we can test both inclusions: $Tr^{priv}(\mathcal{A}) \subseteq Tr^{\overline{priv}}(\mathcal{A})$ and $Tr^{priv}(\mathcal{A}) \supseteq Tr^{\overline{priv}}(\mathcal{A})$. The first inclusion can be decided directly by testing whether $\mathcal{A}$ is weakly opaque. In order to test the second inclusion, we need to build a TA $\mathcal{B}$ where private and public runs are inverted. To do so, we first build $\mathcal{A}_{pub}$ and $\mathcal{A}_{priv}$ and then define $\mathcal{B}$ as the TA constituted of $\mathcal{A}_{pub}$ and $\mathcal{A}_{priv}$ as well as two new locations ℓ'_0 and ℓ'_{priv}. The location ℓ'_0 is the initial location of $\mathcal{B}$ and ℓ'_{priv} is the only private location. For $x \in \mathbb{X}$, both ℓ'_0 and ℓ'_{priv} have the invariant $x = 0$, ensuring no time may elapse in those locations. From ℓ'_0, with a transition labeled by ε, one may reach either the initial location of $\mathcal{A}_{priv}$ (ℓ_0^S) or ℓ'_{priv}, from which an ε-transition leads to the initial location of $\mathcal{A}_{pub}$ (ℓ_0). The final locations of $\mathcal{B}$ are the final locations of $\mathcal{A}_{pub}$ and $\mathcal{A}_{priv}$. The public runs of $\mathcal{B}$ are the ones starting in ℓ'_0, going immediately to ℓ_0^S, and then following a run of $\mathcal{A}_{priv}$ until a final location of $\mathcal{A}_{priv}$ is reached. As the initial transition is labeled by ε, we have $Tr^{\overline{priv}}(\mathcal{B}) = Tr^{priv}(\mathcal{A})$. Similarly, the private runs of $\mathcal{B}$ are the ones starting in ℓ'_0, going immediately to ℓ'_{priv} followed immediately by going to ℓ_0^S, and then follows a run of $\mathcal{A}_{pub}$ until a final location of $\mathcal{A}_{pub}$ is reached. As the two initial transitions are labeled by ε, we have $Tr^{priv}(\mathcal{B}) = Tr^{\overline{priv}}(\mathcal{A})$. Hence, $\mathcal{A}$ is fully opaque if and only if $\mathcal{A}$ and $\mathcal{B}$ are weakly opaque.

Let us now show the converse reduction. Let $\mathcal{A}$ be a TA. We will define a TA $\mathcal{B}$ such that $\mathcal{B}$ is fully opaque if and only if $\mathcal{A}$ is weakly opaque. To do so, we want that $Tr^{\overline{priv}}(\mathcal{B}) = Tr^{\overline{priv}}(\mathcal{A})$ and $Tr^{priv}(\mathcal{B}) = Tr^{\overline{priv}}(\mathcal{A}) \cup Tr^{priv}(\mathcal{A})$. Indeed, if these equalities hold, $Tr^{\overline{priv}}(\mathcal{B}) = Tr^{priv}(\mathcal{B})$ would be equivalent to $Tr^{\overline{priv}}(\mathcal{A}) = Tr^{\overline{priv}}(\mathcal{A}) \cup Tr^{priv}(\mathcal{A})$ which holds if and only if $Tr^{priv}(\mathcal{A}) \subseteq Tr^{\overline{priv}}(\mathcal{A})$. As for the first reduction, $\mathcal{B}$ contains a copy of $\mathcal{A}_{pub}$ and $\mathcal{A}_{priv}$ as well as two new locations ℓ'_0 and ℓ'_{priv}. The location ℓ'_0 is the initial location of $\mathcal{B}$ and ℓ'_{priv} is the only private location. For $x \in \mathbb{X}$, both ℓ'_0 and ℓ'_{priv} have the invariant $x = 0$, ensuring no time may elapse in those locations. From ℓ'_0, with a transition labeled by ε, one may reach either the initial location of $\mathcal{A}_{pub}$ (ℓ_0^S) or ℓ'_{priv}, from which an ε-transition leads either to ℓ_0^S or to the initial location of $\mathcal{A}_{pub}$ (ℓ_0). The final locations of $\mathcal{B}$ are the final locations of $\mathcal{A}_{pub}$ and $\mathcal{A}_{priv}$. The public runs of $\mathcal{B}$ are the ones starting in ℓ'_0, going immediately to ℓ_0, and then following a run of $\mathcal{A}_{pub}$ until a final location of $\mathcal{A}_{pub}$ is reached. As the initial transition is labeled by ε, we have $Tr^{\overline{priv}}(\mathcal{B}) = Tr^{\overline{priv}}(\mathcal{A})$. Similarly, the private runs of $\mathcal{B}$ are the ones starting in ℓ'_0, going immediately to ℓ'_{priv} followed immediately by going to ℓ_0^S followed by a run of $\mathcal{A}_{priv}$, or to ℓ_0, followed by a run of $\mathcal{A}_{pub}$ until a final location of $\mathcal{A}_{pub}$ is reached. As the two initial transitions are labeled by ε, we have $Tr^{priv}(\mathcal{B}) = Tr^{priv}(\mathcal{A}) \cup Tr^{\overline{priv}}(\mathcal{A})$. Hence, $\mathcal{A}$ is weakly opaque if and only if $\mathcal{B}$ is fully opaque. □

5 Opacity Problems for Subclasses of Timed Automata

In this section, we consider the decidability status and complexities of the three opacity problems presented in Section 3 for several subclasses of TAs: TAs with one clock, TAs with one action, TAs under discrete time and observable ERAs. We first show the decidability of the ∃-opacity problem in the general case. Then, we focus on each class of TAs listed above to study weak and full opacity.

5.1 ∃-opacity Problem

We show here (see Appendix B) that in general the ∃-opacity problem is PSPACE-complete relying on the reachability problem in TAs, which is known to be PSPACE-complete [2] as well, even for TAs with two clocks [21]. This theorem considers multiple subclasses of TAs which we will describe more in depth in future sections.

Theorem 2. *Given a TA $\mathcal{A}$, deciding the ∃-opacity problem for $\mathcal{A}$ is* PSPACE-*complete, even when restricting $\mathcal{A}$ to be a one-action TA, discrete-time TA, an oERA*[5]*, or a single clock TA where integers appearing in guards are given in binary.*

If the number of clocks in $\mathcal{A}$ is fixed and integers appearing in guards are given in unary, the ∃-opacity problem is in NLOGSPACE.

[5] See Section 5.5.

5.2 Timed Automata with a Single Action

Recall that the universality problem consists in deciding whether a TA $\mathcal{A}$ accepts the set of all timed words. In [26], it is shown that the class of one-action TAs is one of the simplest cases for which the universality problem is undecidable among TAs. Therefore, this gives the intuition (see Appendix C for proof) that the weak and full opacity problems are undecidable as well for one-action TAs ($|\Sigma| = 1$).

Theorem 3. *The full and weak opacity problems for TAs with one action are undecidable.*

Remark 2. The problems of execution-time opacity introduced in [10] are a particular *decidable* subcase of these undecidable opacity problems with one-action TAs. Indeed, the execution time is equivalent to a *unique* timestamp associated with the last action of the system.

5.3 Timed Automata with a Single Clock

Following the same reasoning as in Section 5.2 (based on a different existing result on TAs), we show that full opacity is undecidable for one-clock TAs.

Theorem 4. *The full and weak opacity problems for one-clock TAs are undecidable.*

Proof. By reusing the same proof argument as in Theorem 3, using the fact that universality for one-clock TAs (with ε-transitions) is undecidable [1].

Without ε-transitions We now prove that the weak and full opacity problems become both decidable in the context of one-clock TAs ($|\mathbb{X}| = 1$) *without* ε-transitions, relying on the fact that the language inclusion problem for one-clock TAs without ε-transitions is decidable [26].

By definition, a TA is weakly opaque if $Tr^{priv}(\mathcal{A})$ is included in $Tr^{\overline{priv}}(\mathcal{A})$. As $Tr^{priv}(\mathcal{A})$ and $Tr^{\overline{priv}}(\mathcal{A})$ are respectively recognized by $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$, the decidability of the weak opacity problem is directly obtained from the decidability of the inclusion of two languages. Full opacity follows immediately, from the bidirectional language inclusion.

Theorem 5. *Full and weak opacity are decidable for one-clock TAs without ε-transitions.*

Note however that, while decidable, this problem cannot be effectively solved as the algorithm given by [26] is non-primitive recursive. Moreover, this bound is tight as shown in [1]. Hence, by imitating the approach of Theorem 3, one can reduce the language inclusion problem to the weak opacity, and thus show the complexity is tight for weak and full opacity as well.

Remark 3. This result might seem to contradict the result of a concurrently written paper [6] that proves undecidability of (language-based) opacity for one-clock TAs without ε-transitions—but it does not. The discrepancy comes from the fact that our attacker observes all actions (the unobservable actions are encoded into ε-transitions), while their setting considers unobservable actions—which can act as ε-transitions even in the absence of syntactic ε-transitions.

Now, due to the undecidability of language universality for TAs with at least two clocks [26, Theorem 21], we can prove the following with the same construction as in Theorem 3:

Theorem 6. *Full and weak opacity are undecidable for TAs with* ≥ 2 *clocks.*

5.4 Timed Automata over Discrete Time

In the general case, clocks are real-valued variables, with valuations thus ranging over $\mathbb{T} = \mathbb{R}_{\geq 0}$. TAs over discrete time however restrict the clock's behavior to valuations over $\mathbb{T} = \mathbb{N}$. Since the arguments used in [2] to prove the undecidability of the universality problem in TAs rely on continuous time, this proof cannot be used to establish undecidability of opacity over discrete time. In fact, relying on the region automaton (defined in Section 2.2) over discrete time and classical results on finite regular automata, we show *decidability* of the opacity problems as well as their exact complexity.

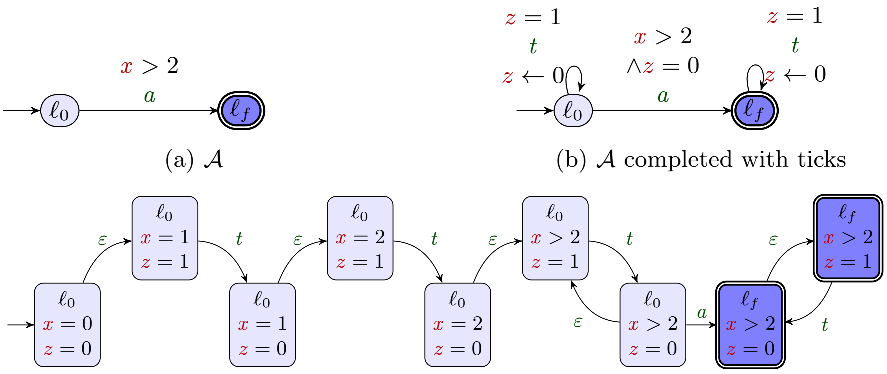
If μ, μ' are two discrete clock valuations (i.e., with values in $\mathbb{N}$), the definition of $\simeq$ from Section 2.2 can be simplified into: $\mu \simeq \mu'$ if and only if for each $x \in \mathbb{X}$, either $\mu(x) = \mu'(x)$ or $\mu(x) > M(x)$ and $\mu'(x) > M(x)$.

In continuous time, for each run of the TA, there is a unique corresponding run of the region automaton. In discrete time, thanks to the simplified form of the definition of $\simeq$, the converse statement that a run of the region automaton corresponds to a unique run of the TA nearly holds. Loss of information however remains when every clock goes beyond their maximum constant, as time elapsing is not measured beyond this point. In order to measure it, we add a letter t (for ticks) which occurs each time that an (integral) time unit passes in the region automaton. This change can be operated directly on the TA $\mathcal{A}$ so that the correspondence between paths of $\mathcal{A}$ and $\mathcal{RA}_{\mathcal{A}}$ becomes immediate.

More precisely, we add a clock z and add self-loop transitions $e_t = (\ell, (z = 1), t, \{z\}, \ell)$ on each location $\ell \in L$ of $\mathcal{A}$. We also add the guard "$z = 0$" to each discrete transition of $\mathcal{A}$.

We illustrate the resulting TA on a simple example in Fig. 3. We depict a discrete-time TA $\mathcal{A}$, its transformation by the procedure we just described and finally its region automaton $\mathcal{RA}_{\mathcal{A}}$ (over discrete time).

With this construction, time information becomes superfluous in the TA as it can be deduced from the number of ticks that were produced, which also appears within a path of the region automaton. For instance, consider the run on the $\mathcal{A}$ of Fig. 3a that remains four time units in ℓ_0 before going to ℓ_f. The timed word $(a, 4)$ on the original TA $\mathcal{A}$ becomes $(t, 1)(t, 2)(t, 3)(t, 4)(a, 4)$ in our transformed



(a) $\mathcal{A}$

(b) $\mathcal{A}$ completed with ticks

(c) $\mathcal{RA}_{\mathcal{A}}$

Fig. 3: A discrete-time region automaton example

TA. The untimed word obtained in $\mathcal{RA}_{\mathcal{A}}$ is $tttta$, which means that four ticks occurred before the action a was produced. From this information, the original timed word $(a, 4)$ can be reconstructed. In the rest of this subsection, we only consider TAs enhanced with ticks. From the previous discussion, we have (see Appendix D):

Lemma 1. *The language of a discrete-time TA and the language of its region automaton are in bijection.*

Thus, we show that the language inclusion problem for discrete-time TAs can be reduced to its decidable equivalent for finite regular automata.

Proposition 1. *Language inclusion in discrete-time TAs is* EXPSPACE-*complete.*

We can then adapt this result to the weak and full opacity problems in a similar way as done in Section 5.3.

Theorem 7. *Both weak and full opacity of discrete-time TAs are* EXPSPACE-*complete.*

Remark 4. Two very recent works [23,6] concurrently established decidability of the opacity of TAs over discrete time. Our main distinct contribution lies in establishing the exact complexity of the problems.

5.5 Observable Event-Recording Automata

In [19], the opacity problems were shown to be undecidable for Event-Recording Automata (ERAs) [3], a subclass of TAs where every clock x is associated with a specific event a_x and x is reset on a transition if and only if this transition is labeled by a_x. Due to this, the valuations of clocks are entirely determined by the duration since the last occurrence of the associated events. One of the main interest of ERAs is that they are determinizable [3]. This determinization is carried out through the standard subset construction.

The undecidability result from [19] on ERAs required to make the events a_x unobservable. Hence, in our framework they would be replaced by ε-transitions. We define observable ERAs (oERAs) as ERAs where the actions resetting the clocks must be observable. This means that the information required for the determinization now belongs to the trace that is observed.

Given an oERA $\mathcal{A}$, we can thus build through the subset construction a TA $Det_{\mathcal{A}}$ such that any path ρ in $\mathcal{A}$ corresponds to a path ρ_D in $Det_{\mathcal{A}}$ with the same trace and ending in a location labeled by the set of all the locations of $\mathcal{A}$ that can be reached with a run that has the same trace as ρ. This information, combined with the construction of $\mathcal{A}_{memo}$ (Remark 1) which stores in the state of the TA whether the private location was visited or not, provides the following result (see Appendix E).

Theorem 8. *Both weak and full opacity are* PSPACE*-complete for oERAs.*

6 Opacity with Limited Attacker Budget

One of the causes for the undecidability of the opacity problems in [19] stems from the unbounded memory the attacker might require to remember a run of the TA. As a consequence, one can wonder whether the opacity problems remain undecidable when the attacker performs only a *finite* number of observations. This models the case of an attacker with a limited attack budget. In this section, we prove that the weak and full opacity problems become decidable whenever, given $N \in \mathbb{N}$, the attacker only observes the first N actions (with their timestamps). To the best of our knowledge, this is *i*) the second result of the literature (after [12]) providing a decidable opacity result for the full class of TAs over dense time, and *ii*) the first result limiting the number of observations of an attacker in the context of opacity for TAs.

For instance, if $(a, 1.2)(b, 1.4)(b, 1.5)(a, 2.1)$ is the trace of a public run of the system, and $N = 2$, then the attacker only observes the trace $(a, 1.2)(b, 1.4)$. If $(a, 1.2)(b, 1.4)(c, 1.6)$ is the trace of a private run, the trace observed by the attacker is $(a, 1.2)(b, 1.4)$ again and the attacker cannot conclude whether a private run occurred or not.

Formally, and in order to define new variants of opacity representing this framework, given a TA $\mathcal{A}$, we define a new TA (depicted in Fig. 4) which emulates the behavior of $\mathcal{A}$ up to the Nth observation. This TA is an unfolding of $\mathcal{A}$ with $N+1$ copies of $\mathcal{A}$, where ε-transitions are taken within each copy, and transitions with an observable action lead to the next copy. A run ends when either a final location or the final copy is reached.

Definition 9 (N-observation unfolding of a TA).
Let $\mathcal{A} = (\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$ *be a TA and let* $N \in \mathbb{N}$. *We call* N-*unfolding of* $\mathcal{A}$ *the TA* $Unfold_N(\mathcal{A}) = (\Sigma, L', \ell_0^0, L'_{priv}, L'_f, \mathbb{X}, I', E')$ *where*

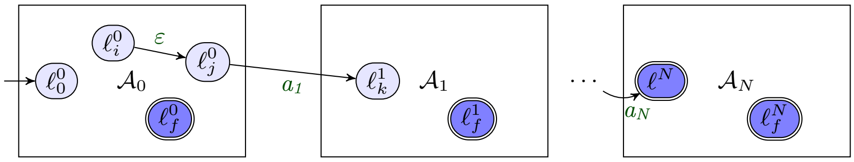


Fig. 4: The construction of an N-observation unfolded TA

1. $L' = \bigcup_{i=0}^{N} L^i$ *where the sets* L^i *are* $N+1$ *disjoint copies of* L *where each location* $\ell \in L$ *has been renamed* $\ell^i \in L^i$*: for* $0 \leq i \leq N$, $L^i = \{\ell^i \mid \ell \in L\}$;
2. $\ell_0^0 \in L^0$ *is the initial location;*
3. $L'_{priv} = \bigcup_{i=0}^{N-1} L^i_{priv}$ *where* L^i_{priv} *are the copies within* L^i *of the private locations of* $\mathcal{A}$;
4. $L'_f = (\bigcup_{i=0}^{N} L^i_f) \cup L^N$ *where* L^i_{priv} *are the copies within* L^i *of the final locations of* $\mathcal{A}$;
5. $I'(\ell^i) = I(\ell)$ *for* $l \in L$ *and* $i \leq N$ *extends* I *to each* L_i;
6. $E' = \bigcup_{i=0}^{N-1} E^i \cup E^{i \to i+1}$ *is the set of transitions where, given* $0 \leq i < N$
   - $E^i = \{(\ell^i, \varepsilon, g, R, \ell'^i) \mid (\ell, \varepsilon, g, R, \ell') \in E\}$;
   - $E^{i \to i+1} = \{(\ell^i, a, g, R, \ell'^{i+1}) \mid (\ell, a, g, R, \ell') \in E \wedge a \in \Sigma\}$.

Definition 10 (Opacity w.r.t. N observations). *Let* $\mathcal{A}$ *be a TA and let* $N \in \mathbb{N}$. *We say that* $\mathcal{A}$ *is* weakly (resp. fully, $\exists$-) opaque w.r.t. N observations *when* $\mathit{Unfold}_N(\mathcal{A})$ *is weakly (resp. fully,* $\exists$*-) opaque.*

We now state our main result. The proof is quite technical, so we only give a high-level sketch. The full proof can be found in Appendix F.

Theorem 9. *The problem of deciding, given a TA* $\mathcal{A}$ *and* $N \in \mathbb{N}$, *whether* $\mathcal{A}$ *is* $\exists$*-opaque w.r.t.* N *observations is* PSPACE*-complete.*

The problems of weak or full opacity w.r.t. N *observations are in* 2-EXPSPACE.

Proof (sketch). $\exists$-opacity can be checked in PSPACE through the same approach as Theorem 2. Indeed, even if N is given in binary, and thus $\mathit{Unfold}_N(\mathcal{A})$ is of exponential size, the region automaton of $\mathit{Unfold}_N(\mathcal{A})$ remains simply exponential in the size of $\mathcal{A}$. Hardness can be achieved with $N = 0$ with the same method as Theorem 2.

Concerning the problems of weak and full opacity w.r.t. N observations, as in Section 5.4, our goal is to rely on the region automaton to translate the opacity problems from the TA to another problem on a finite automaton. However, there is no immediate correspondence between runs of the TA and runs of the region automaton, leading to a more involved proof.

More precisely, given a TA $\mathcal{A} = (\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$ and $N \in \mathbb{N}$, we build the unfolding of the TA $\mathcal{A}_{memo}$ described in Remark 1. Recall that $\mathcal{A}_{memo}$ recognizes the same language as $\mathcal{A}$ but stores within the locations the information whether L_{priv} was visited. As such, $\mathcal{A}_{memo}$ has the same opacity properties as $\mathcal{A}$, so we can consider $\mathit{Unfold}_N(\mathcal{A}_{memo})$ instead of $\mathit{Unfold}_N(\mathcal{A})$ to study the opacity of $\mathcal{A}$.

Additionally, we enrich this TA with ticks. In Section 5.4, we added a single tick to the automaton which counted the time elapsed since the start of the run. Here, the TA includes as well, for each $0 < k \leq N$, a tick clock counting the time elapsed since the kth observation. As multiple ticks may need to occur at the same time, we develop the alphabet of ticks to describe the set of tick clocks that need to be reset, i.e., the tick $t_{\{k_1,\dots,k_m\}}$ is produced by the TA if for every $0 \leq i \leq m$, the k_ith observation (or the start of the run if $k_i = 0$) occurred an integer number of time units before.

Note that the addition of these ticks immediately uses the assumption that only N actions are observed.

In the new ticked automaton, we will establish a correspondence between runs of the TA and paths of the region automaton, allowing us to reduce the opacity problems to non-reachability of bad states in the determinization of the region automaton, implying decidability.

Considering the complexity, the unfolding of the TA, assuming N is in binary, is exponential in the number of states. Adding the ticks means adding an exponential number of clocks as well. Hence the region automaton is doubly exponential in the original TA, and its determinization is triply exponential. Reachability being in NLOGSPACE implies the 2-EXPSPACE algorithm.

A full proof with all technical details can be found in Appendix F. □

7 Conclusion and Perspectives

In this paper, we addressed three definitions of opacity on subclasses of TAs, to circumvent the undecidability from [19]. We first proved the inter-reducibility of weak and full opacity. Then, while undecidability remains for one-action TAs, we retrieve decidability for one-clock TAs without ε-transitions, or over discrete time, or for observable ERAs. Our result for one-clock TAs without ε-transitions is tight, since we showed that increasing the number of clocks or adding ε-transitions leads to undecidability. Finally, we studied the case of an attacker with an observational power with a limited budget, i.e., that can only perform a finite set of observations. We proved this latter case to be decidable on the full TA formalism. We summarize the results from Section 5 in Table 1.

Future work Perspectives include being able to build a controller to ensure a TA is opaque, as well as investigating parametric versions of these problems, where timing constants considered as parameters (à la [4]) can be tuned to ensure opacity.

Table 1: Summary of Section 5 ($\surd$ = decidability, $\times$ = undecidability)

<table>
<tr><th>Subclass</th><th>∃-opacity</th><th>weak opacity</th><th>full opacity</th></tr>
<tr><td>$|\Sigma| = 1$</td><td rowspan="6">$\surd$Theorem 2 (PSPACE-c)</td><td colspan="2">$\times$Theorem 3</td></tr>
<tr><td>$|\mathbb{X}| = 1$ **without ε-transitions**</td><td colspan="2">$\surd$Theorem 5 (non-primitive recursive-c)</td></tr>
<tr><td>$|\mathbb{X}| = 1$</td><td colspan="2">$\times$Theorem 4</td></tr>
<tr><td>$|\mathbb{X}| = 2$</td><td colspan="2">$\times$Theorem 6</td></tr>
<tr><td>$\mathbb{T} = \mathbb{N}$</td><td colspan="2">$\surd$Theorem 7 (EXPSPACE-c)</td></tr>
<tr><td>**oERAs**</td><td colspan="2">$\surd$ Theorem 8 (PSPACE-c)</td></tr>
</table>

Finally, our result in Section 6 considers an attacker with a fixed attack budget; an interesting future work would be to *derive* a maximum attack budget such that the system remains opaque.

References

[1] Abdulla, P.A., Deneux, J., Ouaknine, J., Quaas, K., Worrell, J.: Universality analysis for one-clock timed automata. FI **89**(4), 419–450 (2008)

[2] Alur, R., Dill, D.L.: A theory of timed automata. TCS **126**(2), 183–235 (Apr 1994). https://doi.org/10.1016/0304-3975(94)90010-8

[3] Alur, R., Fix, L., Henzinger, T.A.: Event-clock automata: A determinizable class of timed automata. TCS **211**(1-2), 253–273 (1999). https://doi.org/10.1016/S0304-3975(97)00173-4

[4] Alur, R., Henzinger, T.A., Vardi, M.Y.: Parametric real-time reasoning. In: Kosaraju, S.R., Johnson, D.S., Aggarwal, A. (eds.) STOC. pp. 592–601. ACM, New York, NY, USA (1993). https://doi.org/10.1145/167088.167242

[5] Ammar, I., El Touati, Y., Yeddes, M., Mullins, J.: Bounded opacity for timed systems. JISA **61**, 1–13 (Sep 2021). https://doi.org/10.1016/j.jisa.2021.102926

[6] An, J., Gao, Q., Wang, L., Zhan, N., Hasuo, I.: The opacity of timed automata. In: Platzer, A., Rozier, K.Y., Pradella, M., Rossi, M. (eds.) FM. LNCS, vol. 14933, pp. 620–637. Springer (2024). https://doi.org/10.1007/978-3-031-71162-6_32

[7] André, É., Bolat, S., Lefaucheux, E., Marinho, D.: strategFTO: Untimed control for timed opacity. In: Artho, C., Ölveczky, P. (eds.) FTSCS. pp. 27–33. ACM (2022). https://doi.org/10.1145/3563822.3568013

[8] André, É., Duflot-Krémer, M., Laversa, L., Lefaucheux, E.: Execution-time opacity control for timed automata. In: Madeira, A., Knapp, A. (eds.) SEFM (2024), to appear

[9] André, É., Kryukov, A.: Parametric non-interference in timed automata. In: Li, Y., Liew, A. (eds.) ICECCS. pp. 37–42 (2020). https://doi.org/10.1109/ICECCS51672.2020.00012

[10] André, É., Lefaucheux, E., Lime, D., Marinho, D., Sun, J.: Configuring timing parameters to ensure execution-time opacity in timed automata. In: ter Beek, M.H., Dubslaff, C. (eds.) TiCSA. Electronic Proceedings in Theoretical Computer Science, vol. 392, pp. 1–26 (2023). https://doi.org/10.4204/EPTCS.392.1, invited paper.

[11] André, É., Lefaucheux, E., Marinho, D.: Expiring opacity problems in parametric timed automata. In: Ait-Ameur, Y., Khendek, F. (eds.) ICECCS. pp. 89–98 (2023). https://doi.org/10.1109/ICECCS59891.2023.00020

[12] André, É., Lime, D., Marinho, D., Sun, J.: Guaranteeing timed opacity using parametric timed model checking. ToSEM **31**(4), 1–36 (Oct 2022). https://doi.org/10.1145/3502851

[13] Arcile, J., André, É.: Timed automata as a formalism for expressing security: A survey on theory and practice. CSUR **55**(6), 1–36 (Jul 2023). https://doi.org/10.1145/3534967

[14] Barbuti, R., Francesco, N.D., Santone, A., Tesei, L.: A notion of non-interference for timed automata. FI **51**(1-2), 1–11 (2002)

[15] Barbuti, R., Tesei, L.: A decidable notion of timed non-interference. FI **54**(2-3), 137–150 (2003)

[16] Benattar, G., Cassez, F., Lime, D., Roux, O.H.: Control and synthesis of non-interferent timed systems. IJC **88**(2), 217–236 (2015). https://doi.org/10.1080/00207179.2014.944356

[17] Bouyer, P.: Timed automata. In: Pin, J. (ed.) Handbook of Automata Theory, pp. 1261–1294. European Mathematical Society Publishing House, Zürich, Switzerland (2021). https://doi.org/10.4171/AUTOMATA-2/12, https://doi.org/10.4171/Automata-2/12

[18] Bryans, J.W., Koutny, M., Mazaré, L., Ryan, P.Y.A.: Opacity generalised to transition systems. IseCure **7**(6), 421–435 (2008). https://doi.org/10.1007/s10207-008-0058-x

[19] Cassez, F.: The dark side of timed opacity. In: Park, J.H., Chen, H., Atiquzzaman, M., Lee, C., Kim, T., Yeo, S. (eds.) ISA. LNCS, vol. 5576, pp. 21–30. Springer (2009). https://doi.org/10.1007/978-3-642-02617-1_3

[20] Dima, C.: Real-time automata. JALC **6**(1), 3–23 (2001). https://doi.org/10.25596/jalc-2001-003

[21] Fearnley, J., Jurdzinski, M.: Reachability in two-clock timed automata is PSPACE-complete. I&C **243**, 26–36 (2015). https://doi.org/10.1016/J.IC.2014.12.004

[22] Gardey, G., Mullins, J., Roux, O.H.: Non-interference control synthesis for security timed automata. ENTCS **180**(1), 35–53 (2007). https://doi.org/10.1016/j.entcs.2005.05.046

[23] Klein, J., Kogel, P., Glesner, S.: Verifying opacity of discrete-timed automata. In: Plat, N., Gnesi, S., Furia, C.A., Lopes, A. (eds.) FormaliSE. pp. 55–65. ACM (2024). https://doi.org/10.1145/3644033.3644376

[24] Mazaré, L.: Using unification for opacity properties. In: Ryan, P. (ed.) WITS. pp. 165–176 (Apr 2004)

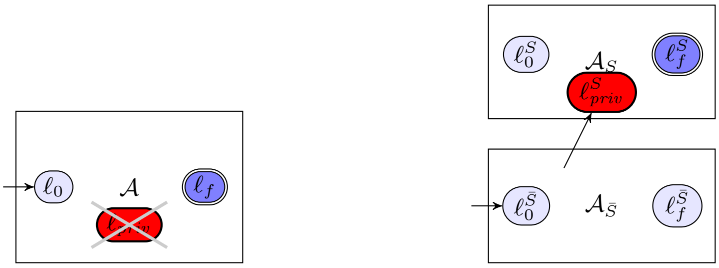
[25] Meyer, A.R., Stockmeyer, L.J.: The equivalence problem for regular expressions with squaring requires exponential space. In: IEEE. pp. 125–129 (1972), https://api.semanticscholar.org/CorpusID:206585190

[26] Ouaknine, J., Worrell, J.: On the language inclusion problem for timed automata: Closing a decidability gap. In: LICS. pp. 54–63. IEEE Computer Society (2004). https://doi.org/10.1109/LICS.2004.1319600

[27] Wang, L., Zhan, N.: Decidability of the initial-state opacity of real-time automata. In: Jones, C.B., Wang, J., Zhan, N. (eds.) Symposium on Real-Time and Hybrid Systems - Essays Dedicated to Professor Chaochen Zhou on the Occasion of His 80th Birthday, LNCS, vol. 11180, pp. 44–60. Springer (2018). https://doi.org/10.1007/978-3-030-01461-2_3

[28] Wang, L., Zhan, N., An, J.: The opacity of real-time automata. TCAD **37**(11), 2845–2856 (2018). https://doi.org/10.1109/TCAD.2018.2857363

[29] Zhang, K.: State-based opacity of labeled real-time automata. TCS **987**, 114373 (2024). https://doi.org/10.1016/J.TCS.2023.114373



(a) $\mathcal{A}_{pub}$ (b) $\mathcal{A}_{priv}$

Fig. 5: Illustrating $\mathcal{A}_{pub}$ and $\mathcal{A}_{priv}$

A Formal Definitions

Definition 11 (Public runs automaton $\mathcal{A}_{pub}$).
Let $\mathcal{A} = (\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$ be a TA. We define the public runs TA $\mathcal{A}_{pub} = (\Sigma, L \setminus L_{priv}, \emptyset, L_f \setminus L_{priv}, \mathbb{X}, I', E')$ with I' and E' precised as follows:

1. *I' is the restriction $I|_{L \setminus L_{priv}}$ of I to the set of locations of $\mathcal{A}_{pub}$;*
2. *$E' = E \setminus \{(\ell, g, a, R, \ell') \in E \mid \ell \in L_{priv} \vee \ell' \in L_{priv}\}$ is the remaining set of transitions when private locations are removed from L.*

Example 4. We illustrate the constructions of $\mathcal{A}_{pub}$ and $\mathcal{A}_{priv}$ in Figs. 5a and 5b.

B Complexity of the ∃-opacity Problem

B.1 ∃-opacity Problem for General TAs

Let us first show that the ∃-opacity problem for TA lies in PSPACE.

Proof. Let $\mathcal{A}$ be a TA. We build $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$ from $\mathcal{A}$ as described in Section 4.1. Noting that the product of two TAs recognizes the intersection of their languages [2, Theorem 3.15] (assuming the two TAs share no clock), we build the TA $\mathcal{A}_{priv} \times \mathcal{A}_{pub}$, product of $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$, which language is $Tr^{priv}(\mathcal{A}) \cap Tr^{\overline{priv}}(\mathcal{A})$. To build this product, we can rename all clocks from $\mathcal{A}_{pub}$ so that $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$ share no clock.

The ∃-opacity problem is by definition the non-emptiness of the intersection of $Tr^{priv}(\mathcal{A})$ and $Tr^{\overline{priv}}(\mathcal{A})$. Moreover, the reachability of a final location of $\mathcal{A}_{priv} \times \mathcal{A}_{pub}$ is equivalent to the non-emptiness of the language of $\mathcal{A}_{priv} \times \mathcal{A}_{pub}$, and thus of the set $Tr^{priv}(\mathcal{A}) \cap Tr^{\overline{priv}}(\mathcal{A})$. Since reachability is decidable in PSPACE in TAs [2], the same holds for the ∃-opacity problem. □

We now reduce the reachability problem for timed automata, known to be PSPACE-complete, to the ∃-opacity problem.

Proof. Let $\mathcal{A} = (\Sigma, L, \ell_0, \emptyset, L_f, \mathbb{X}, I, E)$ be a timed automaton. We suppose that $\mathbb{X}$ is not empty and define (see Fig. 6) $\mathcal{A}' = \left(\Sigma, L \cup \left\{\ell'_0, \ell'_1, \ell'_f\right\}, \ell'_0, L_f, L_f \cup \left\{\ell'_f\right\}, \mathbb{X}, I', E'\right)$ where I' an invariant extending I such that $I'(\ell'_0) = I'(\ell'_1) = I'(\ell'_f) = \mathit{true}$ and $E' = E \cup \left\{(\ell'_0, \varepsilon, (x = 0), \emptyset, \ell_0), (\ell'_0, \varepsilon, (x = 0), \emptyset, \ell'_1), (\ell'_1, \varepsilon, \mathit{true}, \emptyset, \ell'_f)\right\} \cup \left\{(\ell'_f, a, \mathit{true}, \emptyset, \ell'_f) \mid a \in \Sigma\right\}$ for some $x \in \mathbb{X}$.

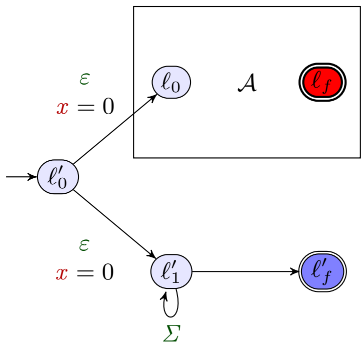


Fig. 6: TA $\mathcal{A}'$ for the PSPACE-hardness of ∃-opacity

The timed automaton $\mathcal{A}'$ is ∃-opaque if and only if a final location is reachable in $\mathcal{A}$. Indeed, the set $\mathit{Tr}^{\overline{priv}}(\mathcal{A}')$ contains all the possible timed traces with the action set Σ, and the private runs on $\mathcal{A}'$ correspond exactly to runs on $\mathcal{A}$. Hence $\mathit{Tr}^{\overline{priv}}(\mathcal{A}') \cap \mathit{Tr}^{priv}(\mathcal{A}') \neq \emptyset$ if and only if $\mathit{Tr}^{priv}(\mathcal{A}') \neq \emptyset$, i.e., if there is a run on $\mathcal{A}$ that reaches a final location. Since the reachability problem in TA is PSPACE-complete, we deduce from this construction that the ∃-opacity problem is PSPACE-hard.

Note that this reduction holds as well for one-action TAs, discrete-time TAs and oERAs.

B.2 ∃-opacity Problem for TAs with a Fixed Number of Clocks

Fix N as a constant. We consider now the ∃-opacity problem for TAs with N clocks.

In the previous section, the ∃-opacity problem for TAs was shown to be within PSPACE. The algorithm reduces the problem to a reachability query on the product automaton $\mathcal{A}_{priv} \times \mathcal{A}_{pub}$ (a TA with $2N$ clocks). The reachability problem for TAs is usually solved by studying reachability in the associated region automaton. Reachability in automata being in NLOGSPACE and the region automata being exponential in general produces the result. More precisely, the number of states of the region automata is bounded by $|L|(N! \cdot 2^N \cdot \prod_{x \in \mathbb{X}} (2M+2))$ [2] where M is the highest constant occuring in guards and invariants. Note that, since N is a constant, this number becomes polynomial when integers in guards and invariants (and thus M) are given in unary. Hence the reachability problem falls to NLOGSPACE, which implies the $\exists$-opacity problem also lies in NLOGSPACE then.

Concerning the hardness, let us show that the $\exists$-opacity problem remains PSPACE-hard for one-clock automata with constants in binary. Note that the reduction of the previous section does not apply, as reachability in TA with one clock is not PSPACE-hard. We reduce the reachability problem in two-clock automata, known to be PSPACE-complete [21], to the $\exists$-opacity problem in one-clock automata.

Let $\mathcal{A}_{x,y} = (\Sigma, L, \ell_0, \emptyset L_f, \mathbb{X}, I, E)$ a TA with clocks x and y. First we relabel every transition (including silent transitions) of $\mathcal{A}_{x,y}$ with a new alphabet $\Sigma' = \{a_i \mid 1 \leq i \leq |E|\}$ such that each letter of Σ' labels exactly one transition of $\mathcal{A}_{x,y}$. We denote the obtained automaton by $\mathcal{A}'_{x,y}$.

Given a guard g, we define g_x and g_y as respectively the constraints in g over x and y. Hence, $g = g_x \wedge g_y$. For $z \in \{x, y\}$, we then define the automaton $\mathcal{A}_z = (\Sigma, L, \ell_0, \emptyset, L_f, \{z\}, I_z, E_z)$ with $E_z = \{(\ell, a, g_z, R \cap \{z\}, \ell') \mid (\ell, a, g, R, \ell') \in E\}$ and I_z is similarly obtained by only keeping the z part of the invariant.

We have that a word accepted by $\mathcal{A}'_{x,y}$ is also accepted by $\mathcal{A}'_x$ and by $\mathcal{A}'_y$, as each of those TAs have less constraints. Moreover, if a word is accepted by $\mathcal{A}'_x$ and by $\mathcal{A}'_y$, as the corresponding run is entirely characterized by its trace (since each transition has its own label) and satisfied the constraints on both clocks, then it is accepted by $\mathcal{A}'_{x,y}$.

We build the TA $\mathcal{B}$ over the single clock x as the classical union construction of $\mathcal{A}'_x$ and $\mathcal{A}'_y$, and set as private locations the final locations of $\mathcal{A}'_x$. More precisely, we add a new initial location ℓ'_0 from which one can reach the initial location of $\mathcal{A}'_x$ and $\mathcal{A}'_y$ by a transition labelled by a new letter $\sharp$ and with the guard $(x = 0)$. Moreover, we relabel every occurrence of y in the copy of $\mathcal{A}'_y$ into x.

As the runs of $\mathcal{A}'_x$ (resp. $\mathcal{A}'_y$) provide the private (resp. public) runs of $\mathcal{B}$, $\mathcal{B}$ is $\exists$-opaque if and only if there is a pair of runs of same trace accepted by $\mathcal{A}'_x$ and $\mathcal{A}'_y$, thus a word accepted by $\mathcal{A}'_{x,y}$ or equivalently a reachable final location in $\mathcal{A}_{x,y}$. Moreover, $\mathcal{B}$ is polynomial in the size of $\mathcal{A}_{x,y}$. Therefore the $\exists$-opacity problem in one-clock automata is PSPACE-hard.

C Opacity of One-Action TAs

Proof. We first prove the undecidability of the full opacity problem. Let $\mathcal{A}$ be a TA with a single action. We want to build a TA such that if we can answer the full opacity problem of this TA, then we can decide the universality problem for $\mathcal{A}$. We consider the following TA: we add an initial location exited by two ε-transitions that must be taken urgently (i.e., no time may elapse before taking them). The first ε-transition leads to a secret location which leads (again via

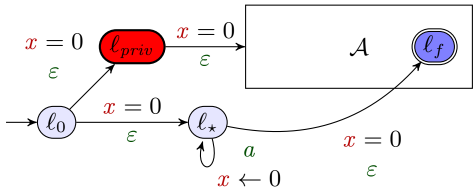


Fig. 7: Automaton $\mathcal{B}$: Reduction from universality to full opacity

an urgent ε-transition) to the initial location of the TA $\mathcal{A}$ and the other leads to a location where every finite timed words on Σ can be read before reaching a final location. We denote this TA $\mathcal{B}$ and illustrate its construction in Fig. 7. The language recognized by $\mathcal{A}$ corresponds exactly to the traces of private runs of $\mathcal{B}$, and the traces of public runs of $\mathcal{B}$ are all the finite timed words on Σ. Therefore, $\mathcal{B}$ is fully opaque if and only if $Tr^{priv}(\mathcal{B}) = Tr^{\overline{priv}}(\mathcal{B})$ if and only if $Tr(\mathcal{A}) = TW^*(\Sigma)$ if and only if $\mathcal{A}$ is universal. Since universality for TAs with one action is undecidable [26], we conclude that the full opacity problem for one-action TAs is undecidable.

Finally, with Theorem 1, we deduce the undecidability of weak opacity for TAs with one action. □

D Opacity of TAs over Discrete Time

Lemma 1. *The language of a discrete-time TA and the language of its region automaton are in bijection.*

Proof. Let $\mathcal{A}$ be a discrete-time TA. We explicit the bijection of the lemma.

Given a path ρ of $\mathcal{A}$ generating the timed word w, as $\mathcal{A}$ includes ticks, w is of the form

$$(t,1)\dots(t,\tau_0)(a_0,\tau_0)\ (t,\tau_0{+}1)\dots(t,\tau_1)(a_1,\tau_1)\ \dots\ (t,\tau_{n-1}{+}1)\dots(t,\tau_n)(a_n,\tau_n).$$

To the timed word w, we associate the untimed word produced within the region automaton by the path $[\rho]$) corresponding to ρ:

$$\underbrace{tt\dots t}_{\tau_0 \text{ times}}\ a_0\ \underbrace{tt\dots t}_{(\tau_1-\tau_0) \text{ times}}\ a_1\ \dots\ \underbrace{tt\dots t}_{(\tau_n-\tau_{n-1}) \text{ times}}\ a_n.$$

This association is injective as the sequence $(\tau_i)_{i\leq n}$ which was removed in the transformation depends only on the number of t of the timed word. Moreover, it is surjective as given an untimed word in $\mathcal{RA}_{\mathcal{A}}$ $w' = \underbrace{tt\dots t}_{k_0 \text{ times}}\ a_0\ \underbrace{tt\dots t}_{k_1 \text{ times}}\ a_1\ \dots\ \underbrace{tt\dots t}_{k_n \text{ times}}\ a_n$ produced by a path $[\rho']$ of the region automaton, defining

$$w = (t,1)\dots(t,k_0)(a_0,k_0)(t,k_0+1)\dots(t,k_0+k_1)(a_1,k_0+k_1)\dots(a_n,\sum_{i=0}^{n} k_i)$$

we have that w is the timed word generated by the unique path of the TA corresponding to ρ' and w is associated with w'. □

Proposition 1. *Language inclusion in discrete-time TAs is* EXPSPACE-*complete.*

We separate both directions of the proof, due to how voluminous the hardness is. We start by showing that the language inclusion in discrete-time TAs can be achieved in EXPSPACE.

Proof. Let $\mathcal{A}$ and $\mathcal{B}$ be two discrete-time TAs, and let $\mathcal{RA}_\mathcal{A}$ and $\mathcal{RA}_\mathcal{B}$ be their respective region automata. Then from Lemma 1, we have

$$Tr(\mathcal{A}) \subseteq Tr(\mathcal{B}) \text{ if and only if } Tr(\mathcal{RA}_\mathcal{A}) \subseteq Tr(\mathcal{RA}_\mathcal{B})$$

Thus deciding the language inclusion in discrete-time TAs amounts to solving the language inclusion problem in the context of finite regular automata, which can be done in PSPACE in the size of the region automata. Noting that the region automata of the ticked TA is exponential in the size of the initial TA, this produces an EXPSPACE algorithm. □

Let us now show that the language inclusion in discrete-time TAs is EXPSPACE-hard. To do so, we will reduce a succinct variant of the equality of rational expressions.

Definition 12 (Rational expressions with square). *The expressions $\emptyset$, ε, and a with $a \in \Sigma$ are* rational expressions with square. *If exp_1 and exp_2 are rational expressions with square, then so are $exp_1 + exp_2$, $exp_1 \cdot exp_2$, exp_1^* and exp_1^2.*

A rational language with square *is a set of words on Σ represented by a rational expression with square.*

The operators on the rational expressions are interpreted in the usual way. For instance, the expression $(a+ab)^2$ represents the set of words $\{aa, aab, aba, abab\}$. There can be several expressions representing the same language.

The expressivity of rational languages with square is exactly the same as of rational languages since using the square is equivalent to concatenating an expression with itself. However, the description of a language with square may be exponentially more succinct. Hence why we have

Proposition 2. *[25] Let L_1 and L_2 be two rational languages with square. Deciding whether $L_1 = L_2$ is* EXPSPACE-*complete.*

Proof (Proof of the hardness of language inclusion for discrete-time TA). Let L be a rational language with square and *exp* be the rational expression with squares that represents it. From the structure of *exp*, we will build a timed automaton which untimed language is L.

Since we will compare the timed language of TAs, and we only want to compare their untimed languages, we need to impose a standard for the timestamps of their words. We choose that each action must occur at an even number of time units. More precisely, if the automaton recognizes words of at least one letter it will read the first one without any delay and waits two time units between each letter. To do this we use the clock x which is reused for all the constructions, and which is reset only when a letter occurs. Every operation, beside reading a letter, must then be done in time 0. In particular, our constructions always start and end with $x = 0$, and only allows time to elapse when a letter is read. We present in the following table (Fig. 8) the inductive constructions corresponding to the basic rational expressions and the operators $+, \cdot, ^*$. The case of the square operator is explained separately.

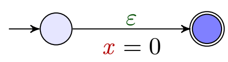
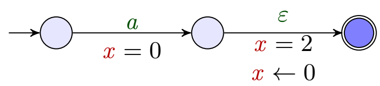
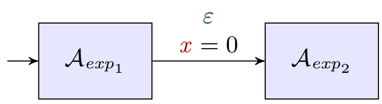
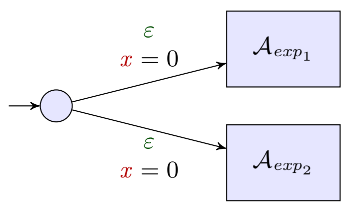
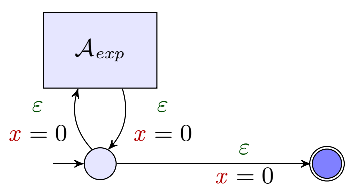
Rational expression with square exp	Timed automaton $\mathcal{A}_{exp}$
ε	
$a \in \Sigma$	
$exp_1 \cdot exp_2$	
$exp_1 + exp_2$	
exp^*	

Fig. 8: Table of timed automata constructions $\mathcal{A}_{exp}$ for regular expressions

For the square construction $\mathcal{A}_{exp^2}$ (Fig. 9) we need to add three additional clocks per square occurrence: the first one, z, manages the particular case of the empty word ε by detecting whether some time has passed during the crossing of $\mathcal{A}_{exp}$, while the clocks y and v are used to force exactly two passages in $\mathcal{A}_{exp}$.

Indeed, the shift between the clocks x, y and v (with values kept between zero and two all along the run) permits to keep in memory the number of remaining passage in $\mathcal{A}_{exp}$ by being modified once during the first passage (①), a second time between the first as second passage (②), and being checked at the end of the second passage (③). These added clocks cannot be reused in nested squares constructions. Thus we introduce a number of clocks equal to three times the maximal number of nested squares in the expression to build the corresponding timed automaton.

More precisely, the previously built TA $\mathcal{A}_{exp}$ is modified into $\tilde{\mathcal{A}}_{exp}$ by adding on every location silent loop transitions resetting y and v when they reach 2, as well as a silent loop transition with guard $x = 1 \wedge y = 1$ and reset set $\{y, v\}$. At most one of the latter loops, denoted by ①, is taken during an execution, and it requires at least one letter to be triggered. This transition ensures that $y = v \neq x$ in the following. This property is necessary to take the transition ②, which now ensures that $x = v \neq y$, which will allow taking the transition ③. As mentionned, taking the transition ① requires at least one letter to be read, hence why, when *exp* contains the empty word, we need the clock z to give an alternative way to exit the gadget. Formally, we have

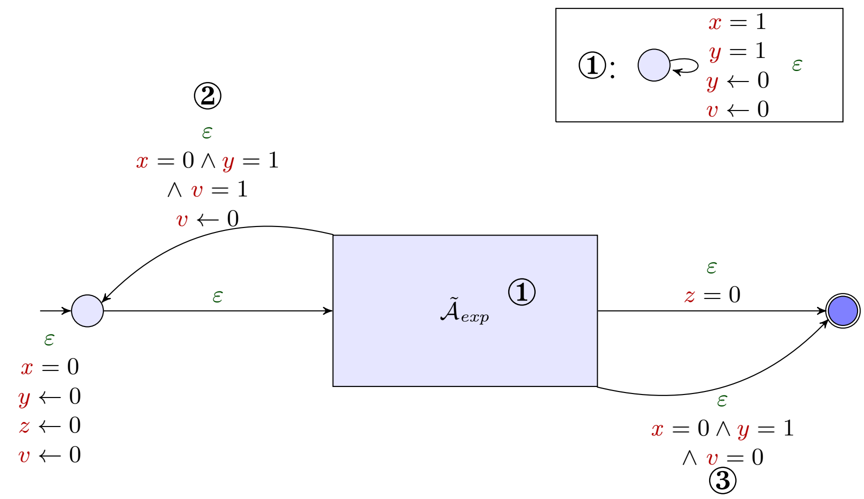


Fig. 9: TA $\mathcal{A}_{exp^2}$
$\tilde{\mathcal{A}}_{exp}$ is the automaton $\mathcal{A}_{exp}$ modified through ①.

Definition 13 (Square construction). *Let* $\mathcal{A}_{exp} = (\Sigma, L, \ell_0, \emptyset, L_f, \mathbb{X}, I, E)$ *be the timed automaton corresponding to the rational expression with square exp. Then, the TA corresponding to* exp^2 *is* $\mathcal{A}_{exp^2} = (\Sigma, L \cup \{\ell'_0, \ell'_f\}, \ell'_0, \emptyset, \left\{\ell'_f\right\}, \mathbb{X} \cup \{v, y, z\}, I', E')$ *where* I' *is the extension of* I *such that* $I'(\ell'_0) = I'(\ell'_f) = true$*, and*

$$E' = E \cup \{(\ell'_0, \varepsilon, true, \emptyset, \ell_0)\} \cup \bigcup_{\ell \in L} \{(\ell, \varepsilon, y = 2, \{y\}, \ell), (\ell, \varepsilon, v = 2, \{v\}, \ell), (\ell, \varepsilon, x = 1 \wedge y = 1, \{v, y\}, \ell)\} \cup \bigcup_{\ell_f \in L_f} \{(\ell_f, \varepsilon, x = 0 \wedge y = 1 \wedge v = 1, \{v\}, \ell'_0), (\ell_f, \varepsilon, z = 0, \emptyset, \ell'_f), (\ell_f, \varepsilon, x = 0 \wedge y = 1 \wedge v = 0, \emptyset, \ell'_f)\}.$$

Two rational languages with square L_1 and L_2, respectively represented by the expressions exp_1 and exp_2, are equal if and only if the automata $\mathcal{A}_{exp_1}$ and $\mathcal{A}_{exp_2}$ recognize the same timed language. The obtained automata are timed automata with discrete time of polynomial size in the rational expressions. Thus from Proposition 2 follows the EXPSPACE-hardness of language inclusion for discrete-time TA. ◻

Theorem 7. *Both weak and full opacity of discrete-time TAs are* EXPSPACE-*complete.*

Proof. Let $\mathcal{A}$ be a discrete-time automaton with private locations set L_{priv}. The construction in Section 4.1 is still compatible with discrete time clocks so we can build two discrete-time TAs $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$ such that $Tr(\mathcal{A}_{priv}) = Tr^{priv}(\mathcal{A})$ and $Tr(\mathcal{A}_{pub}) = Tr^{\overline{priv}}(\mathcal{A})$. Then testing the weak opacity property on $\mathcal{A}$ is equivalent to testing the inclusion $Tr(\mathcal{A}_{priv}) \subseteq Tr(\mathcal{A}_{pub})$. Therefore the weak opacity problem in discrete-time TAs is in EXPSPACE.

EXPSPACE-hardness can easily be obtained by the following reduction: Given two TA $\mathcal{A}$ and $\mathcal{B}$, one can build a TA which private runs are the runs of $\mathcal{A}$ and which public ones are those of $\mathcal{B}$. We do this by making the initial location of $\mathcal{A}$ private and considering the natural construction of the union of $\mathcal{A}$ and $\mathcal{B}$. Hence comparing the languages of $\mathcal{A}$ and $\mathcal{B}$ amounts to testing weak opacity on the built automaton.

As before, thanks to Theorem 1, we can extend this result to the full opacity problem. ◻

E PSPACE-completeness of Weak / Full Opacity for oERAs

Let us first explain why the algorithm for weak opacity presented in the main document is in PSPACE. As a summary, this algorithm consists in, given an oERA $\mathcal{A}$, building the corresponding $\mathcal{A}_{memo}$, determinizing it through the subset construction, taking its region automaton, and then testing reachability of a location containing a final private location, but no final public location.

The determinization of the oERA causes the number of locations to become exponential in the size of the entry, and the construction of the region automaton gives an exponential number of clock regions, bounded by $|\mathbb{X}| \cdot 2^{|\mathbb{X}|} \cdot \prod_{x \in \mathbb{X}} (2M(x) + 2)$ [2]. The size of the region automaton is thus exponential in the number of locations and the number of clocks of $\mathcal{A}$. On the region automaton, testing the reachability of a location can be done in NLOGSPACE. Hence the problem of weak opacity in oERA is in PSPACE.

Let us now explain why these problems are PSPACE-hard. We reduce from the reachability problem for TA, which is PSPACE-complete.

Let $\mathcal{A}$ be a timed automaton, with a set of final locations L_f. We consider $\mathcal{A}'$ the TA obtained by setting in $\mathcal{A}$ the set of private locations to L_f. This way, every run of $\mathcal{A}'$ are private. Thus $\mathcal{A}'$ is weakly opaque if and only if no final location of $\mathcal{A}$ is reachable. Hence, the weak opacity problem in oERA is hence PSPACE-hard.

These results extend to full opacity thanks to Theorem 1.

F Opacity with N Observations

Given a $\mathcal{A} = (\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$ and $N \in \mathbb{N}$. We build the unfolding of the TA $\mathcal{A}_{memo}$ described in Remark 1. Recall that $\mathcal{A}_{memo}$ recognizes the same language as $\mathcal{A}$ but stores within the locations the information whether L_{priv} was visited. As such, $\mathcal{A}_{memo}$ has the same opacity properties as $\mathcal{A}$, so we can consider $\mathit{Unfold}_N(\mathcal{A}_{memo})$ instead of $\mathit{Unfold}_N(\mathcal{A})$ to study the opacity of $\mathcal{A}$.

Additionally, we enrich this TA with ticks. In Section 5.4, we added a single tick to the automaton which counted the time elapsed since the start of the run. Here, the TA includes as well, for each $0 < k \leq N$, a tick counting the time elapsed since the k'th observation. As multiple ticks may need to occur at the same time, we develop the alphabet of ticks to describe the set of tick clocks that need to be reset, i.e., the tick $t_{\{k_1,\dots,k_m\}}$ is produced by the TA if for every $0 \leq i \leq m$, the k_i'th observation (or the start of the run if $k_i = 0$) occurred an integer number of time units beforehand. Note that the addition of these ticks immediately uses the assumption that only N actions are observed.

Definition 14 (Addition of ticks to the Unfolding construction). *Let $\mathcal{A} = (\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$ be a TA, $N \in \mathbb{N}$ and let $\mathit{Unfold}_N(\mathcal{A}) = (\Sigma, L', \ell_0^0, L'_{priv}, L'_f, \mathbb{X}, I', E')$ the unfolding of $\mathcal{A}$. We define the Tick construction $\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A})) = (\Sigma', L', \ell_0^0, L'_{priv}, L'_f, \mathbb{X}', I', E'')$ where*

1. *$\Sigma' = \Sigma \cup \Sigma^0 \cup \Sigma_t$ where $\Sigma^0 = \left\{a^0 \mid a \in \Sigma\right\}$ is a copy of the alphabet Σ that is used to represent within the action's name that it occurred at the same time as the previous action, and $\Sigma_t = \{t_K \mid K \subseteq [\![0; N]\!], K \neq \emptyset\}$ is the set of ticks associated with each set of added clocks;*
2. *$\mathbb{X}' = \mathbb{X} \cup \mathbb{X}_t$ where $\mathbb{X}_t = \{x_i \mid i \in [\![0; N]\!]\}$ is the set of the $N+1$ tick clocks;*

3. $E'' = \bigcup_{i=0}^{N-1} E^i \cup E^{i \to i+1}$ *is the set of transitions where, given* $0 \leq i < N$

– $E^i = \{(\ell^i, \varepsilon, g \wedge \bigwedge_{k=0}^{i} (x_k < 1), R, \ell'^i) \mid (\ell, \varepsilon, g, R, \ell') \in E'\} \cup \{(\ell^i, t_K, \bigwedge_{k \in K} (x_k = 1) \wedge \bigwedge_{m \in [\![0;i]\!] \setminus K} (0 < x_m < 1), \{x_k \mid k \in K\}, \ell^i) \mid \ell^i \in L^i \wedge K \subseteq [\![0;i]\!] \wedge K \neq \emptyset\}$;

– $E^{i \to i+1} = \{(\ell^i, a^0, g \wedge \bigwedge_{k=0}^{i} (x_k < 1) \wedge \bigvee_{m=0}^{i} (x_m = 0), R \cup \{x_{i+1}\}, \ell'^{i+1}) \mid (\ell, a, g, R, \ell') \in E'\} \cup \{(\ell^i, a, g \wedge \bigwedge_{k=0}^{i} (0 < x_k < 1), R \cup \{x_{i+1}\}, \ell'^{i+1}) \mid (\ell, a, g, R, \ell') \in E'\}$.

We obtain in this way the timed automaton $\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo}))$. Let $\mathcal{RA}_{\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo}))}$ be the region automaton of this automaton. Thanks to the added ticks, paths of $\mathcal{RA}_{\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo}))}$ sharing the same trace correspond to runs of $\mathcal{A}$ for which the (at most) N observations occurred within the same time intervals (due to the tick representing the total time) and the fractional part of the timing of those observations have the same order. This is the information we mainly need, and thus we wish to regroup every path of the region automaton with the same trace. As the region automaton is a finite automaton, we can realize usual operations on it, that is, first remove ε-transitions (by fusing them with the following non-ε-transition) and then determinizing the automaton through the subset construction. We denote by $\mathcal{B}(\mathcal{A})$ the resulting automaton. We call *beliefs* the states of $\mathcal{B}(\mathcal{A})$, i.e., they describe the set of regions the attacker believes the system may be in.

Let B be a belief of $\mathcal{B}(\mathcal{A})$ and B_{priv} (resp. B_{pub}) be the subset of B containing the regions which associated location in $\mathcal{A}_{memo}$ is private (resp. public) and final. We say that B is weakly (resp. fully) *discriminating* if $B_{priv} \neq \emptyset$ and $B_{pub} = \emptyset$ (resp. if either $B_{priv} \neq \emptyset$ and $B_{pub} = \emptyset$ or $B_{priv} = \emptyset$ and $B_{pub} \neq \emptyset$). The discriminating beliefs in $\mathcal{B}(\mathcal{A})$ allow to characterize the opacity problems.

Proposition 3 (Relation between opacity and discriminating belief). *A TA $\mathcal{A}$ is weakly (resp. fully) opaque w.r.t. N observations if and only if $\mathcal{B}(\mathcal{A})$ does not contain any weakly (resp. fully) discriminating belief.*

Proof. We focus on weak opacity, the full opacity case can be treated similarly.

– Assume first that $\mathcal{B}(\mathcal{A})$ contains a weakly discriminating belief B. Let r be a region in B_{priv} and w be the trace of a path leading from the initial belief of $\mathcal{B}(\mathcal{A})$ to B. By construction of the region automaton, there exists a run ρ of $\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo}))$ whose untimed trace (i.e., the trace of ρ projected on the actions) is w and such that the run corresponding to ρ in the region automaton ends in r. In particular, ρ is a private run. Moreover, any run whose untimed trace is w ends in a region of B. Thus, there is no public run with trace w and in particular $\mathit{Tr}(\rho) \in \mathit{Tr}^{priv}(\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo})))$ and $\mathit{Tr}(\rho) \in \mathit{Tr}^{\overline{priv}}(\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo})))$, hence $\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo}))$ is not weakly opaque and $\mathcal{A}$ is not weakly opaque w.r.t. N observations.

– Assume now that $\mathcal{A}$ is not weakly opaque w.r.t. N observations. Let ρ be a run of $\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo}))$ such that $\mathit{Tr}(\rho) \in \mathit{Tr}^{priv}(\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo})))$ and $\mathit{Tr}(\rho) \notin \overline{\mathit{Tr}^{priv}}(\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo})))$. Let $[\rho]$ be the run corresponding to ρ in the region automaton.[6] We denote by $T([\rho])$ the set of traces of runs of $\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo}))$ associated with $[\rho]$.

Lemma 2. *Denoting $w = a_0, \dots, a_m$ the trace of $[\rho]$, $T([\rho])$ contains exactly the words $(a_0, \tau_0) \dots (a_m, \tau_m)$ satisfying the following constraints:*

1. $\forall i \in [\![0; m]\!], (a_i \in \Sigma \cup \Sigma_t \implies \tau_i - \tau_{i-1} > 0) \wedge (a_i \in \Sigma^0 \implies \tau_i - \tau_{i-1} = 0)$ *(where $\tau_{-1} = 0$), meaning that two consecutive observable actions occur at the same time if and only if the second one is in Σ^0.*
2. $\forall i, j \in [\![0; m]\!], \forall J \subseteq [\![0; N]\!], \forall I \subseteq J, \big(i < j \wedge a_i = t_I \wedge a_j = t_J \wedge \forall k \in [\![i+1; j-1]\!], \forall K \subseteq [\![0; N]\!](a_k = t_K \implies K \cap J = \emptyset)\big) \implies \tau_j - \tau_i = 1$, *meaning that two successive ticks of the same clocks are separated by exactly 1 time unit.*
3. $\tau_m \geq 1 \implies \big(\exists i \in [\![0; m]\!], \exists I \subseteq [\![0; N]\!] \forall j < i, a_i = t_I \wedge 0 \in I \wedge \tau_i = 1 \wedge (a_j \notin \Sigma_t)\big)$*, meaning that the first occurrence of the tick of the clock x_0 is at time 1.*
4. $\forall i \in [\![0; m]\!], (a_i \in \Sigma \cup \Sigma^0 \wedge \tau_m - \tau_i \geq 1) \implies (\exists k \in [\![0; m]\!], \exists K \subseteq [\![0; N]\!], a_k = t_K \wedge |\{j \in [\![0; i]\!] \mid a_j \in \Sigma \cup \Sigma^0\}| \in K \wedge \tau_k - \tau_i = 1)$ *meaning that each of the N observations is followed by its corresponding tick exactly one time unit after it.*
5. $\forall i \in [\![0; m]\!], \forall I \subseteq [\![0; N]\!], (a_i = t_I \wedge \tau_m - \tau_i \geq 1) \implies \exists j \in [\![i+1; m]\!], \exists J \subseteq [\![0; N]\!], (I \subseteq J \wedge a_j = t_J)$ *meaning that if a clock ticked and the run is still at least one time unit long, then there will be a new tick of this clock within the rest of the run.*

Due to its size, we postpone the proof of this lemma to the bottom of this section.

Note that this lemma implies that $T([\rho])$ depends exclusively on the trace w, not on the path within the region automaton. Hence, given $[\rho']$ such that the trace of $[\rho']$ is w, we have $T([\rho']) = T([\rho])$. In particular, let B be the belief reached in $\mathcal{B}(\mathcal{A})$ with trace w. For any region $r \in B$ associated with a final location, there exists a run ρ' such that $\mathit{Tr}(\rho) = \mathit{Tr}(\rho')$ and $[\rho']$ ends in r. As $\mathit{Tr}(\rho) \notin \overline{\mathit{Tr}^{priv}}(\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo})))$ by assumption, we have that r is a region associated with a private location. Hence $B_{priv} \neq \emptyset$ and $B_{pub} = \emptyset$, thus B is a weakly discriminating belief. □

Proof (Proof of Theorem 9). From Proposition 3, deciding weak and full opacity of $\mathcal{A}$ amounts to checking the existence of a discriminating belief in $\mathcal{B}(\mathcal{A})$. This is simply achieved by a reachability test in the finite automaton $\mathcal{B}(\mathcal{A})$.

[6] The notation $[\cdot]$ represents that $[\rho]$ implicitly defines an equivalence class of runs of $\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo}))$. For a run ρ' of $\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo}))$, we thus write $\rho' \in [\rho]$ to say that the run associated with ρ' in the region automaton is $[\rho]$.

Considering the complexity, the unfolding of the TA, assuming N is in binary, has exponentially many states. Adding the ticks means adding an exponential number of clocks as well. Hence the region automaton is doubly exponential and its determinisation is triply exponential. Reachability being in NLOGSPACE implies the 2-EXPSPACE algorithm. If N is given in unary, the complexity falls to EXPSPACE. $\square$

Let us finally establish Lemma 2. For ease of readability, we separate the proofs of the two inclusions implying the lemma.

First direction of the proof We first show the easy direction of the proof: the timed words in $T([\rho])$ satisfy the five properties.

Proof. Let $u = (a_0, \tau_0)\ldots(a_m, \tau_m)$ be in $T([\rho])$. Since $u \in T([\rho])$, there exists a run ρ' in $[\rho]$ on $\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo}))$ which produces the trace u:

$$\rho' = (\ell_0, \mu_0) \xrightarrow{(d_0,e_0)} \ldots (\ell_{j_i-1}, \mu_{j_i-1}) \xrightarrow{(d_{j_i-1},e_{j_i-1})} (\ell_{j_i}, \mu_{j_i}) \ldots (\ell_n, \mu_n)$$

Recall the link between the trace of a run and its transitions: for every $i \in [\![0; m]\!]$, the j_i index corresponds to the i-th observable action, i.e., j_i satisfies $\tau_i = \sum_{k=0}^{j_i} d_k$ and e_{j_i} is labeled by a_i. We set $\tau_{-1} = 0$.

- *Property 1*. Let $i \in [\![0; m]\!]$. The set of clocks reset by ε-transitions is included in $\mathbb{X}$. By definition of the indices j_i, there are only ε-transitions between the configurations $(\ell_{j_{i-1}+1}, \mu_{j_{i-1}+1})$ if $i > 0$ or the initial configuration if $i = 0$, and (ℓ_{j_i}, μ_{j_i}). Thus, no clock from $\mathbb{X}_t$ is reset among these transitions and we have $\mu_{j_{i-1}+1}(x) + \sum_{k=j_{i-1}+1}^{j_i-1} d_k = \mu_{j_i}(x)$ for each $x \in \mathbb{X}_t$. Assume $a_i \in \Sigma^0$. As $a_i \in \Sigma^0$, the guard of e_{j_i} is of the form $g \wedge \bigwedge_{k=0}^{h} (x_k < 1) \wedge \bigvee_{k=0}^{h} (x_k = 0)$ with some guard g and the copy number $h \leq i$ of $e_{j_i} \in E^{h \to h+1}$ (in other words, the integer h such that the transition of the original automaton starts in the copy L^h). In particular, there exists a clock $x \in \mathbb{X}_t$ such that $\mu_{j_i}(x) + d_{j_i} = 0$. Therefore we obtain $\mu_{j_{i-1}+1}(x) + \sum_{k=j_{i-1}+1}^{j_i} d_k = 0$, which implies $\tau_i - \tau_{i-1} = 0$. Let us now assume that $a_i \in \Sigma \cup \Sigma_t$, and show that $\tau_i - \tau_{i-1} > 0$. Depending on whether a_i is in Σ or in Σ_t, the guard of the transition e_{j_i} can be of two forms : $g \wedge \bigwedge_{k=0}^{h} (0 < x_k < 1)$ with g some guard and h some copy number, if $a_i \in \Sigma$; or $\bigwedge_{k \in K} (x_k = 1) \wedge \bigwedge_{m \in [\![0;h]\!] \setminus K} (0 < x_m < 1)$ with $K \subseteq [\![0; h]\!]$ non-empty and h a copy number, if $a_i \in \Sigma_t$. In both cases, we must have $\mu_{j_i}(x_k) + d_{j_i} > 0$ for each $x_k \in \mathbb{X}_t$ such that $k \leq h$. However the last observable transition $e_{j_{i-1}}$ resets one of those clocks x, which gives $\mu_{j_{i-1}+1}(x) = 0$. Hence $0 < \mu_{j_i}(x) + d_{j_i} = \mu_{j_{i-1}+1}(x) + \sum_{k=j_{i-1}}^{j_i} d_k = \sum_{k=j_{i-1}}^{j_i} d_k$.

- *Property 2*. Let $i, i' \in [\![0; m]\!]$, with $i < i'$. Assume there are $I' \subseteq [\![0; N]\!]$ and $I \subseteq I'$ such that $a_i = t_I$ and $a_{i'} = t_{I'}$. Suppose that for every $k \in [\![i{+}1; i'{-}1]\!]$ and for each $K \subseteq [\![0; N]\!]$, $a_k = t_K \implies K \cap I' = \emptyset$. Let us show that $\tau_{i'} - \tau_i = 1$.

  Let $x \in \{x_k \mid k \in I\} \subseteq \mathbb{X}_t$. After the reset applied by e_{j_i}, we have: $\mu_{j_i+1} = [\mu_{j_i} + d_{j_i}]_{\{x_k|k\in I\}} \models (x = 0)$. In order to take the transition $e_{j_{i'}}$, the clock valuation $\mu_{j_{i'}} + d_{j_{i'}}$ needs to satisfy the guard $(x = 1)$. It remains to show that x is not reset between e_{j_i} and $e_{j_{i'}}$. Let e be a transition between e_{j_i} and $e_{j_{i'}}$. If e is an ε-transition, it does not reset any clock in $\mathbb{X}_t$. If e is labeled by a letter in $\Sigma \cup \Sigma^0$, it is in some $E^{h' \to h'+1}$ with $h' \geq h$ and h the copy number of the configuration preceding e_{j_i}. The only clock e resets that is not in $\mathbb{X}$ is $x_{h'+1}$, and cannot be x since $I \subseteq [\![0; h]\!]$. Finally, if $e = (\ell, t_K, g, R, \ell')$ is labeled by a tick t_K for some $K \subseteq [\![0; N]\!]$, the hypothesis $K \cap I' = \emptyset$ in the property ensures that x is not part of the set of clocks reset by e. Hence, we obtain $\mu_{j_{i'}}(x) + d_{j_{i'}} = [\mu_{j_i} + d_{j_i}]_{\{x_k|k\in I\}} + \sum_{k=j_i+1}^{j_{i'}} d_k = \tau_{i'} - \tau_i = 1$, which concludes the proof of the second point.
- *Properties 3 and 4*. Both Properties 3 and 4 are similar and require the first tick of a clock in $\mathbb{X}_t$ to occur one time unit after reaching the corresponding copy of the automaton. Indeed Property 3 is the particular case of $h = 0$ in the following proof, and Property 4 is the case $h > 0$. Let $h \in [\![0; N]\!]$. We focus on the clock $x_h \in \mathbb{X}_t$. Let i be the index of the h-th observation in the trace, that is to say the h-th letter of the trace that is in $\Sigma \cup \Sigma^0$. If $h = 0$, we set $i = -1$. In any cases τ_i is the time of arrival in the copy h via the transition $e_{j_i} \in E^{h-1 \to h}$, and at this precise moment $x_h = 0$. Every guard of the transitions of $E^{h'}$ and $E^{h' \to h'+1}$ with $h' \geq h$ requires that $x_h < 1$, except for the transitions labeled by t_K with $h \in K$, which reset x_h and require $x_h = 1$. Thus, x_h must have been reset if the run lasts more than one time unit after the h-th observation, i.e., if $\tau_m - \tau_i \geq 1$. The only transitions from the copy h that can reset x_h are those labeled by t_K with $h \in K$; their guards require $x_h = 1$ so the first reset of x_h after reaching the copy h needs to occur at time $\tau_i + 1$.
- *Property 5*. We use the same argument as Properties 3 and 4 to prove the fifth property. Let $i \in [\![0; m]\!]$, and suppose $a_i = t_I$ for some non-empty $I \subseteq [\![0; N]\!]$ and $\tau_m - \tau_i \geq 1$. The transitions following e_{j_i} require $\bigwedge_{k\in I} (x_k < 1)$, unless they reset all the clocks in $R_I = \{x_k \mid k \in I\}$. As $\tau_m - \tau_i \geq 1$, an observable action occurs at least one time unit after e_{j_i} and the clocks of R_I thus needed to be reset. Hence there is a tick transition labeled by t_J with $I \subseteq J$ happening after e_{j_i}.

□

Second direction of the proof Now we tackle the second direction of the proof: we show that if $a_0 \dots a_m$ is the trace of a run $[\rho]$, then all the timed words

$(a_0, \tau_0) \dots (a_m, \tau_m)$ that satisfy the five properties of the lemma are in $T([\rho])$, i.e., there are runs in $[\rho]$ which produce these timed traces.

Proof. Let $w = a_0 \dots a_m$ be a trace in $\mathcal{RA}_{Tick(Unfold_N(\mathcal{A}_{memo}))}$, and let $\tau = (\tau_0, \dots, \tau_m)$ be a timestamps sequence such that the timed word $(a_0, \tau_0) \dots (a_m, \tau_m)$ verifies the five properties of the lemma. We define the sequence $(f_i)_{0 \leq i \leq N}$ of the timestamps' fractional parts associated with each tick clock. For each $i \in [\![0; N]\!]$ we set $J_i = \{j \in [\![0; m]\!] | \exists I \subseteq [\![0; N]\!], i \in I \wedge a_j = t_I\}$. Let $\mathit{first}(i) = \min \{j \in [\![0; m]\!] | | \{k \in [\![0; j]\!] | a_k \in \Sigma \cup \Sigma^0\} | = i\}$ (be the subscript of the i-th observation) if $i > 0$ and $\mathit{first}(0) = -1$. Thus $\mathit{first}(i)$ is the subscript of the first observation in w that resets the clock $x_i \in \mathbb{X}_t$. We set $f_i = \mathrm{frac}(\tau_{\mathit{first}(i)})$. Then, from Properties 2 and 4 we have that for each i in $[\![0; N]\!]$ and for each $j \in J_i$, $\mathrm{frac}(\tau_j) = f_i$. From Property 3, we get $f_0 = 0$.

If $i, i' \in [\![0; N]\!]$ with $i < i'$, then Property 1 ensures that $f_i = f_{i'}$ if and only if there exist $j \in J_i \cup \{\mathit{first}(i)\}$ such that $j + 1 = \mathit{first}(i')$ and $a_{\mathit{first}(i')} \in \Sigma^0$. Note that the sequence $(f_i)_{0 \leq i \leq N}$ depends only on the timed trace and not on the path in the region automaton.

Let $[\rho] : r_0, b_0, r_1, \dots b_{p-1} r_p$ be a run in $\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo}))$ of trace w. For $n \in [\![0; p]\!]$ we define (deduced from τ) the constraint C_{τ, r_n} on the tick clocks, in order to produce a run of $[\rho]$ that corresponds to τ. We denote by ℓ_n^h the location of r_n with h the number of the copy of L it belongs to, and by $[\mu_n]$ the clock region of r_n.

We distinguish two cases depending on whether r_n is a region where at least one tick clock has integer value (noting that, by property of the region automaton, if one valuation of $[\mu_n]$ give an integer value to a clock, then all valuations of $[\mu_n]$ do).

- $\exists x_i \in \mathbb{X}_t, \mu_n(x_i) \in \mathbb{N}$**:** In this case and if $i \leq h$ we set *(type 1 constraint)*

$$\begin{aligned} C_{\tau, r_n} = \bigwedge_{k=0}^{h} & ((f_k \leq f_i \implies \mathrm{frac}(x_k) = f_i - f_k) \\ & \wedge (f_i < f_k \implies \mathrm{frac}(x_k) = f_i - f_k + 1)) \\ & \wedge \bigwedge_{k=h+1}^{N} (\mathrm{frac}(x_k) = x_0) \end{aligned}$$

- $\forall x_i \in \mathbb{X}_t, \mu_n(x_i) \notin \mathbb{N}$**:**
  Let i be the index of one of the last tick clocks that were reset in $[\rho]$ before r_n. If there is no tick clock reset before r_n, we set $i = 0$. Similarly, we consider $j \in [\![0; h+1]\!]$ one of the clock indices of the next tick clock reset after r_n. If there is no tick clock reset after r_n, we take the next tick clock reset that is supposed to occur, i.e., $j = \min \{j' \in [\![1; h]\!] | f_i < f_{j'}\}$ or $j = 0$ if this set is empty.
  If $j \in J_0$, we set *(type 2 constraint)*

$$C_{\tau, r_n} = \exists \delta \in (0; 1) \bigwedge_{k=0}^{h} (f_i - f_k < x_k < 1 - f_k \wedge 1 - f_k - x_k = \delta) \wedge \bigwedge_{k=h+1}^{N} (\mathrm{frac}(x_k) = x_0)$$

Otherwise, and if we have $0 < j < h+1$ and $j \notin J_0$, we set *(type 3 constraint)*

$$\begin{aligned} C_{\tau,r_n} = \exists \delta \in (0;1) \bigwedge_{k=0}^{h} &((f_k < f_j \implies (f_i - f_k < x_k < f_j - f_k \wedge f_j - f_k - x_k = \delta)) \\ &\wedge (f_j \le f_k \implies (f_i - f_k + 1 < x_k < f_j - f_k + 1 \wedge f_j - f_k + 1 - x_k = \delta))) \\ &\wedge \bigwedge_{k=h+1}^{N} (\mathrm{frac}(x_k) = x_0) \end{aligned}$$

If $j = h+1$ we operate a slight change on this formula to obtain the following one: *(type 4 constraint)*

$$\begin{aligned} C_{\tau,r_n} = \exists \delta \in (0;1) \bigwedge_{k=0}^{h} &((f_k < f_{h+1} \implies (f_i - f_k < x_k \le f_{h+1} - f_k \wedge f_{h+1} - f_k - x_k = \delta)) \\ &\wedge (f_{h+1} < f_k \implies (1 - f_k < x_k \le f_{h+1} - f_k + 1 \wedge f_{h+1} - f_k + 1 - x_k = \delta))) \\ &\wedge \bigwedge_{k=h+1}^{N} (\mathrm{frac}(x_k) = x_0) \end{aligned}$$

Now we combine this information to the constraints of the clock regions of a run in the region automaton to build a set of runs in $\mathit{Tick}(\mathit{Unfold}_N(\mathcal{A}_{memo}))$.

For $n \in [\![0;p]\!]$ we define the set of valuations

$$M_n := \{\mu \in [\mu_n] \mid \mu \models C_{\tau,r_n}\}.$$

We denote by $[\rho]_\tau$ the subset of $[\rho]$ defined by $r'_0, b_0, r'_1, \ldots b_{p-1} r'_p$ where for each $n \in [\![0;p]\!], r'_n = (\ell^h_n, M_n)$. The idea is that the successive clock valuations of runs of $[\rho]_\tau$ are in these sets M_n of valuations which correspond to region changes: a tick clock reaches or exits an integer value each time the next clock valuation μ'_{n+1} does no more satisfy C_{τ,r_n}.

We prove by induction that $[\rho]_\tau$ is not empty. The set M_0 contains at least μ_0 since it verifies $C_{\tau,r_0} = (\mathrm{frac}(x_0) = 0) \wedge \bigwedge_{k=1}^{N} \mathrm{frac}(x_k) = 0$. Assume now there is a path $(\ell^0_0, \mu'_0), b_0, \ldots, (\ell^h_n, \mu'_n)$ with $\mu'_j \in M_j$ for each $j \in [\![0;n]\!]$. We show there exists $\mu'_{n+1} \in M_{n+1}$ such that $(\ell^0_0, \mu'_0), b_0, \ldots, (\ell^h_{n+1}, \mu'_{n+1})$ is a path in $r'_0, b_0, r'_1 \ldots r'_{n+1}$. It is well known [17] that in the region automaton, $((\ell, [\mu]), a, (\ell', [\mu'])) \in E_R$ if and only if for all $\mu \in [\mu]$ there exists $\mu' \in [\mu']$ such that $(\ell, \mu) \overset{e}{\mapsto} (\ell', \mu') \in E$, with e being the transition associated with $((\ell, [\mu]), a, (\ell', [\mu']))$ in the timed automaton. Since $\mu'_n \in [\mu_n]$, by the above property we can find $\mu'_{n+1} \in [\mu_{n+1}]$ following the transition (r_n, b_n, r_{n+1}). We show that there is such a μ'_{n+1} that is also in M_{n+1}, i.e., such that $\mu'_{n+1} \models C_{\tau,r_{n+1}}$.

- Suppose first (r_n, b_n, r_{n+1}) is a discrete transition, with reset set R. Only one clock valuation μ'_{n+1} can succede to μ'_n in this case. We show it is in M_{n+1}. If $b_n \in \Sigma$ (resp. Σ^0), then μ'_n verifies a type 1 (resp. 4) constraint.

We move to copy $h+1$ with a reset of some tick clocks (including x_{h+1}) so the next constraint to be verified is of type 1. The clock valuation μ'_{n+1} is entirely determined by μ'_n and the clock reset set.

Since we move to copy $h+1$, the clock x_{h+1} is now part of the tick clocks indexed from 0 to the current copy number, and $\mu'_{n+1}(x_{h+1}) = 0$, which is required by the constraint $C_{\tau,r_{n+1}}$. Moreover the constraints on the other clocks did not change. Thus the new type 1 constraint $C_{\tau,r_{n+1}}$ is verified by μ'_{n+1}. Now, if $b_n \in \Sigma_t$, only clocks reaching 1 are reset so their fractional part is not affected and the clock valuation μ'_{n+1} still satisfies C_{τ,r_n}. The constraint $C_{\tau,r_{n+1}}$ depends on the time elapsed since the last tick and whether the transition was an observation, so in this case $C_{\tau,r_{n+1}} = C_{\tau,r_n}$. Finally, if b_n is an ε-transition, it only resets clocks from $\mathbb{X}$ and we have again $C_{\tau,r_{n+1}} = C_{\tau,r_n}$. Since this constraint restricts only clocks from $\mathbb{X}_t$ and since their valuation does not change, μ'_{n+1} still verifies C_{τ,r_n} and is in M_{n+1}.

– We now assume that (r_n, b_n, r_{n+1}) is a delay transition: there is $d_n \in (0;1)$ such that $\mu'_{n+1} = \mu'_n + d_n$. This delay must verify some conditions because μ'_n is fixed and a transition can change region only once. In all this paragraph, we take i (resp. j) some index of the last (resp. next) reset tick clocks. Suppose that C_{τ,r_n} is a constraint of type 2, 3 or 4. Assume in the first case that $x_j \in \mathbb{X}_t$ is such that $\mu_{n+1}(x_j) \in \mathbb{N}$ (so the next constraint $C_{\tau,r_{n+1}}$ is of type 1). Thus this clock has reached 1 ($\mu'_{n+1}(x_j) = 1$) and C_{τ,r_n} is a type 2 or 3 constraint. This setting entirely determines the unique reachable clock valuation μ'_{n+1}. From the fact that μ'_n models C_{τ,r_n}, we have some $\delta \in (0;1)$ such that $1 + f_j - f_j - \mu'_n(x_j) = \delta$. Consequently $d_n = \delta$. We easily verify that the obtained μ_{n+1} satisfies $C_{\tau,r_{n+1}}$. Now if the next clock to be reset is x_{h+1} and there is no other delay transition before the next observation, then C_{τ,r_n} is a type 4 constraint and we need to have $\mu'_{n+1}(x_{h+1}) = f_{h+1}$, which means that $d_n = f_{h+1} - \mu'_n(x_0)$. The next constraint $C_{\tau,r_{n+1}}$ is still the same as C_{τ,r_n}. We have some δ for C_{τ,r_n} and obtain the new $\delta' = \delta - d_n = 0$ to satisfy $C_{\tau,r_{n+1}}$.

Suppose now that there will be at least one other delay transition before the next tick clock reset. Assume C_{τ,r_n} is of type 2, 3 or 4. Then, if this transition changes region, it only concerns clocks from $\mathbb{X}$ and we have $C_{\tau,r_n} = C_{\tau,r_{n+1}}$. The preceding constraint gives $\delta = 1 - \mu'_n(x_j)$ if C_{τ,r_n} is of type 2 or 3 and $\delta = f_j - \mu'_n(x_j)$ if it is of type 4. The delay d_n cannot reach δ (otherwise $\mu'_{n+1}(x_j) \in \mathbb{N}$ or the transition crosses more than one clock region at once, two excluded cases). We can construct the new $\delta' = \delta - d_n$ involved in $\mu'_{n+1} \models C_{\tau,r_{n+1}}$ and verify that μ'_{n+1} is thus still in M_{n+1}. Assume now that C_{τ,r_n} is of type 1. Then the next region must give a constraint of type 2, 3 or 4. This case is similar to the last tackled one, and for the same reasons the delay d_n cannot reach $1 - f_i + f_*$ where $f_* \in \{-f_j, 1 - f_j\}$. In both cases we obtain from the δ of C_{τ,r_n} the new one $\delta' = \delta - d_n$ for $C_{\tau,r_{n+1}}$, which is verified.

Hence and by induction, $[\rho]_\tau$ is not empty.

It remains to show that the timed trace these runs produce is $(a_0, \tau_0) \dots (a_m, \tau_m)$ (recall that is is the trace of $[\rho]$ timed by a timestamp τ which verifies the five properties of the lemma). Let $\rho' \in [\rho]_\tau$. Let (a_i, τ_i) be a letter of the timed trace. Suppose the action a_i occurs in ρ' at time τ'_i following the clock region $[\mu_{j_i}]$. Then the number of ticks of x_0 before a_i is given by the trace and is exactly $\lfloor \tau_i \rfloor = \lfloor \tau'_i \rfloor$. We know that $\mathrm{frac}(\tau'_i) = \mathrm{frac}(\mu_{j_i}(x_0))$. Moreover, a_i occurs at the same time as the reset of a clock x_k. If a_i is a tick, the constraint of M_{j_i} forces $\mu_{j_i}(x_0) = f_k$ and we have $i \in J_k$ so $\mathrm{frac}(\tau'_i) = f_k = \mathrm{frac}(\tau_i)$. Otherwise a_i is an observation, so there exists $h \in [\![0; N-1]\!]$ such that $i = \mathit{first}(h+1)$. The clock x_{h+1} has never been reset yet so $\tau'_i = \mu_{j_i}(x_{h+1})$. The constraint of M_{j_i} gives $\mathrm{frac}(\mu_{j_i}(x_{h+1})) = f_{h+1}$ and $\mathrm{frac}(\mu_{j_i}(x_0)) = f_{h+1}$. Since $i = \mathit{first}(h+1)$, we obtain $\mathrm{frac}(\tau_i) = f_{h+1} = \mu_{j_i}(x_0) = \mathrm{frac}(\tau'_i)$. □