



HAL
open science

Opacity problems in subclasses of timed automata

Étienne André, Sarah Dépernet, Engel Lefauchaux

► **To cite this version:**

Étienne André, Sarah Dépernet, Engel Lefauchaux. Opacity problems in subclasses of timed automata. 2024. hal-04631012v1

HAL Id: hal-04631012




<https://hal.science/hal-04631012v1>

Preprint submitted on 1 Jul 2024 (v1), last revised 1 Oct 2024 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Opacity problems in subclasses of timed automata

Étienne André^{1,2} , Sarah Dépernet^{3,4} , and Engel Lefauchaux⁴  *

¹ Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030, F-93430 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.

1 Introduction

The notion of *opacity* [21,14] 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 [18] 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.

1.1 Related works

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

* This work is partially supported by ANR BisoUS (ANR-22-CE48-0012).

30 deduce from the observation of a run whether it was a private or a public run.
 1 The attacker in [15] is able to observe a subset $\Sigma_o \subseteq \Sigma$ of actions with their
 2 timestamps. In this context, a timed word w is said to be opaque if there exists a
 3 public run that produces the projection of w following Σ_o as an observed timed
 4 word. In this configuration, one can consider the opacity problem consisting of
 5 determining, knowing a TA \mathcal{A} and a set of timed words, whether all words in
 6 this set are opaque in \mathcal{A} . This problem has been shown to be undecidable for
 7 TAs [15]. This notably relates to the undecidability of timed language inclusion
 8 for TAs [2]. However, the undecidability holds in [15] even for the restricted class
 9 of event-recording automata (ERAs) [3] (a subclass of TAs), for which language
 10 inclusion is decidable. The aforementioned negative results leave hope only if the
 11 definition or the setting is changed, which was done in three main lines of works.

12 First, in [24,25], the input model is simplified to *real-time automata* [16],
 13 a restricted formalism compared to TAs. In this setting, (initial-state) opacity
 14 becomes decidable [24,25].

15 Second, in [5], the authors consider a time-bounded notion of the opacity
 16 of [15], where the attacker has to disclose the secret before an upper bound, using
 17 a partial observability. This can be seen as a secrecy with an *expiration date*. In
 18 addition, the analysis is carried over a time-bounded horizon. The authors prove
 19 that this problem is decidable for TAs.

20 Third, in [9,8], the authors present an alternative definition to Cassez’ opacity
 21 by studying *execution-time opacity*: the attacker has only access to the execu-
 22 tion time of the system, as opposed to Cassez’ partial observations with some
 23 observable events (with their timestamps). In that case, most problems become
 24 decidable (see [7] for a survey).

25 Regarding non-interference for TAs, some decidability results are proved
 26 in [11,12,6], while control was considered in [13]. General security problems for
 27 TAs are surveyed in [10].

28 1.2 Contributions

29 Considering the negative results from [15] we have mainly two directions: one can
 30 consider more restrictive classes of automata, or one can limit the capabilities
 31 of the attacker—we address both directions in this work.

32 We address here \exists -opacity (“there exists a pair of runs, one visiting and
 33 one not visiting the private locations set, that cannot be distinguished”), weak
 34 opacity (“for any run visiting the private locations set, there is another run not
 35 visiting it and both cannot be distinguished”) and full opacity (weak opacity,
 36 but with the other direction holding as well).

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

41 Our set of contributions is threefold.

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

3 *Opacity in subclasses of TAs* Throughout the second part of this paper (Sec-
4 tion 5), we consider the same attacker settings as in [15] but for natural subclasses
5 of TAs: first we deal with one-action TAs, then with one-clock TAs (both with
6 and without ε -transitions—which makes a difference in decidability), TAs over
7 discrete time, and a new subclass which we call observable ERAs. Precisely, we
8 show that:

- 9 1. The problem of \exists -opacity is decidable for general TAs and thus for all sub-
10 classes of TAs we consider as well (Section 5.1).
- 11 2. The problems of weak and full opacity are both undecidable for TAs with
12 only one action (Section 5.2) or two clocks (Section 5.3).
- 13 3. These two problems are also undecidable for TAs with a single clock, un-
14 less we forbid ε -transitions, in which case the problems become decidable
15 (Section 5.3).
- 16 4. These two problems are decidable for unrestricted TAs over discrete time
17 (Section 5.4), and for observable ERAs (Section 5.5).

18 These results overall build on existing results from the literature, with rather
19 straightforward proofs. They still allow us to draw a clear border between decid-
20 ability and undecidability. Moreover, we provide the exact complexity for most
21 of the decidable results, which in some cases, complexify the proofs.

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

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

31 1.3 Outline

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

36 2 Preliminaries

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

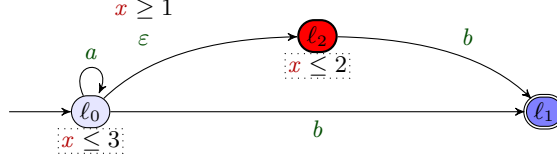


Fig. 1: A TA example

40 We let \mathbb{T} be the domain of the time, which will be either non-negative reals
 1 $\mathbb{R}_{\geq 0}$ (continuous-time semantics) or naturals \mathbb{N} (discrete-time semantics). Unless
 2 otherwise specified, we assume $\mathbb{T} = \mathbb{R}_{\geq 0}$.

3 *Clocks* are real-valued variables that all evolve over time at the same rate.
 4 Throughout this paper, we assume a set $\mathbb{X} = \{x_1, \dots, x_H\}$ of *clocks*. A *clock*
 5 *valuation* is a function $\mu : \mathbb{X} \rightarrow \mathbb{T}$, assigning a non-negative value to each clock.
 6 We write $\mathbf{0}$ for the clock valuation assigning 0 to all clocks. Given a constant
 7 $d \in \mathbb{T}$, $\mu + d$ denotes the valuation s.t. $(\mu + d)(x) = \mu(x) + d$, for all $x \in \mathbb{X}$. If
 8 R is a subset of \mathbb{X} and μ a clock valuation, we call *reset* of the clocks of R and
 9 denote by $[\mu]_R$ the valuation s.t. for all clock $x \in \mathbb{X}$, $[\mu]_R(x) = 0$ if $x \in R$ and
 10 $[\mu]_R(x) = \mu(x)$ otherwise.

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

14 2.1 Timed automata

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

18 **Definition 1 (TA [2]).** A TA \mathcal{A} is a tuple $\mathcal{A} = (\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$,
 19 where: 1) Σ is a finite set of actions, 2) L is a finite set of locations, $\ell_0 \in L$ is
 20 the initial location, 3) $L_{priv} \subseteq L$ is a set of private locations, $L_f \subseteq L$ is a set
 21 of final locations, 4) \mathbb{X} is a finite set of clocks, 5) I is the invariant, assigning
 22 to every $\ell \in L$ a constraint $I(\ell)$ over \mathbb{X} (called invariant), 6) E is a finite set
 23 of edges $e = (\ell, g, a, R, \ell')$ where $\ell, \ell' \in L$ are the source and target locations,
 24 $a \in \Sigma \cup \{\varepsilon\}$ (where ε denotes an unobservable action), $R \subseteq \mathbb{X}$ is a set of clocks
 25 to be reset, and g is a constraint over \mathbb{X} (called guard).

26 *Example 1.* In Fig. 1, we give an example of a TA with three locations ℓ_0 , ℓ_1
 27 and ℓ_2 , three edges, two action $\{a, b\}$, and one clock x . ℓ_0 is the initial location,
 28 ℓ_2 is the (unique) private location, and ℓ_1 is the (unique) final location. ℓ_0 has
 29 an invariant “ $x \leq 3$ ” and the edge from ℓ_0 to ℓ_2 has a guard “ $x \geq 1$ ”.

30 **Definition 2 (Semantics of a TA).** Given a TA $\mathcal{A} =$
 31 $(\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$, the semantics of \mathcal{A} is given by the timed transition
 32 system $\mathfrak{T}_{\mathcal{A}} = (\mathfrak{S}, \mathfrak{s}_0, \Sigma \cup \{\varepsilon\} \cup \mathbb{R}_{\geq 0}, \rightarrow)$, with

- 33 1. $\mathfrak{S} = \{(\ell, \mu) \in L \times \mathbb{R}_{\geq 0}^{\mathbb{X}} \mid \mu \models I(\ell)\}$, $\mathfrak{s}_0 = (\ell_0, \mathbf{0})$,
 1 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)
 2 delay transition relations:
 3 (a) discrete transitions: $((\ell, \mu), e, (\ell', \mu')) \in \rightarrow$, and we write $(\ell, \mu) \xrightarrow{e} (\ell', \mu')$,
 4 if $(\ell, \mu), (\ell', \mu') \in \mathfrak{S}$, $e = (\ell, g, a, R, \ell') \in E$, $\mu' = [\mu]_R$, and $\mu \models g$.
 5 (b) delay transitions: $((\ell, \mu), d, (\ell, \mu + d)) \in \rightarrow$, and we write $(\ell, \mu) \xrightarrow{d} (\ell, \mu +$
 6 $d)$, if $d \in \mathbb{R}_{\geq 0}$ and $\forall d' \in [0, d], (\ell, \mu + d') \in \mathfrak{S}$.

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

9 Given a TA \mathcal{A} with semantic $(\mathfrak{S}, \mathfrak{s}_0, \Sigma \cup \{\varepsilon\} \cup \mathbb{R}_{\geq 0}, \rightarrow)$, we refer to the el-
 10 ements of \mathfrak{S} as the *configurations* of \mathcal{A} . A (finite) *run* of \mathcal{A} is an alternating
 11 sequence of configurations of \mathcal{A} and pairs of delays and edges starting from
 12 the initial configuration \mathfrak{s}_0 and ending in a final configuration (i.e., whose loca-
 13 tion is final), of the form $(\ell_0, \mu_0), (d_0, e_0), (\ell_1, \mu_1), \dots, (\ell_n, \mu_n)$ for some $n \in \mathbb{N}$,
 14 with $\ell_n \in L_f$ and for $i = 0, 1, \dots, n-1$, $\ell_i \notin L_f$, $e_i \in E$, $d_i \in \mathbb{R}_{\geq 0}$, and
 15 $(\ell_i, \mu_i) \xrightarrow{(d_i, e_i)} (\ell_{i+1}, \mu_{i+1})$. A *path* of \mathcal{A} is a prefix of a run ending with a config-
 16 uration.

17 2.2 Region automaton

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

20 Given a TA \mathcal{A} and its set of clocks \mathbb{X} , we define $M : \mathbb{X} \rightarrow \mathbb{N}$ the map that
 21 associates to a clock x the greatest value to which the interpretations of x are
 22 compared within the guards and invariants; if x appears in no constraint, we set
 23 $M(x) = 0$.

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

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

- 30 1. $\lfloor \mu(x) \rfloor = \lfloor \mu'(x) \rfloor$;
 31 2. $\text{frac}(\mu(x)) = 0$ if and only if $\text{frac}(\mu'(x)) = 0$;
 32 3. for each $y \in \mathbb{X}$, $\text{frac}(\mu(x)) \leq \text{frac}(\mu(y))$ if and only if $\text{frac}(\mu'(x)) \leq$
 33 $\text{frac}(\mu'(y))$.

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

37 The equivalence class of a valuation μ is denoted $[\mu]$ and is called a *clock*
 38 *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 in 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 to 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 [8]). *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 which 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 $a \in \Sigma \cup \{\varepsilon\}$:*

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

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 $i \in \llbracket 0; n-1 \rrbracket, r_i \notin R_f$, 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*.

35 Conversely, we say that L_{priv} is *avoided in* ρ if the run ρ does not visit L_{priv} .
 1 We denote the set of the runs avoiding L_{priv} by $Visit^{\overline{priv}}(\mathcal{A})$, referring to them
 2 as *public* runs.

3 A timed word is a sequence of pairs made of an action and a non-decreasing
 4 timestamp in $\mathbb{R}_{\geq 0}$. We denote by $TW^*(\Sigma)$ the set of all finite timed words
 5 on the alphabet Σ . A run ρ on a TA \mathcal{A} defines a timed word: if ρ is of
 6 the form $(\ell_0, \mu_0), (d_0, e_0), (\ell_1, \mu_1), \dots, (\ell_n, \mu_n)$ where for each $i \in \llbracket 0; n-1 \rrbracket$,
 7 $e_i = (\ell_i, g_i, a_i, R_i, \ell_{i+1})$ and $a_i \in \Sigma \cup \{\varepsilon\}$, then it generates the timed
 8 word $(a_{j_0}, \sum_{i=0}^{j_0} d_i)(a_{j_1}, \sum_{i=0}^{j_1} d_i) \dots (a_{j_m}, \sum_{i=0}^{j_m} d_i)$, where $j_0 < j_1 < \dots < j_m$ and
 9 $\{j_k \mid k \in \llbracket 0; m \rrbracket\} = \{i \in \llbracket 0; n-1 \rrbracket \mid a_i \neq \varepsilon\}$. We denote by $Tr(\rho)$ and call *trace*
 10 of ρ the timed word generated by the run ρ and, by extension, given a set of
 11 runs Ω , we denote by $Tr(\Omega)$ the set of the traces of runs in Ω .

12 The set of timed words recognized by a TA \mathcal{A} is the set of traces generated
 13 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
 14 shorten these notations, we use $Tr(\mathcal{A})$ for the set of timed words recognized
 15 by \mathcal{A} , also called *language* of \mathcal{A} . Similarly, we use $Tr^{priv}(\mathcal{A}) = Tr(Visit^{priv}(\mathcal{A}))$
 16 to denote the set of traces of private runs, and $Tr^{\overline{priv}}(\mathcal{A}) = Tr(Visit^{\overline{priv}}(\mathcal{A}))$ for
 17 the set of traces of public runs.

18 In Cassez's original definition [15], actions were partitioned into two sets,
 19 depending on whether an attacker could observe them or not. For simplicity,
 20 here we replaced every unobservable transition in \mathcal{A} by ε -transitions. Projecting
 21 the sequence of actions in a run onto the observable actions, as done by Cassez,
 22 is equivalent to replacing these actions by ε and taking the trace of the run.
 23 Therefore, with respect to opacity, our model is equivalent to [15].

24 3.2 Defining timed opacity

25 In this section, a definition of timed opacity based on the one from [15] is intro-
 26 duced, with three variants inspired by [7]: existential, full and weak opacity. If
 27 the attacker observes a set of runs of the system (i.e., observes their associated
 28 traces), we do not want them to deduce whether L_{priv} was visited or not during
 29 these observed runs. Opacity holds when the traces can be produced by both
 30 private and public runs.

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

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

\exists -opacity decision problem:

37 INPUT: A TA \mathcal{A}

PROBLEM: Is \mathcal{A} \exists -opaque?

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

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

Full (resp. weak) opacity decision problem:

8 INPUT: A TA \mathcal{A}

PROBLEM: Is \mathcal{A} fully (resp. weakly) opaque?

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

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

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

9 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
 10 $(b, 1.5) \in Tr^{priv}(\mathcal{A})$.

11 4 Inter-reducibility of weak and full opacity

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

15 4.1 \mathcal{A}_{priv} and \mathcal{A}_{pub}

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

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

21 The private runs TA \mathcal{A}_{priv} is obtained by duplicating all locations and transi-
 22 tions of \mathcal{A} : one copy \mathcal{A}_S corresponds to the paths that already visited the private
 23 locations set, and the other copy $\mathcal{A}_{\bar{S}}$ corresponds to the paths that did not (this
 24 is a usual way to encode a Boolean, here “ L_{priv} was visited”, in the locations
 25 of a TA). For each private location ℓ_{priv} in \mathcal{A} we copy all transitions leading
 26 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
 27 initial location is the one from $\mathcal{A}_{\bar{S}}$ and the final locations are the ones from \mathcal{A}_S .
 28 Hence all runs need to go from $\mathcal{A}_{\bar{S}}$ to \mathcal{A}_S before reaching a final location, which
 29 requires visiting a private location.

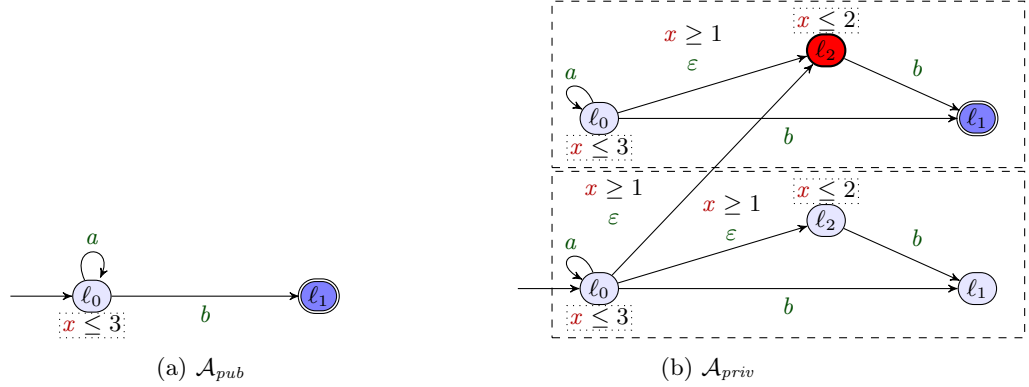


Fig. 2: \mathcal{A}_{pub} and \mathcal{A}_{priv} with the example from Fig. 1

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 $\mathcal{A}_{priv} = (\Sigma, L_S \uplus L_{\bar{S}}, \ell_0^S, L_{priv}^S, L_f^S, \mathbb{X}, I', E')$ 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. $E' = E_S \uplus E_{\bar{S}} \uplus E_{\bar{S} \rightarrow S}$ where E_S and $E_{\bar{S}}$ are the two disjoint copies of E respectively associated to the sets of locations L_S and $L_{\bar{S}}$, and $E_{\bar{S} \rightarrow S}$ is a copy of the set of all transitions that go toward L_{priv}^S where the target location $\ell_{priv}^{\bar{S}}$ has been changed into ℓ_{priv}^S . More formally:

$$\begin{aligned}
 E_S &= \{(\ell^S, g, a, R, \ell'^S) \mid (\ell, g, a, R, \ell') \in E\} \\
 E_{\bar{S}} &= \{(\ell^{\bar{S}}, g, a, R, \ell'^{\bar{S}}) \mid (\ell, g, a, R, \ell') \in E\} \\
 E_{\bar{S} \rightarrow S} &= \{(\ell^{\bar{S}}, g, a, R, \ell_{priv}^S) \mid (\ell, g, a, R, \ell_{priv}) \in E\}.
 \end{aligned}$$

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 $\{\ell_f^S \mid \ell_f \in L_f\}$ 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.

20 4.2 Inter-reducibility of weak and full opacity

1 While the distinction between weak and full notions of opacity can lead to mean-
 2 ingful changes [7], within our framework both associated problems are inter-
 3 reducible.

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

6 *Proof.* Let us first show that the full opacity decision problem reduces to the
 7 weak opacity decision problem. Let \mathcal{A} be a TA. In order to test whether \mathcal{A} is fully
 8 opaque, we can test both inclusions: $Tr^{priv}(\mathcal{A}) \subseteq Tr^{\overline{priv}}(\mathcal{A})$ and $Tr^{priv}(\mathcal{A}) \supseteq$
 9 $Tr^{\overline{priv}}(\mathcal{A})$. The first inclusion can be decided directly by testing whether \mathcal{A} is
 10 weakly opaque. In order to test the second inclusion, we need to build a TA
 11 \mathcal{A}_{memo} where private and public runs are inverted. To do so, we first build \mathcal{A}_{pub}
 12 and \mathcal{A}_{priv} and then define \mathcal{A}' as the TA constituted of \mathcal{A}_{pub} and \mathcal{A}_{priv} as well as
 13 two new locations ℓ'_0 and ℓ'_{priv} . The location ℓ'_0 is the initial location of \mathcal{A}' and
 14 ℓ'_{priv} is the only private location. For $x \in \mathbb{X}$, both ℓ'_0 and ℓ'_{priv} have the invariant
 15 $x = 0$, ensuring no time may elapse in those locations. From ℓ'_0 , with a transition
 16 labeled by ε , one may reach either the initial location of \mathcal{A}_{priv} (ℓ_0^S) or ℓ'_{priv} , from
 17 which an ε -transition leads to the initial location of \mathcal{A}_{pub} (ℓ_0). The final locations
 18 of \mathcal{A}' are the final locations of \mathcal{A}_{pub} and \mathcal{A}_{priv} . The public runs of \mathcal{A}' are the
 19 ones starting in ℓ'_0 , going immediately to ℓ_0^S , and then following a run of \mathcal{A}_{priv}
 20 until a final location of \mathcal{A}_{priv} is reached. As the initial transition is labeled by ε ,
 21 we have $Tr^{\overline{priv}}(\mathcal{A}') = Tr^{priv}(\mathcal{A})$. Similarly, the private runs of \mathcal{A}' are the ones
 22 starting in ℓ'_0 , going immediately to ℓ'_{priv} followed immediately by going to ℓ_0^S ,
 23 and then follows a run of \mathcal{A}_{pub} until a final location of \mathcal{A}_{pub} is reached. As the
 24 two initial transitions are labeled by ε , we have $Tr^{priv}(\mathcal{A}') = Tr^{\overline{priv}}(\mathcal{A})$. Hence,
 25 \mathcal{A} is fully opaque if and only if \mathcal{A} and \mathcal{A}' are weakly opaque.

Let us now show the converse reduction. Let \mathcal{A} be a TA. We will define a
 TA \mathcal{A}' such that \mathcal{A}' is fully opaque if and only if \mathcal{A} is weakly opaque. To do so,
 we want that $Tr^{\overline{priv}}(\mathcal{A}') = Tr^{\overline{priv}}(\mathcal{A})$ and $Tr^{priv}(\mathcal{A}') = Tr^{\overline{priv}}(\mathcal{A}) \cup Tr^{priv}(\mathcal{A})$.
 Indeed, if these equalities hold, $Tr^{priv}(\mathcal{A}') = Tr^{\overline{priv}}(\mathcal{A}')$ 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{A}' 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{A}' 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{A}' are the final locations of \mathcal{A}_{pub} and \mathcal{A}_{priv} .
 The public runs of \mathcal{A}' 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 $Tr^{\overline{priv}}(\mathcal{A}') = Tr^{\overline{priv}}(\mathcal{A})$. Similarly, the
 private runs of \mathcal{A}' 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 $Tr^{priv}(\mathcal{A}') = Tr^{priv}(\mathcal{A}) \cup Tr^{\overline{priv}}(\mathcal{A})$. Hence, \mathcal{A} is weakly opaque if and only if \mathcal{A}' is fully opaque. \square

26 5 Opacity problems for subclasses of timed automata

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

6 5.1 \exists -opacity problem

7 We show here (see Appendix B) that in general the \exists -opacity problem is PSPACE-
 8 complete relying on the reachability problem in TAs, which is known to be
 9 PSPACE-complete [2] as well, even for TA with two clocks [17]. This theorem
 10 considers multiple subclasses of TAs as well that we will describe more in depth
 11 in future sections.

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

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

18 5.2 Timed automata with a single action

19 Recall that the universality problem consists in deciding whether a TA \mathcal{A} accepts
 20 the set of all timed words. In [23], it is shown that the class of one-action TAs
 21 is one of the simplest cases for which the universality problem is undecidable
 22 among TAs. Therefore, this gives the intuition (see Appendix C for proof) that
 23 the weak and full opacity problems are undecidable as well for one-action TAs
 24 ($|\Sigma| = 1$).

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

27 *Remark 2.* The problems of execution-time opacity introduced in [7] are a
 28 particular *decidable* subcase of these undecidable opacity problems with one-
 29 action TAs. Indeed, the execution time is equivalent to a *unique* timestamp
 30 associated to the last action of the system.

31 5.3 Timed automata with a single clock

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

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

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

7 **Without ε -transitions** We now prove that the weak and full opacity problems
8 become both decidable in the context of one-clock TAs ($|\mathbb{X}| = 1$) *without* ε -
9 transitions, relying on the fact that the language inclusion problem for one-
10 clock TAs without ε -transitions is decidable [23].

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

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

18 Note however that, while decidable, this problem cannot be effectively solved
19 currently as the algorithm given by [23] is non-primitive recursive.

20 In addition, due to the undecidability of language universality for TAs with
21 at least two clocks [23, Theorem 21], we can prove the following with the same
22 construction as in Theorem 3:

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

24 5.4 Timed automata over discrete time

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

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

36 Over continuous time, for each run of the TA, there is a unique corresponding
37 run of the region automaton. Over discrete time, thanks to the simplified form

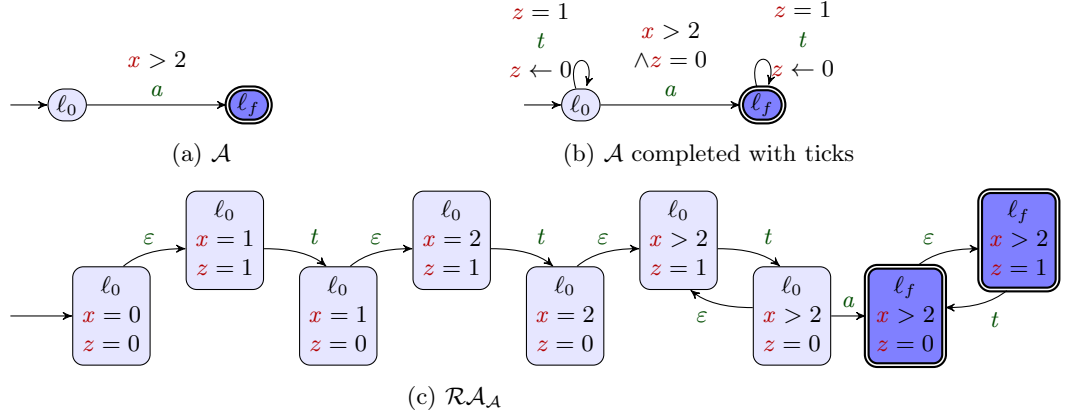


Fig. 3: A discrete-time region automaton example

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 become 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
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. This

27 result was indirectly proven recently in [19], our contribution hence lies mainly
1 in establishing exact complexity.

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

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

6 **Theorem 7.** *Weak and full opacity of discrete-time TAs are EXPSPACE-*
7 *complete.*

8 5.5 Observable Event-Recording Automata

9 In [15], the opacity problems were shown to be undecidable for Event-Recording
10 Automata (ERAs) [3], a subclass of TAs where every clock x is associated to a
11 specific event a_x and x is reset on a transition iff this transition is labeled by a_x .
12 Due to this, the valuations of clocks are entirely determined by the duration
13 since the last occurrence of the associated events. One of the main interest of
14 ERAs is that they are determinizable [3]. This determinization is carried out
15 through the standard subset construction.

16 The undecidability result from [15] on ERAs required to make the events a_x
17 unobservable. Hence, in our framework they would be replaced by ε -transitions.
18 We define observable ERAs (oERAs) as ERAs where the actions resetting the
19 clocks must be observable. This means that the information required for the
20 determinization now belongs to the trace that is observed.

21 Given an oERA \mathcal{A} , we can thus build through the subset construction a TA
22 $Det_{\mathcal{A}}$ such that any path ρ in \mathcal{A} corresponds to a path ρ_D in $Det_{\mathcal{A}}$ with the
23 same trace and ending in a location labeled by the set of all the locations of \mathcal{A}
24 that can be reached with a run that has the same trace as ρ . This information,
25 combined with the construction of \mathcal{A}_{memo} (Remark 1) which stores in the state
26 of the TA whether the private location was visited or not, provides the following
27 result (see Appendix E).

28 **Theorem 8.** *Weak and full opacity are PSPACE-complete for oERAs.*

29 6 Opacity with limited attacker budget

30 One of the causes for the undecidability of the opacity problems in [15] stems
31 from the unbounded memory the attacker might require to remember a run of
32 the TA. As a consequence, one can wonder whether the opacity problems remain
33 undecidable when the attacker performs only a *finite* number of observations.
34 This models the case of an attacker with a limited attack budget. In this section,
35 we prove that the weak and full opacity problems become decidable whenever,
36 given $N \in \mathbb{N}$, the attacker only observes the first N actions (with their times-
37 tamps). To the best of our knowledge, this is *i*) the second result of the literature

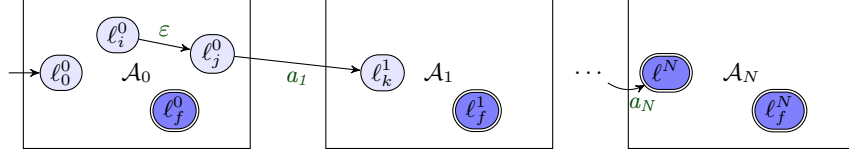


Fig. 4: The construction on an N -observation unfolded TA

38 (after [9]) providing a decidable opacity result for the full class of TAs over dense
 1 time, and *ii*) the first result limiting the number of observations of an attacker
 2 in the context of opacity for TAs.

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

8 Formally, and in order to define new variants of opacity representing this
 9 framework, given a TA \mathcal{A} , we define a new TA (depicted in Fig. 4) which emulates
 10 the behavior of \mathcal{A} up to the N th observation. This TA is an unfolding of \mathcal{A} with
 11 $N + 1$ copies of \mathcal{A} , where ε -transitions are taken within each copy, and transitions
 12 with an observable action lead to the next copy. A run ends when either a final
 13 location or the final copy is reached.

14 **Definition 9 (N -observation unfolding of a TA).**

15 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 -
 16 unfolding of \mathcal{A} the TA $Unfold_N(\mathcal{A}) = (\Sigma, L', \ell_0^0, L'_{priv}, L'_f, \mathbb{X}, I', E')$ where

- 17 1. $L' = \bigcup_{i=0}^N L^i$ where the sets L^i are $N + 1$ disjoint copies of L where each
 18 location $\ell \in L$ has been renamed $\ell^i \in L^i$: for $0 \leq i \leq N$, $L^i = \{\ell^i \mid \ell \in L\}$;
- 19 2. $\ell_0^0 \in L^0$ is the initial location;
- 20 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
 21 of \mathcal{A} ;
- 22 4. $L'_f = (\bigcup_{i=0}^N L^i_f) \cup L^N$ where L^i_f are the copies within L^i of the final locations
 23 of \mathcal{A} ;
- 24 5. $I'(\ell^i) = I(\ell)$ for $\ell \in L$ and $i \leq N$ extends I to each L^i ;
- 25 6. $E' = \bigcup_{i=0}^{N-1} E^i \cup E^{i \rightarrow i+1}$ is the set of transitions where, given $0 \leq i < N$
 26 $- E^i = \{(\ell^i, \varepsilon, g, R, \ell^i) \mid (\ell, \varepsilon, g, R, \ell') \in E\}$;
- 27 $- E^{i \rightarrow i+1} = \{(\ell^i, a, g, R, \ell^{i+1}) \mid (\ell, a, g, R, \ell') \in E \wedge a \in \Sigma\}$.

28 **Definition 10 (Opacity w.r.t. N observations).** Let \mathcal{A} be a TA and let
 29 $N \in \mathbb{N}$. We say that \mathcal{A} is weakly (resp. fully) opaque w.r.t. N observations when
 30 $Unfold_N(\mathcal{A})$ is weakly (resp. fully) opaque.

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

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

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

6 *Proof (sketch of proof, see Appendix F for full proof).* \exists -opacity can be checked
 7 in PSPACE through the same approach as Theorem 2. Indeed, even if N is given
 8 in binary, and thus $Unfold_N(\mathcal{A})$ is of exponential size, the region automaton
 9 of $Unfold_N(\mathcal{A})$ remains simply exponential in the size of \mathcal{A} . Hardness can be
 10 achieved with $N = 0$ with the same method as Theorem 2.

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

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

22 Additionally, we enrich this TA with ticks. In Section 5.4, we added a single
 23 tick to the automaton which counted the time elapsed since the start of the run.
 24 Here, the TA includes as well, for each $0 < k \leq N$, a tick clock counting the
 25 time elapsed since the k 'th observation. As multiple ticks may need to occur at
 26 the same time, we develop the alphabet of ticks to describe the set of tick clocks
 27 that need to be reset, i.e., the tick $t_{\{k_1, \dots, k_m\}}$ is produced by the TA if for every
 28 $0 \leq i \leq m$, the k_i 'th observation (or the start of the run if $k_i = 0$) occurred an
 29 integer number of time units beforehand.

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

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

36 7 Conclusion and perspectives

1 In this paper, we addressed three definitions of opacity on subclasses of TAs, to
 2 circumvent the undecidability from [15]. We first proved the inter-reducibility of
 3 weak and full opacity. Then, while undecidability remains for one-action TAs,
 4 we retrieve decidability for one-clock TAs without ε -transitions, or over discrete
 5 time, or for observable ERAs. Our result for one-clock TAs without ε -transitions
 6 is tight, since we showed that increasing the number of clocks or adding ε -
 7 transitions leads to undecidability. Finally, we studied the case of an attacker
 8 with an observational power with a limited budget, i.e., that can only perform a
 9 finite set of observations. We proved this latter case to be decidable on the full
 10 TA formalism. We summarize the results from Section 5 in Table 1.

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

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

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

Subclass	\exists -opacity	weak opacity	full opacity
$ \Sigma = 1$		\times Theorem 3	
$ \mathbb{X} = 1$ without ε-transitions		\checkmark Theorem 5 (PSPACE-c)	
$ \mathbb{X} = 1$	\checkmark Theorem 2 (PSPACE-c)	\times Theorem 4	
$ \mathbb{X} = 2$		\times Theorem 6	
$\mathbb{T} = \mathbb{N}$		\checkmark Theorem 7 (EXPSPACE-c)	
oERAs		\checkmark Theorem 8 (PSPACE-c)	

18 References

- 19 [1] Abdulla, P.A., Deneux, J., Ouaknine, J., Quaas, K., Worrell, J.: Universality anal-
 20 ysis for one-clock timed automata. FI **89**(4), 419–450 (2008)
- 21 [2] Alur, R., Dill, D.L.: A theory of timed automata. TCS **126**(2), 183–235 (Apr
 22 1994). [https://doi.org/10.1016/0304-3975\(94\)90010-8](https://doi.org/10.1016/0304-3975(94)90010-8)
- 23 [3] Alur, R., Fix, L., Henzinger, T.A.: Event-clock automata: A deter-
 24 minizable class of timed automata. TCS **211**(1-2), 253–273 (1999).
 25 [https://doi.org/10.1016/S0304-3975\(97\)00173-4](https://doi.org/10.1016/S0304-3975(97)00173-4)
- 26 [4] Alur, R., Henzinger, T.A., Vardi, M.Y.: Parametric real-time reasoning. In:
 27 Kosaraju, S.R., Johnson, D.S., Aggarwal, A. (eds.) STOC. pp. 592–601. ACM,
 28 New York, NY, USA (1993). <https://doi.org/10.1145/167088.167242>

- 29 [5] Ammar, I., El Touati, Y., Yeddes, M., Mullins, J.: Bounded opacity for timed
1 systems. *Journal of Information Security and Applications* **61**, 1–13 (Sep 2021).
2 <https://doi.org/10.1016/j.jisa.2021.102926>
- 3 [6] André, É., Kryukov, A.: Parametric non-interference in timed au-
4 tomata. In: Li, Y., Liew, A. (eds.) ICECCS. pp. 37–42 (2020).
5 <https://doi.org/10.1109/ICECCS51672.2020.00012>
- 6 [7] André, É., Lefauchaux, E., Lime, D., Marinho, D., Sun, J.: Configuring timing pa-
7 rameters to ensure execution-time opacity in timed automata. In: ter Beek, M.H.,
8 Dubslaff, C. (eds.) TiCSA. *Electronic Proceedings in Theoretical Computer Sci-*
9 *ence*, vol. 392, pp. 1–26 (2023). <https://doi.org/10.4204/EPTCS.392.1>, invited
10 paper.
- 11 [8] André, É., Lefauchaux, E., Marinho, D.: Expiring opacity problems in parametric
12 timed automata. In: Ait-Ameur, Y., Khendek, F. (eds.) ICECCS. pp. 89–98 (2023).
13 <https://doi.org/10.1109/ICECCS59891.2023.00020>
- 14 [9] André, É., Lime, D., Marinho, D., Sun, J.: Guaranteeing timed opacity using
15 parametric timed model checking. *ACM Transactions on Software Engineering*
16 *and Methodology* **31**(4), 1–36 (Oct 2022). <https://doi.org/10.1145/3502851>
- 17 [10] Arcile, J., André, É.: Timed automata as a formalism for expressing security: A
18 survey on theory and practice. *ACM Computing Surveys* **55**(6), 1–36 (Jul 2023).
19 <https://doi.org/10.1145/3534967>
- 20 [11] Barbuti, R., Francesco, N.D., Santone, A., Tesei, L.: A notion of non-interference
21 for timed automata. *FI* **51**(1-2), 1–11 (2002)
- 22 [12] Barbuti, R., Tesei, L.: A decidable notion of timed non-interference. *FI* **54**(2-3),
23 137–150 (2003)
- 24 [13] Benattar, G., Cassez, F., Lime, D., Roux, O.H.: Control and synthesis of non-
25 interferent timed systems. *International Journal of Control* **88**(2), 217–236 (2015).
26 <https://doi.org/10.1080/00207179.2014.944356>
- 27 [14] Bryans, J.W., Koutny, M., Mazaré, L., Ryan, P.Y.A.: Opacity generalised to tran-
28 sition systems. *International Journal of Information Security* **7**(6), 421–435 (2008).
29 <https://doi.org/10.1007/s10207-008-0058-x>
- 30 [15] Cassez, F.: The dark side of timed opacity. In: Park, J.H., Chen, H., Atiquzzaman,
31 M., Lee, C., Kim, T., Yeo, S. (eds.) ISA. LNCS, vol. 5576, pp. 21–30. Springer
32 (2009). https://doi.org/10.1007/978-3-642-02617-1_3
- 33 [16] Dima, C.: Real-time automata. *Journal of Automata, Languages and Combina-*
34 *torics* **6**(1), 3–23 (2001). <https://doi.org/10.25596/jalc-2001-003>
- 35 [17] Fearnley, J., Jurdziński, M.: Reachability in two-clock timed automata
36 is pspace-complete. *Information and Computation* **243**, 26–36 (2015).
37 <https://doi.org/https://doi.org/10.1016/j.ic.2014.12.004>, <https://www.sciencedirect.com/science/article/pii/S0890540114001564>, 40th Interna-
38 tional Colloquium on Automata, Languages and Programming (ICALP 2013)
- 39 [18] Gardey, G., Mullins, J., Roux, O.H.: Non-interference control syn-
40 thesis for security timed automata. *ENTCS* **180**(1), 35–53 (2007).
41 <https://doi.org/10.1016/j.entcs.2005.05.046>
- 42 [19] Klein, J., Kogel, P., Glesner, S.: Verifying opacity of discrete-timed automata.
43 In: Plat, N., Gnesi, S., Furia, C.A., Lopes, A. (eds.) FormaliSE. pp. 55–65. ACM
44 (2024). <https://doi.org/10.1145/3644033.3644376>
- 45 [20] Li, J., Lefebvre, D., Hadjicostis, C.N., Li, Z.: Observers for a class of timed au-
46 tomata based on elapsed time graphs. *IEEE Transactions on Automatic Control*
47 **67**(2), 767–779 (2022). <https://doi.org/10.1109/TAC.2021.3064542>
- 48 [21] Mazaré, L.: Using unification for opacity properties. In: Ryan, P. (ed.) WITS. pp.
49 165–176 (Apr 2004)
- 50

- 51 [22] Meyer, A.R., Stockmeyer, L.J.: The equivalence problem for regular expressions
1 with squaring requires exponential space. In: IEEE. pp. 125–129 (1972), <https://api.semanticscholar.org/CorpusID:206585190>
2
- 3 [23] Ouaknine, J., Worrell, J.: On the language inclusion problem for
4 timed automata: Closing a decidability gap pp. 54–63 (2004).
5 <https://doi.org/10.1109/LICS.2004.1319600>
- 6 [24] Wang, L., Zhan, N.: Decidability of the initial-state opacity of real-time au-
7 tomata. In: Jones, C.B., Wang, J., Zhan, N. (eds.) Symposium on Real-Time
8 and Hybrid Systems - Essays Dedicated to Professor Chaochen Zhou on the
9 Occasion of His 80th Birthday, LNCS, vol. 11180, pp. 44–60. Springer (2018).
10 https://doi.org/10.1007/978-3-030-01461-2_3
- 11 [25] Wang, L., Zhan, N., An, J.: The opacity of real-time automata. IEEE Transactions
12 on Computer-Aided Design of Integrated Circuits and Systems **37**(11), 2845–2856
13 (2018). <https://doi.org/10.1109/TCAD.2018.2857363>
- 14 [26] Zhang, K.: State-based opacity of labeled real-time automata. TCS **987**, 114373
15 (2024). <https://doi.org/10.1016/J.TCS.2023.114373>

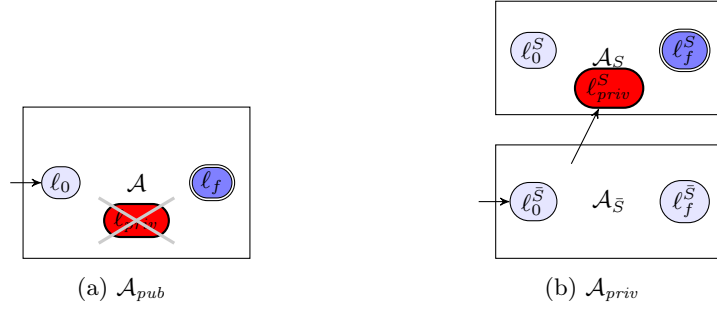


Fig. 5: Illustrating \mathcal{A}_{pub} and \mathcal{A}_{priv}

16 A Formal definitions

1 **Definition 11 (Public runs automaton \mathcal{A}_{pub}).**

2 Let $\mathcal{A} = (\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$ be a TA. We define the public runs TA
 3 $\mathcal{A}_{pub} = (\Sigma, L \setminus L_{priv}, \emptyset, L_f \setminus L_{priv}, \mathbb{X}, I', E')$ with I' and E' precised as follows:

- 4 1. I' is the restriction $I|_{L \setminus L_{priv}}$ of I to the set of locations of \mathcal{A}_{pub} ;
- 5 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
 6 transitions when private locations are removed from L .

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

8 B Complexity of the \exists -opacity problem

9 B.1 \exists -opacity problem for general TAs

10 Let us first show that the \exists -opacity problem for TA lies in PSPACE.

11 *Proof.* Let \mathcal{A} be a TA. We build \mathcal{A}_{priv} and \mathcal{A}_{pub} from \mathcal{A} as described in Sec-
 12 tion 4.1. Noting that the product of two TAs recognizes the intersection of
 13 their languages [2, Theorem 3.15] (assuming the two TAs share no clock), we
 14 build the TA $\mathcal{A}_{priv} \times \mathcal{A}_{pub}$, product of \mathcal{A}_{priv} and \mathcal{A}_{pub} , which language is
 15 $Tr^{priv}(\mathcal{A}) \cap Tr^{\overline{priv}}(\mathcal{A})$. To build this product, we can rename all clocks from
 16 \mathcal{A}_{pub} so that \mathcal{A}_{priv} and \mathcal{A}_{pub} share no clock.

The \exists -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 \exists -opacity problem. \square

17 We now reduce the reachability problem for timed automata, known to be
 18 PSPACE-complete, to the \exists -opacity problem.

19 *Proof.* Let $\mathcal{A} = (\Sigma, L, \ell_0, \emptyset, L_f, \mathbb{X}, I, E)$ be a timed automaton.
 1 We suppose that \mathbb{X} is not empty and define (see Fig. 6) $\mathcal{A}' =$
 2 $(\Sigma, L \cup \{\ell'_0, \ell'_1, \ell'_f\}, \ell'_0, L_f, L_f \cup \{\ell'_f\}, \mathbb{X}, I', E')$ where I' an invari-
 3 ant extending I such that $I'(\ell'_0) = I'(\ell'_1) = I'(\ell'_f) = \text{true}$ and
 4 $E' = E \cup \{(\ell'_0, \varepsilon, (x=0), \emptyset, \ell_0), (\ell'_0, \varepsilon, (x=0), \emptyset, \ell'_1), (\ell'_1, \varepsilon, \text{true}, \emptyset, \ell'_f)\} \cup$
 5 $\{(\ell'_f, a, \text{true}, \emptyset, \ell'_f) \mid a \in \Sigma\}$ for some $x \in \mathbb{X}$.

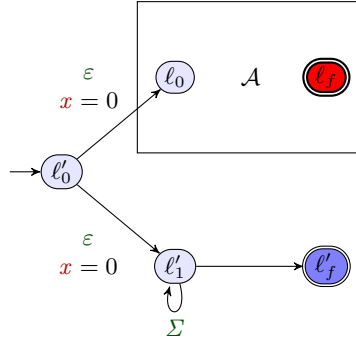


Fig. 6: TA \mathcal{A}' for the PSPACE-hardness of \exists -opacity

6 The timed automaton \mathcal{A}' is \exists -opaque if and only if a final location is reachable
 7 in \mathcal{A} . Indeed, the set $Tr^{\text{priv}}(\mathcal{A}')$ contains all the possible timed traces with the
 8 action set Σ , and the private runs on \mathcal{A}' correspond exactly to runs on \mathcal{A} . Hence
 9 $Tr^{\text{priv}}(\mathcal{A}') \cap Tr^{\text{priv}}(\mathcal{A}') \neq \emptyset$ if and only if $Tr^{\text{priv}}(\mathcal{A}') \neq \emptyset$, i.e., if there is a
 10 run on \mathcal{A} that reaches a final location. Since the reachability problem in TA is
 11 PSPACE-complete, we deduce from this construction that the \exists -opacity problem
 12 is PSPACE-hard.

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

15 B.2 \exists -opacity problem for TAs with a fixed number of clocks

16 Fix N as a constant. We consider now the \exists -opacity problem for TAs with N
 17 clocks.

18 In the previous section, the \exists -opacity problem for TAs was shown to be within
 19 PSPACE. The algorithm reduces the problem to a reachability query on the prod-
 20 uct automaton $\mathcal{A}_{\text{priv}} \times \mathcal{A}_{\text{pub}}$ (a TA with $2N$ clocks). The reachability problem for
 21 TAs is usually solved by studying reachability in the associated region automa-
 22 ton. Reachability in automata being in NLOGSPACE and the region automata
 23 being exponential in general produces the result. More precisely, the number of

24 states of the region automata is bounded by $|L|(N! \cdot 2^N \cdot \prod_{x \in \mathbb{X}} (2M + 2))$ [2] where
 1 M is the highest constant occurring in guards and invariants. Note that, since
 2 N is a constant, this number becomes polynomial when integers in guards and
 3 invariants (and thus M) are given in unary. Hence the reachability problem falls
 4 to NLOGSPACE, which implies the \exists -opacity problem also lies in NLOGSPACE
 5 then.

6 Concerning the hardness, let us show that the \exists -opacity problem remains
 7 PSPACE-hard for one-clock automata with constants in binary. Note that the
 8 reduction of the previous section does not apply, as reachability in TA with
 9 one clock is not PSPACE-hard. We reduce the reachability problem in two-clock
 10 automata, known to be PSPACE-complete [17], to the \exists -opacity problem in one-
 11 clock automata.

12 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
 13 every transition (including silent transitions) of $\mathcal{A}_{x,y}$ with a new alphabet $\Sigma' =$
 14 $\{a_i \mid 1 \leq i \leq |E|\}$ such that each letter of Σ' labels exactly one transition of
 15 $\mathcal{A}_{x,y}$. We denote the obtained automaton by $\mathcal{A}'_{x,y}$.

16 Given a guard g , we define g_x and g_y as respectively the constraints in g over
 17 x and y . Hence, $g = g_x \wedge g_y$. For $z \in \{x, y\}$, we then define the automaton $\mathcal{A}_z =$
 18 $(\Sigma, L, \ell_0, \emptyset, L_f, \{z\}, I_z, E_z)$ with $E_z = \{(\ell, a, g_z, R \cap \{z\}, \ell') \mid (\ell, a, g, R, \ell') \in E\}$
 19 and I_z is similarly obtained by only keeping the z part of the invariant.

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

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

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

36 C Opacity of one-action TAs

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

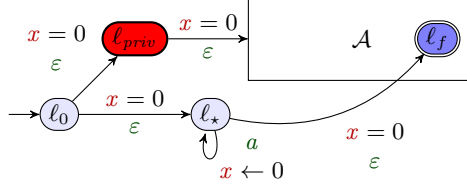


Fig. 7: Automaton \mathcal{B} : Reduction from universality to full opacity

43 an urgent ε -transition) to the initial location of the TA \mathcal{A} and the other leads
 1 to a location where every finite timed words on Σ can be read before reaching
 2 a final location. We denote this TA \mathcal{B} and illustrate its construction in Fig. 7.
 3 The language recognized by \mathcal{A} corresponds exactly to the traces of private runs
 4 of \mathcal{B} , and the traces of public runs of \mathcal{B} are all the finite timed words on Σ .
 5 Therefore, \mathcal{B} is fully opaque iff $Tr^{priv}(\mathcal{B}) = Tr^{\overline{priv}}(\mathcal{B})$ iff $Tr(\mathcal{A}) = TW^*(\Sigma)$ iff
 6 \mathcal{A} is universal. Since universality for TAs with one action is undecidable [23], we
 7 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. \square

8 D Opacity of TAs over discrete time

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

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

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

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

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

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

16 This association is injective as the sequence $(\tau_i)_{i \leq n}$ which was removed
 17 in the transformation depends only on the number of t of the timed
 18 word. Moreover, it is surjective as given an untimed word in $\mathcal{RA}_{\mathcal{A}}$ $w' =$
 19 $\underbrace{tt\dots t}_{k_0 \text{ times}} a_0 \underbrace{tt\dots t}_{k_1 \text{ times}} a_1 \dots \underbrace{tt\dots t}_{k_n \text{ times}} a_n$ produced by a path $[\rho']$ of the region au-
 20 tomaton, 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 to w' . \square

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

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

5 *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
6 respective region automata. Then from Lemma 1, we have

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

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. \square

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

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

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

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

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

23 **Proposition 2.** [22] *Let L_1 and L_2 be two rational languages with square. De-*
24 *termining whether $L_1 = L_2$ is EXPSPACE-complete.*

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

29 Since we will compare the timed language of TAs, and we only want to com-
30 pare their untimed languages, we need to impose a standard for the timestamps

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

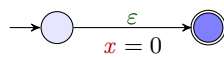
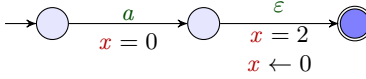
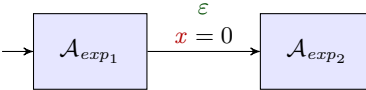
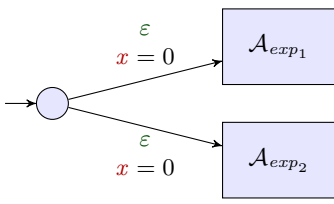
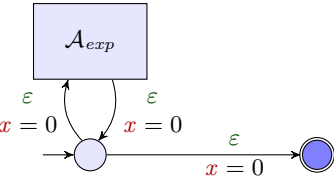
Rational expression with square exp	Timed automaton \mathcal{A}_{exp}
ε	
$a \in \Sigma$	
$exp_1 \cdot exp_2$	
$exp_1 + exp_2$	
exp^*	

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

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

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

13 More precisely, the previously built TA \mathcal{A}_{exp} is modified into $\tilde{\mathcal{A}}_{exp}$ by adding
 14 on every location silent loop transitions resetting y and v when they reach 2, as
 15 well as a silent loop transition with guard $x = 1 \wedge y = 1$ and reset set $\{y, v\}$.
 16 At most one of the latter loops, denoted by ①, is taken during an execution,
 17 and it requires at least one letter to be triggered. This transition ensures that
 18 $y = v \neq x$ in the following. This property is necessary to take the transition ②,
 19 which now ensures that $x = v \neq y$, which will allow taking the transition ③.
 20 As mentionned, taking the transition ① requires at least one letter to be read,
 21 hence why, when exp contains the empty word, we need the clock z to give an
 22 alternative way to exit the gadget. Formally, we have

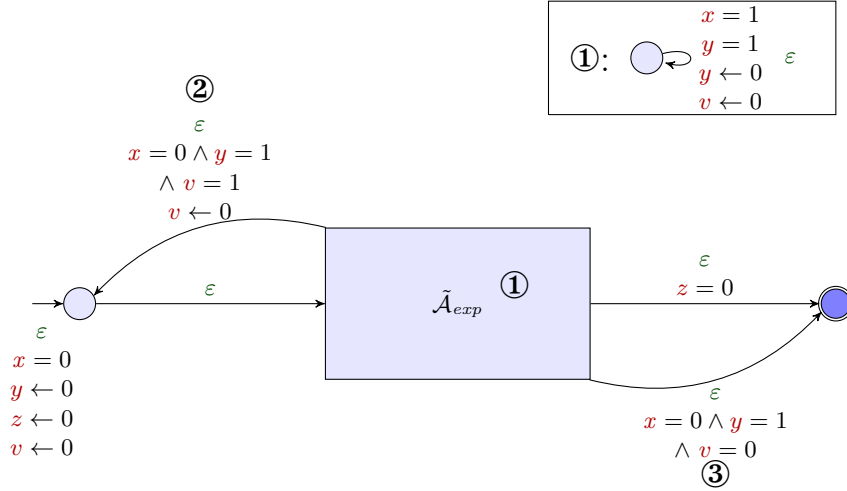


Fig. 9: TA \mathcal{A}_{exp^2}
 $\tilde{\mathcal{A}}_{exp}$ is the automaton \mathcal{A}_{exp} modified through ①.

23 **Definition 13 (Square construction).** Let $\mathcal{A}_{exp} = (\Sigma, L, \ell_0, \emptyset, L_f, \mathbb{X}, I, E)$ be
 1 the timed automaton corresponding to the rational expression with square exp .
 2 Then, the TA corresponding to exp^2 is $\mathcal{A}_{exp^2} = (\Sigma, L \cup \{\ell'_0, \ell'_f\}, \ell'_0, \emptyset, \{\ell'_f\}, \mathbb{X} \cup$
 3 $\{v, y, z\}, I', E')$ where

4 – I' is the extension of I such that $I'(\ell'_0) = I'(\ell'_f) = true$,
 –

$$E' = E \cup \{(\ell'_0, \varepsilon, true, \emptyset, \ell_0)\} \cup$$

$$\bigcup_{\ell \in L} \{(\ell, \varepsilon, y = 2, \{y\}, \ell), (\ell, \varepsilon, v = 2, \{v\}, \ell), (\ell, \varepsilon, x = 1 \wedge y = 1, \{v, y\}, \ell)\} \cup$$

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

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. \square

5 **Theorem 7.** *Weak and full opacity of discrete-time TAs are EXPSPACE-*
 6 *complete.*

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

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

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

19 E PSPACE-completeness of weak / full opacity for oERAs

20 Let us first explain why the algorithm for weak opacity presented in the main
 21 document is in PSPACE. As a summary, this algorithm consists in, given an
 22 oERA \mathcal{A} , building the corresponding \mathcal{A}_{memo} , determining it through the subset

23 construction, taking its region automaton, and then testing reachability of a
 1 location containing a final private location, but no final public location.

2 The determinization of the oERA causes the number of locations to become
 3 exponential in the size of the entry, and the construction of the region automaton
 4 gives an exponential number of clock regions, bounded by $|\mathbb{X}| \cdot 2^{|\mathbb{X}|} \cdot \prod_{x \in \mathbb{X}} (2M(x) +$
 5 $2)[2]$. The size of the region automaton is thus exponential in the number of
 6 locations and the number of clocks of \mathcal{A} . On the region automaton, testing the
 7 reachability of a location can be done in NLOGSPACE. Hence the problem of
 8 weak opacity in oERA is in PSPACE.

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

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

16 These results extend to full opacity thanks to [Theorem 1](#).

17 F Opacity with N observations

18 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
 19 the TA \mathcal{A}_{memo} described in [Remark 1](#). Recall that \mathcal{A}_{memo} recognizes the same
 20 language as \mathcal{A} but stores within the locations the information whether L_{priv}
 21 was visited. As such, \mathcal{A}_{memo} has the same opacity properties as \mathcal{A} , so we can
 22 consider $Unfold_N(\mathcal{A}_{memo})$ instead of $Unfold_N(\mathcal{A})$ to study the opacity of \mathcal{A} .

23 Additionally, we enrich this TA with ticks. In [Section 5.4](#), we added a single
 24 tick to the automaton which counted the time elapsed since the start of the run.
 25 Here, the TA includes as well, for each $0 < k \leq N$, a tick counting the time
 26 elapsed since the k 'th observation. As multiple ticks may need to occur at the
 27 same time, we develop the alphabet of ticks to describe the set of tick clocks
 28 that need to be reset, i.e., the tick $t_{\{k_1, \dots, k_m\}}$ is produced by the TA if for every
 29 $0 \leq i \leq m$, the k_i 'th observation (or the start of the run if $k_i = 0$) occurred an
 30 integer number of time units beforehand. Note that the addition of these ticks
 31 immediately uses the assumption that only N actions are observed.

32 Definition 14 (Addition of ticks to the Unfolding construction).

33 Let $\mathcal{A} = (\Sigma, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$ be a TA, $N \in \mathbb{N}$ and let $Unfold_N(\mathcal{A}) =$
 34 $(\Sigma, L', \ell_0^0, L'_{priv}, L'_f, \mathbb{X}, I', E')$ the unfolding of \mathcal{A} . We define the Tick construction
 35 $Tick(Unfold_N(\mathcal{A})) = (\Sigma', L', \ell_0^0, L'_{priv}, L'_f, \mathbb{X}', I', E'')$ where

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

$$\begin{aligned}
41 \quad 3. \quad E'' &= \bigcup_{i=0}^{N-1} E^i \cup E^{i \rightarrow i+1} \text{ is the set of transitions where, given } 0 \leq i < N \\
1 \quad - E^i &= \{(\ell^i, \varepsilon, g \wedge \bigwedge_{k=0}^i (x_k < 1), R, \ell^i) \mid (\ell, \varepsilon, g, R, \ell') \in E'\} \cup \\
2 \quad &\{(\ell^i, t_K, \bigwedge_{k \in K} (x_k = 1) \wedge \bigwedge_{m \in \llbracket 0; i \rrbracket \setminus K} (0 < x_m < 1), \{x_k \mid k \in K\}, \ell^i) \mid \ell^i \in \\
3 \quad &L^i \wedge K \subseteq \llbracket 0; i \rrbracket \wedge K \neq \emptyset\}; \\
4 \quad - E^{i \rightarrow i+1} &= \{(\ell^i, a^0, g \wedge \bigwedge_{k=0}^i (x_k < 1) \wedge \bigvee_{m=0}^i (x_m = 0), R \cup \{x_{i+1}\}, \ell^{i+1}) \mid \\
5 \quad &(\ell, a, g, R, \ell') \in E'\} \cup \{(\ell^i, a, g \wedge \bigwedge_{k=0}^i (0 < x_k < 1), R \cup \{x_{i+1}\}, \ell^{i+1}) \mid \\
6 \quad &(\ell, a, g, R, \ell') \in E'\}.
\end{aligned}$$

7 We obtain in this way the timed automaton $Tick(Unfold_N(\mathcal{A}_{memo}))$. Let
8 $\mathcal{RA}_{Tick(Unfold_N(\mathcal{A}_{memo}))}$ be the region automaton of this automaton. Thanks to
9 the added ticks, paths of $\mathcal{RA}_{Tick(Unfold_N(\mathcal{A}_{memo}))}$ sharing the same trace correspond
10 to runs of \mathcal{A} for which the (at most) N observations occurred within
11 the same time intervals (due to the tick representing the total time) and the
12 fractional part of the timing of those observations have the same order. This is
13 the information we mainly need, and thus we wish to regroup every path of the
14 region automaton with the same trace. As the region automaton is a finite automaton,
15 we can realize usual operations on it, that is, first remove ε -transitions
16 (by fusing them with the following non- ε -transition) and then determinizing the
17 automaton through the subset construction. We denote by $\mathcal{B}(\mathcal{A})$ the resulting
18 automaton. We call *beliefs* the states of $\mathcal{B}(\mathcal{A})$, i.e., they describe the set of regions
19 the attacker believes the system may be in.

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

25 **Proposition 3 (Relation between opacity and discriminating belief).**

26 *A TA \mathcal{A} is weakly (resp. fully) opaque w.r.t. N observations iff $\mathcal{B}(\mathcal{A})$ does not*
27 *contain any weakly (resp. fully) discriminating belief.*

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

29 – Assume first that $\mathcal{B}(\mathcal{A})$ contains a weakly discriminating belief B . Let r be a
30 region in B_{priv} and w be the trace of a path leading from the initial belief of
31 $\mathcal{B}(\mathcal{A})$ to B . By construction of the region automaton, there exists a run ρ of
32 $Tick(Unfold_N(\mathcal{A}_{memo}))$ whose untimed trace (i.e., the trace of ρ projected
33 on the actions) is w and such that the run corresponding to ρ in the region
34 automaton ends in r . In particular, ρ is a private run. Moreover, any run
35 whose untimed trace is w ends in a region of B . Thus, there is no public run
36 with trace w and in particular $Tr(\rho) \in Tr^{priv}(Tick(Unfold_N(\mathcal{A}_{memo})))$ and
37 $Tr(\rho) \in Tr^{\overline{priv}}(Tick(Unfold_N(\mathcal{A}_{memo})))$, hence $Tick(Unfold_N(\mathcal{A}_{memo}))$ is
38 not weakly opaque and \mathcal{A} is not weakly opaque w.r.t. N observations.

39 – Assume now that \mathcal{A} is not weakly opaque w.r.t. N ob-
 1 servations. Let ρ be a run of $\text{Tick}(\text{Unfold}_N(\mathcal{A}_{memo}))$ such
 2 that $\text{Tr}(\rho) \in \text{Tr}^{priv}(\text{Tick}(\text{Unfold}_N(\mathcal{A}_{memo})))$ and $\text{Tr}(\rho) \notin$
 3 $\text{Tr}^{priv}(\text{Tick}(\text{Unfold}_N(\mathcal{A}_{memo})))$. Let $[\rho]$ be the run corresponding to ρ
 4 in the region automaton.⁵ We denote by $T([\rho])$ the set of traces of runs of
 5 $\text{Tick}(\text{Unfold}_N(\mathcal{A}_{memo}))$ associated to $[\rho]$.

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

- 8 1. $\forall i \in \llbracket 0; m \rrbracket, (a_i \in \Sigma \cup \Sigma_t \implies \tau_i - \tau_{i-1} > 0) \wedge (a_i \in \Sigma^0 \implies \tau_i - \tau_{i-1} = 0)$
 9 (where $\tau_{-1} = 0$), meaning that two consecutive observable actions occur
 10 at the same time if and only if the second one is in Σ^0 .
- 11 2. $\forall i, j \in \llbracket 0; m \rrbracket, \forall J \subseteq \llbracket 0; N \rrbracket, \forall I \subseteq J, (i < j \wedge a_i = t_I \wedge a_j = t_J \wedge \forall k \in$
 12 $\llbracket i+1; j-1 \rrbracket, \forall K \subseteq \llbracket 0; N \rrbracket (a_k = t_K \implies K \cap J = \emptyset)) \implies \tau_j - \tau_i = 1,$
 13 meaning that two successive ticks of the same clocks are separated by
 14 exactly 1 time unit.
- 15 3. $\tau_m \geq 1 \implies (\exists i \in \llbracket 0; m \rrbracket, \exists I \subseteq \llbracket 0; N \rrbracket \forall j < i, a_i = t_I \wedge 0 \in I \wedge \tau_i =$
 16 $1 \wedge (a_j \notin \Sigma_t)),$ meaning that the first occurrence of the tick of the clock
 17 x_0 is at time 1.
- 18 4. $\forall i \in \llbracket 0; m \rrbracket, (a_i \in \Sigma \cup \Sigma^0 \wedge \tau_m - \tau_i \geq 1) \implies (\exists k \in$
 19 $\llbracket 0; m \rrbracket, \exists K \subseteq \llbracket 0; N \rrbracket, a_k = t_K \wedge \{j \in \llbracket 0; i \rrbracket \mid a_j \in \Sigma \cup \Sigma^0\} \in K \wedge \tau_k -$
 20 $\tau_i = 1)$ meaning that each of the N observations is followed by its corre-
 21 sponding tick exactly one time unit after it.
- 22 5. $\forall i \in \llbracket 0; m \rrbracket, \forall I \subseteq \llbracket 0; N \rrbracket, (a_i = t_I \wedge \tau_m - \tau_i \geq 1) \implies \exists j \in \llbracket i+1; m \rrbracket, \exists J \subseteq$
 23 $\llbracket 0; N \rrbracket, (I \subseteq J \wedge a_j = t_J)$ meaning that if a clock ticked and the run is
 24 still at least one time unit long, then there will be a new tick of this clock
 25 within the rest of the run.

26 Due to its size, we postpone the proof of this lemma to the bottom of this
 27 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 to a final
 location, there exists a run ρ' such that $\text{Tr}(\rho) = \text{Tr}(\rho')$ and $[\rho']$ ends in r .
 As $\text{Tr}(\rho) \notin \text{Tr}^{priv}(\text{Tick}(\text{Unfold}_N(\mathcal{A}_{memo})))$ by assumption, we have that r
 is a region associated to a private location. Hence $B_{priv} \neq \emptyset$ and $B_{pub} = \emptyset$,
 thus B is a weakly discriminating belief. \square

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

⁵ The notation $[\cdot]$ represents that $[\rho]$ implicitly defines an equivalence class of runs
 of $\text{Tick}(\text{Unfold}_N(\mathcal{A}_{memo}))$. For a run ρ' of $\text{Tick}(\text{Unfold}_N(\mathcal{A}_{memo}))$, we thus write
 $\rho' \in [\rho]$ to say that the run associated to ρ' in the region automaton is $[\rho]$.

«««< HEAD 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 ===== 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 »»»> 91b7d5290400fe8c6a830f7adc7b28d266cc994d 2-EXPSpace algorithm. If N is given in unary, the complexity falls to EXPSpace. \square

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

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

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

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

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

9 – *Property 1.* Let $i \in \llbracket 0; m \rrbracket$. The set of clocks reset by ε -transitions is included
10 in \mathbb{X} . By definition of the indices j_i , there are only ε -transitions between the
11 configurations $(\ell_{j_{i-1}+1}, \mu_{j_{i-1}+1})$ if $i > 0$ or the initial configuration if $i = 0$,
12 and (ℓ_{j_i}, μ_{j_i}) . Thus, no clock from \mathbb{X}_t is reset among these transitions and
13 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 $a_i \in \Sigma^0$.

14 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
15 some guard g and the copy number $h \leq i$ of $e_{j_i} \in E^{h \rightarrow h+1}$ (in other words,
16 the integer h such that the transition of the original automaton starts in the
17 copy L^h). In particular, there exists a clock $x \in \mathbb{X}_t$ such that $\mu_{j_i}(x) + d_{j_i} = 0$.

18 Therefore we obtain $\mu_{j_{i-1}+1}(x) + \sum_{k=j_{i-1}+1}^{j_i} d_k = 0$, which implies $\tau_i - \tau_{i-1} = 0$.

19 Let us now assume that $a_i \in \Sigma \cup \Sigma_t$, and show that $\tau_i - \tau_{i-1} > 0$. Depending
20 on whether a_i is in Σ or in Σ_t , the guard of the transition e_{j_i} can be of two
21 forms : $g \wedge \bigwedge_{k=0}^h (0 < x_k < 1)$ with g some guard and h some copy number, if

- 22 $a_i \in \Sigma$; or $\bigwedge_{k \in K} (x_k = 1) \wedge \bigwedge_{m \in \llbracket 0; h \rrbracket \setminus K} (0 < x_m < 1)$ with $K \subseteq \llbracket 0; h \rrbracket$ non-empty
 1 and h a copy number, if $a_i \in \Sigma_t$. In both cases, we must have $\mu_{j_i}(x_k) + d_{j_i} > 0$
 2 for each $x_k \in \mathbb{X}_t$ such that $k \leq h$. However the last observable transition
 3 $e_{j_{i-1}}$ resets one of those clocks x , which gives $\mu_{j_{i-1}+1}(x) = 0$. Hence $0 <$
 4 $\mu_{j_i}(x) + d_{j_i} = \mu_{j_{i-1}+1}(x) + \sum_{k=j_{i-1}}^{j_i} d_k = \sum_{k=j_{i-1}}^{j_i} d_k$.
- 5 – *Property 2.* Let $i, i' \in \llbracket 0; m \rrbracket$, with $i < i'$. Assume there are $I' \subseteq \llbracket 0; N \rrbracket$ and
 6 $I \subseteq I'$ such that $a_i = t_I$ and $a_{i'} = t_{I'}$. Suppose that for every $k \in \llbracket i+1; i'-1 \rrbracket$
 7 and for each $K \subseteq \llbracket 0; N \rrbracket$, $a_k = t_K \implies K \cap I' = \emptyset$. Let us show that
 8 $\tau_{i'} - \tau_i = 1$.
- 9 Let $x \in \{x_k \mid k \in I\} \subseteq \mathbb{X}_t$. After the reset applied by e_{j_i} , we have: $\mu_{j_{i+1}} =$
 10 $[\mu_{j_i} + d_{j_i}]_{\{x_k \mid k \in I\}} \models (x = 0)$. In order to take the transition $e_{j_{i'}}$, the clock
 11 valuation $\mu_{j_{i'}} + d_{j_{i'}}$ needs to satisfy the guard $(x = 1)$. It remains to show
 12 that x is not reset between e_{j_i} and $e_{j_{i'}}$. Let e be a transition between e_{j_i}
 13 and $e_{j_{i'}}$. If e is an ε -transition, it does not reset any clock in \mathbb{X}_t . If e is labeled
 14 by a letter in $\Sigma \cup \Sigma^0$, it is in some $E^{h' \rightarrow h'+1}$ with $h' \geq h$ and h the copy
 15 number of the configuration preceding e_{j_i} . The only clock e resets that is not
 16 in \mathbb{X} is $x_{h'+1}$, and cannot be x since $I \subseteq \llbracket 0; h \rrbracket$. Finally, if $e = (\ell, t_K, g, R, \ell')$
 17 is labeled by a tick t_K for some $K \subseteq \llbracket 0; N \rrbracket$, the hypothesis $K \cap I' = \emptyset$
 18 in the property ensures that x is not part of the set of clocks reset by e . Hence,
 19 we obtain $\mu_{j_{i'}}(x) + d_{j_{i'}} = [\mu_{j_i} + d_{j_i}]_{\{x_k \mid k \in I\}} + \sum_{k=j_i+1}^{j_{i'}} d_k = \tau_{i'} - \tau_i = 1$, which
 20 concludes the proof of the second point.
- 21 – *Properties 3 and 4.* Both properties 3 and 4 are similar and require the first
 22 tick of a clock in \mathbb{X}_t to occur one time unit after reaching the corresponding
 23 copy of the automaton. Indeed Property 3 is the particular case of $h = 0$ in
 24 the following proof, and Property 4 is the case $h > 0$. Let $h \in \llbracket 0; N \rrbracket$. We
 25 focus on the clock $x_h \in \mathbb{X}_t$. Let i be the index of the h -th observation in the
 26 trace, that is to say the h -th letter of the trace that is in $\Sigma \cup \Sigma^0$. If $h = 0$,
 27 we set $i = -1$. In any cases τ_i is the time of arrival in the copy h via the
 28 transition $e_{j_i} \in E^{h-1 \rightarrow h}$, and at this precise moment $x_h = 0$. Every guard
 29 of the transitions of $E^{h'}$ and $E^{h' \rightarrow h'+1}$ with $h' \geq h$ requires that $x_h < 1$,
 30 except for the transitions labeled by t_K with $h \in K$, which reset x_h and
 31 require $x_h = 1$. Thus, x_h must have been reset if the run lasts more than one
 32 time unit after the h -th observation, i.e., if $\tau_m - \tau_i \geq 1$. The only transitions
 33 from the copy h that can reset x_h are those labeled by t_K with $h \in K$; their
 34 guards require $x_h = 1$ so the first reset of x_h after reaching the copy h needs
 35 to occur at time $\tau_i + 1$.
- 36 – *Property 5.* We use the same argument as Properties 3 and 4 to prove the fifth
 37 property. Let $i \in \llbracket 0; m \rrbracket$, and suppose $a_i = t_I$ for some non-empty $I \subseteq \llbracket 0; N \rrbracket$
 38 and $\tau_m - \tau_i \geq 1$. The transitions following e_{j_i} require $\bigwedge_{k \in I} (x_k < 1)$, unless they
 39 reset all the clocks in $R_I = \{x_k \mid k \in I\}$. As $\tau_m - \tau_i \geq 1$, an observable action
 40 occurs at least one time unit after e_{j_i} and the clocks of R_I thus needed to

41 be reset. Hence there is a tick transition labeled by t_J with $I \subseteq J$ happening
 1 after e_{j_i} . □

2 **Second direction of the proof** Now we tackle the second direction of the
 3 proof: we show that if $a_0 \dots a_m$ is the trace of a run $[\rho]$, then all the timed words
 4 $(a_0, \tau_0) \dots (a_m, \tau_m)$ that satisfy the five properties of the lemma are in $T([\rho])$,
 5 i.e., there are runs in $[\rho]$ which produce these timed traces.

6 *Proof.* Let $w = a_0 \dots a_m$ be a trace in $\mathcal{RA}_{Tick}(Unfold_N(\mathcal{A}_{memo}))$, and let
 7 $\tau = (\tau_0, \dots, \tau_m)$ be a timestamps sequence such that the timed word
 8 $(a_0, \tau_0) \dots (a_m, \tau_m)$ verifies the five properties of the lemma. We define the se-
 9 quence $(f_i)_{0 \leq i \leq N}$ of the timestamps' fractional parts associated to each tick
 10 clock. For each $i \in \llbracket 0; N \rrbracket$ we set $J_i = \{j \in \llbracket 0; m \rrbracket \mid \exists I \subseteq \llbracket 0; N \rrbracket, i \in I \wedge a_j = t_I\}$.
 11 Let $first(i) = \min \{j \in \llbracket 0; m \rrbracket \mid \{k \in \llbracket 0; j \rrbracket \mid a_k \in \Sigma \cup \Sigma^0\} \mid = i\}$ (be the subscript
 12 of the i -th observation) if $i > 0$ and $first(0) = -1$. Thus $first(i)$ is the subscript of
 13 the first observation in w that resets the clock $x_i \in \mathbb{X}_t$. We set $f_i = \text{frac}(\tau_{first(i)})$.
 14 Then, from properties 2 and 4 we have that for each i in $\llbracket 0; N \rrbracket$ and for each
 15 $j \in J_i$, $\text{frac}(\tau_j) = f_i$. From property 3, we get $f_0 = 0$.

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

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

25 We distinguish two cases depending on whether r_n is a region where at least
 26 one tick clock has integer value (noting that, by property of the region automa-
 27 ton, if one valuation of $[\mu_n]$ give an integer value to a clock, then all valuations
 28 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*)

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

29 – $\forall x_i \in \mathbb{X}_t, \mu_n(x_i) \notin \mathbb{N}$:

30 Let i be the index of one of the last tick clocks that were reset in $[\rho]$ before r_n .
 31 If there is no tick clock reset before r_n , we set $i = 0$. Similarly, we consider

32 $j \in \llbracket 0; h+1 \rrbracket$ one of the clock indices of the next tick clock reset after r_n . If
 1 there is no tick clock reset after r_n , we take the next tick clock reset that is
 2 supposed to occur, i.e., $j = \min \{j' \in \llbracket 1; h \rrbracket \mid f_i < f_{j'}\}$ or $j = 0$ if this set is
 3 empty.
 4 If $j \in J_0$, we set (*type 2 constraint*)

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

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

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

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

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

5
 6 Now we combine this information to the constraints of the clock regions of a
 7 run in the region automaton to build a set of runs in $\text{Tick}(\text{Unfold}_N(\mathcal{A}_{\text{memo}}))$.
 8 For $n \in \llbracket 0; p \rrbracket$ we define the set of valuations

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

9 We denote by $[\rho]_\tau$ the subset of $[\rho]$ defined by $r'_0, b_0, r'_1, \dots, b_{p-1}, r'_p$ where for each
 10 $n \in \llbracket 0; p \rrbracket$, $r'_n = (\ell_n^h, M_n)$. The idea is that the successive clock valuations of runs
 11 of $[\rho]_\tau$ are in these sets M_n of valuations which correspond to region changes:
 12 a tick clock reaches or exits an integer value each time the next clock valuation
 13 μ'_{n+1} does no more satisfy C_{τ, r_n} .

14 We prove by induction that $[\rho]_\tau$ is not empty. The set M_0 contains at
 15 least μ_0 since it verifies $C_{\tau, r_0} = (\text{frac}(x_0) = 0) \wedge \bigwedge_{k=1}^N \text{frac}(x_k) = 0$. Assume
 16 now there is a path $(\ell_0^0, \mu'_0), b_0, \dots, (\ell_n^h, \mu'_n)$ with $\mu'_j \in M_j$ for each $j \in \llbracket 0; n \rrbracket$.
 17 We show there exists $\mu'_{n+1} \in M_{n+1}$ such that $(\ell_0^0, \mu'_0), b_0, \dots, (\ell_{n+1}^h, \mu'_{n+1})$ is

18 a path in $r'_0, b_0, r'_1 \dots r'_{n+1}$. It is well known [?] that in the region automaton,
 1 $((\ell, [\mu]), a, (\ell', [\mu'])) \in E_R$ if and only if for all $\mu \in [\mu]$ there exists $\mu' \in [\mu']$
 2 such that $(\ell, \mu) \xrightarrow{e} (\ell', \mu') \in E$, with e being the transition associated to
 3 $((\ell, [\mu]), a, (\ell', [\mu']))$ in the timed automaton. Since $\mu'_n \in [\mu_n]$, by the above prop-
 4 erty we can find $\mu'_{n+1} \in [\mu_{n+1}]$ following the transition (r_n, b_n, r_{n+1}) . We show
 5 that there is such a μ'_{n+1} that is also in M_{n+1} , i.e., such that $\mu'_{n+1} \models C_{\tau, r_{n+1}}$.

6 – Suppose first (r_n, b_n, r_{n+1}) is a discrete transition, with reset set R . Only
 7 one clock valuation μ'_{n+1} can succede to μ'_n in this case. We show it is in
 8 M_{n+1} . If $b_n \in \Sigma$ (resp. Σ^0), then μ'_n verifies a type 1 (resp. 4) constraint.
 9 We move to copy $h + 1$ with a reset of some tick clocks (including x_{h+1}) so
 10 the next constraint to be verified is of type 1. The clock valuation μ'_{n+1} is
 11 entirely determined by μ'_n and the clock reset set.

12 Since we move to copy $h + 1$, the clock x_{h+1} is now part of the tick clocks
 13 indexed from 0 to the current copy number, and $\mu'_{n+1}(x_{h+1}) = 0$, which is
 14 required by the constraint $C_{\tau, r_{n+1}}$. Moreover the constraints on the other
 15 clocks did not change. Thus the new type 1 constraint $C_{\tau, r_{n+1}}$ is verified
 16 by μ'_{n+1} . Now, if $b_n \in \Sigma_t$, only clocks reaching 1 are reset so their frac-
 17 tional part is not affected and the clock valuation μ'_{n+1} still satisfies C_{τ, r_n} .
 18 The constraint $C_{\tau, r_{n+1}}$ depends on the time elapsed since the last tick and
 19 whether the transition was an observation, so in this case $C_{\tau, r_{n+1}} = C_{\tau, r_n}$.
 20 Finally, if b_n is an ε -transition, it only resets clocks from \mathbb{X} and we have again
 21 $C_{\tau, r_{n+1}} = C_{\tau, r_n}$. Since this constraint restricts only clocks from \mathbb{X}_t and since
 22 their valuation does not change, μ'_{n+1} still verifies C_{τ, r_n} and is in M_{n+1} .

23 – We now assume that (r_n, b_n, r_{n+1}) is a delay transition: there is $d_n \in (0; 1)$
 24 such that $\mu'_{n+1} = \mu'_n + d_n$. This delay must verify some conditions because μ'_n
 25 is fixed and a transition can change region only once. In all this paragraph,
 26 we take i (resp. j) some index of the last (resp. next) reset tick clocks.
 27 Suppose that C_{τ, r_n} is a constraint of type 2, 3 or 4. Assume in the first case
 28 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
 29 type 1). Thus this clock has reached 1 ($\mu'_{n+1}(x_j) = 1$) and C_{τ, r_n} is a type 2
 30 or 3 constraint. This setting entirely determines the unique reachable clock
 31 valuation μ'_{n+1} . From the fact that μ'_n models C_{τ, r_n} , we have some $\delta \in (0; 1)$
 32 such that $1 + f_j - f_j - \mu'_n(x_j) = \delta$. Consequently $d_n = \delta$. We easily verify
 33 that the obtained μ_{n+1} satisfies $C_{\tau, r_{n+1}}$. Now if the next clock to be reset is
 34 x_{h+1} and there is no other delay transition before the next observation, then
 35 C_{τ, r_n} is a type 4 constraint and we need to have $\mu'_{n+1}(x_{h+1}) = f_{h+1}$, which
 36 means that $d_n = f_{h+1} - \mu'_n(x_0)$. The next constraint $C_{\tau, r_{n+1}}$ is still the same
 37 as C_{τ, r_n} . We have some δ for C_{τ, r_n} and obtain the new $\delta' = \delta - d_n = 0$ to
 38 satisfy $C_{\tau, r_{n+1}}$.

39 Suppose now that there will be at least one other delay transition before the
 40 next tick clock reset. Assume C_{τ, r_n} is of type 2, 3 or 4. Then, if this transition
 41 changes region, it only concerns clocks from \mathbb{X} and we have $C_{\tau, r_n} = C_{\tau, r_{n+1}}$.
 42 The preceding constraint gives $\delta = 1 - \mu'_n(x_j)$ if C_{τ, r_n} is of type 2 or 3 and
 43 $\delta = f_j - \mu'_n(x_j)$ if it is of type 4. The delay d_n cannot reach δ (otherwise
 44 $\mu'_{n+1}(x_j) \in \mathbb{N}$ or the transition crosses more than one clock region at once,

45 two excluded cases). We can construct the new $\delta' = \delta - d_n$ involved in
 1 $\mu'_{n+1} \models C_{\tau, r_{n+1}}$ and verify that μ'_{n+1} is thus still in M_{n+1} . Assume now that
 2 C_{τ, r_n} is of type 1. Then the next region must give a constraint of type 2, 3
 3 or 4. This case is similar to the last tackled one, and for the same reasons
 4 the delay d_n cannot reach $1 - f_i + f_*$ where $f_* \in \{-f_j, 1 - f_j\}$. In both cases
 5 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
 6 verified.

7 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 $\text{frac}(\tau'_i) = \text{frac}(\mu_{j_i}(x_0))$. 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 $\text{frac}(\tau'_i) = f_k = \text{frac}(\tau_i)$. Otherwise a_i
 is an observation, so there exists $h \in \llbracket 0; N - 1 \rrbracket$ such that $i = \text{first}(h + 1)$. 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 $\text{frac}(\mu_{j_i}(x_{h+1})) = f_{h+1}$ and $\text{frac}(\mu_{j_i}(x_0)) = f_{h+1}$. Since $i = \text{first}(h + 1)$, we
 obtain $\text{frac}(\tau_i) = f_{h+1} = \mu_{j_i}(x_0) = \text{frac}(\tau'_i)$. \square