Buffered Control for Opacity in Timed Automata

Étienne André
Nantes Université, CNRS, LS2N, Nantes, France
Institut universitaire de France (IUF), Paris, France

Sarah Dépernet
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Engel Lefaucheux
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract

Timed automata are an extension of finite automata that can measure and react to the passage of time, handling real-time constraints by using clocks. The timed opacity problem, where an attacker attempts to infer from observed actions and timestamps whether a secret location was visited, was shown undecidable for timed automata. Execution-time opacity is a decidable though limited setting in which the attacker attempts to detect whether the secret location was visited, by only relying on the run duration. Here, we significantly extend this setting, by allowing the attacker to observe all observable actions, in the right order though with only the integral parts of their timestamps, which we call buffered observations. We consider the controlled setting, in which we aim at dynamically defining a sequence of sets of enabled actions ensuring opacity with buffered observations. We first prove the inter-reducibility of full opacity (observations must not leak the visit of the secret location) and weak opacity (the attacker might prove that the location was not visited, but not that it was visited) in this new controlled setting. Then, we prove the undecidability of the problem of existence of a sequential control strategy ensuring opacity under buffered observations. Finally and most importantly, we prove that decidability is retrieved in two independent cases, with their theoretical complexities, with and without control. These two assumptions express realistic limitations of the controller. The first case is when the strategy of the controller changes at most an *a priori* fixed number of times per time unit, which is not a strong practical assumption. The second case is when all controllable actions are observable and distinguishable by an attacker.

2012 ACM Subject Classification Theory of computation → Timed and hybrid models; Security and privacy → Logic and verification; Theory of computation → Verification by model checking

Keywords and phrases timed automata, side-channel attack, observation with finite precision, control

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2026.11

Related Version *Extended Version including proofs*: `http://arxiv.org/abs/2606.28170`

Funding This work is partially supported by ANR BisoUS (ANR-22-CE48-0012) and ANR GUMMIS (ANR-25-CE48-5096).

Acknowledgements We would like to thank the CONCUR reviewers for useful suggestions of improvement.

1 Introduction

Information-system security remains a cornerstone of modern computing, yet its challenges become markedly more intricate when timing constraints are introduced. In many safety-critical and real-time applications – such as embedded controllers, network protocols, and cyber-physical systems – the correctness of security mechanisms depends not only on what actions occur but also on when they occur. Timed automata (TAs) [1] provide a natural and expressive formalism for modelling such time-dependent behaviours, allowing to capture both discrete transitions and continuous clock evolutions within a unified framework. Within

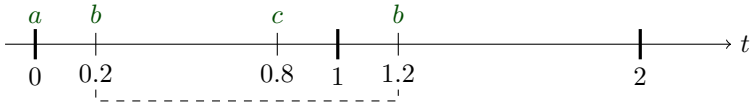


Figure 1 Example of observation with infinite precision.

this timed setting, opacity has emerged as a fundamental privacy property: a system is opaque with respect to a secret if an external observer, having full or partial visibility of the system's observable actions and timestamps, can never infer a given secret, typically the reachability of a given location. This information can be hidden by ensuring that each observation associated to a secret behaviour is also produced by a public behaviour. Opacity thus formalizes the notion of "information hiding" against attackers who exploit timing information.

It is well-known since Cassez' seminal work on timed opacity [15] that deciding opacity for TAs is undecidable in dense time when the attacker observes the full timed word produced by a run, even for restricted subclasses of TAs such as event-recording automata [2]. This undecidability result underscores the inherent difficulty of guaranteeing privacy in real-time systems and motivates the development of approximate or restricted analyses. This problem becomes however decidable in a discrete-time setting [5, 9, 18], where actions only occur at integral times. However, discrete-time analyses can be limited, and we address here the general dense-time case.

Contributions. In this paper (as in a number of former works), the secret consists in the visit (or not) of a given set of private locations. The attacker aims at deciding whether at least one such private location was visited, by observing the system traces; the attacker additionally knows the system model (here, a TA). In practice, an attacker may not have infinite precision in its observations. Therefore, it is interesting to study timed opacity in the setting of *buffered observations*: instead of the exact timed word as in [15], the attacker observes all actions within each one-unit time interval, in the right order but without their precise timestamp. The analogy is therefore a *buffer* that records all observable actions, and that is read every time unit. This is particularly realistic when the attacker does not have the ability (e.g., in terms of memory) to record all exact timestamps. For example, consider the sequence of actions in Figure 1. An attacker with infinite precision in its observations will see that the two occurrences of b are separated by exactly 1 time unit, which may give crucial information towards the knowledge of a secret. However, under buffered observations, the attacker will only be able to see that a occurs at $t = 0$, b and c occur (in this order) between 0 and 1, while b occurs within $(1, 2)$ – in which case the aforementioned observation that both b are separated by 1 time unit is lost. In fact, this behaviour becomes indistinguishable from another behaviour where the second b would occur, e.g., at $t = 1.1$ or at $t = 1.3$.

In the setting of buffered observations, opacity has been shown decidable [17] (under the name of "timed opacity against intruders with discrete-time precision"). We extend to this setting of buffered observations the notion of *control*: our aim is to decide the *existence of a strategy*, i.e., a dynamic selection of controllable actions such that the system is opaque. This is of utmost importance when we aim at making a real-time system insensitive to timing attacks with a finite precision.

In this sense, our main contribution is twofold: we first prove the undecidability of the control problem with buffered observations. Second, we provide two assumptions expressing realistic limitations of the controller, each of them allowing to regain the decidability separately.

The first assumption is that the strategy of the controller changes at most an *a priori* fixed number of times per time unit. The second assumption is that all controllable actions are observable and distinguishable by an attacker.

Moreover, in each result of this paper, we address both opacity levels: *weak* opacity (in which we only hide having visited a private location); and *full* opacity (in which we additionally hide *not* having visited it). We show all of our proofs work for both flavours, by proving in the setting of control the inter-reducibility between weak and full opacity.

Related work. The aforementioned negative result from [15] leaves hope only if the definition or the setting is changed. First, in [22, 23], the input model is simplified to *real-time automata*. [19, 20] exhibits decidability results for constant-time labelled automata. In [24], Zhang studies decidability for labelled real-time automata.

Second, in [5, 9, 18, 16, 4], decidability results are exhibited in the setting of Cassez' definition, with other restrictions in the model: one-clock automata, one-action automata, event-recording automata, over discrete time or equivalently with integer resets.

Third, in [3, 6], the authors consider a *time-bounded* notion of the opacity of [15], where the attacker has to disclose the secret before a deadline, using a partial observability. A somehow similar framework is considered in [21], in which the attacker has a bounded memory, and a finite duration between distinct observations is required, in which case the problem is decidable and PSPACE-complete.

Fourth, in [10], an alternative definition is proposed, by studying execution-time opacity (ET-opacity): the attacker has only access to the *execution time* of the system, as opposed to Cassez' partial observations where some events (with their timestamps) are observable. The goal for the attacker is to deduce whether a special secret location was visited, by observing only the execution time. In that case, most problems for TAs become decidable.

Control of opacity was addressed in the simpler setting of ET-opacity: in this setting, control was investigated with an untimed controller in [7] and a timed controller in [8]. Control was also addressed in [13] in the setting of *non-interference* in TAs.

Opacity in discrete-event systems was also considered, e.g., recently in [25].

General security problems for TAs are surveyed in [12].

Outline. We recall necessary preliminaries, and introduce our setting of buffered observations and formal problem in Section 2. We then prove that full and weak opacity can be inter-reduced (Section 3). We then prove our main contributions: controlling opacity under buffered observations is undecidable in general (Section 4), but decidability can be retrieved whenever the number of controller changes per time unit is statically bounded (Section 5) or whenever all controllable actions are distinguishable by an attacker (Section 6).

2 Preliminaries

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

Clocks are real-valued variables that all evolve over time at the same rate. Throughout this paper, we assume a set $\mathbb{X} = \{x_1, \ldots, x_H\}$ of *clocks*. A *clock valuation* is a function $\mu : \mathbb{X} \to \mathbb{R}_{\geq 0}$, assigning a non-negative value to each clock. We write $\vec{0}$ for the clock valuation assigning 0 to all clocks. Given a constant $d \in \mathbb{R}_{\geq 0}$, $\mu + d$ denotes the valuation

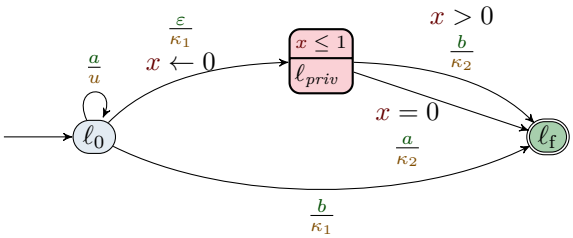


Figure 2 A TA example.

$(\mu + d)(x) = \mu(x) + d$, for all $x \in \mathbb{X}$. If R is a subset of $\mathbb{X}$ and μ a clock valuation, we call *reset* of the clocks of R and denote by $[\mu]_R$ the valuation for all clock $x \in \mathbb{X}$, $[\mu]_R(x) = 0$ if $x \in R$ and $[\mu]_R(x) = \mu(x)$ otherwise.

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

2.1 Timed automata

Timed automata (TAs) extend finite automata with a finite set of clocks with values in $\mathbb{R}_{\geq 0}$. We extend standard TAs with **i)** a set of private locations (to model the secret), and **ii)** *two* action sets: one for observable events and one for controllable events. Each action is thus represented as a pair $\frac{a}{\kappa}$ specifying both its observable component a and its controllable component κ.

▶ **Definition 1.** *A TA $\mathcal{A}$ is a tuple $\mathcal{A} = (\Sigma_o, \Sigma_c, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$, where:* **1)** *$\Sigma_o$ is a finite set of observable actions;* **2)** *Σ_c is a finite set of controllable actions;* **3)** *L is a finite set of locations,* **4)** *$\ell_0 \in L$ is the initial location,* **5)** *$L_{priv} \subseteq L$ is a set of private locations,* **6)** *$L_f \subseteq L$ is a set of final locations,* **7)** *$\mathbb{X}$ is a finite set of clocks,* **8)** *I is the invariant, assigning to every $\ell \in L$ a constraint $I(\ell)$, and* **9)** *E is a finite set of edges $e = (\ell, g, \frac{a}{\kappa}, R, \ell')$ where $\ell, \ell' \in L$ are the source and target locations, $a \in \Sigma_o \cup \{\varepsilon\}$ (where ε denotes an unobservable action), $\kappa \in \Sigma_c \cup \{u\}$ (where u denotes an uncontrollable action), $R \subseteq \mathbb{X}$ is a set of clocks to be reset, and g is a constraint (called* guard*).*

▶ **Example 2.** In Figure 2, we give an example of a TA with three locations ℓ_0, ℓ_{priv} and ℓ_f, five edges, two observable actions $\{a, b\}$, two controllable actions $\{\kappa_1, \kappa_2\}$, and one clock x. The initial location is ℓ_0, ℓ_{priv} is the (unique) private location, and ℓ_f is the (unique) final location. The location ℓ_{priv} has an invariant "$x \leq 1$". The edge from ℓ_0 to ℓ_{priv} is labelled by both the unobservable action ε and the controllable action κ_1, and resets the clock x.

▶ **Definition 3** (Semantics of a TA). *Given a TA $\mathcal{A} = (\Sigma_o, \Sigma_c, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$, the semantics of $\mathcal{A}$ is given by the timed transition system $\mathfrak{T}_{\mathcal{A}} = (\mathfrak{S}, \mathfrak{s}_0, \Sigma^{\times} \cup \mathbb{R}_{\geq 0}, \rightarrow)$, with*

1. $\mathfrak{S} = \{(\ell, \mu) \in L \times \mathbb{R}_{\geq 0}^{\mathbb{X}} \mid \mu \models I(\ell)\}$, $\mathfrak{s}_0 = (\ell_0, \vec{0})$,
2. $\Sigma^{\times} = \{\frac{a}{\kappa} \mid (a, \kappa) \in (\Sigma_o \cup \{\varepsilon\}) \times (\Sigma_c \cup \{u\})\}$,
3. $\rightarrow \subseteq (\mathfrak{S} \times E \times \mathfrak{S}) \cup (\mathfrak{S} \times \mathbb{R}_{\geq 0} \times \mathfrak{S})$ *consists of the discrete and (continuous) delay transition relations:*
   - **a.** *discrete transitions:* $((\ell, \mu), e, (\ell', \mu')) \in \rightarrow$, *if* $(\ell, \mu), (\ell', \mu') \in \mathfrak{S}$, $e = (\ell, g, \frac{a}{\kappa}, R, \ell') \in E$, $\mu' = [\mu]_R$, *and* $\mu \models g$.
   - **b.** *delay transitions:* $((\ell, \mu), d, (\ell, \mu{+}d)) \in \rightarrow$, *if* $d \in \mathbb{R}_{\geq 0}$ *and* $\forall d' \in [0, d], (\ell, \mu{+}d') \in \mathfrak{S}$.

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

Given a TA $\mathcal{A}$ with semantics $(\mathfrak{S}, \mathfrak{s}_0, \Sigma^\times \cup \mathbb{R}_{\geq 0}, \to)$, we refer to the elements of $\mathfrak{S}$ as the *configurations* of $\mathcal{A}$. A (finite) *run* of $\mathcal{A}$ is an alternating sequence of configurations of $\mathcal{A}$ and pairs of delays and edges starting from the initial configuration $\mathfrak{s}_0$ and ending in a final configuration (i.e., whose location is final), of the form $(\ell_0, \mu_0), (d_0, e_0), (\ell_1, \mu_1), \dots (\ell_n, \mu_n)$ for some $n \in \mathbb{N}$ (called the length of the run), 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 $(\ell_i, \mu_i) \xrightarrow{(d_i, e_i)} (\ell_{i+1}, \mu_{i+1})$. We denote by $\Omega(\mathcal{A})$ the set of runs of $\mathcal{A}$. A *path* of $\mathcal{A}$ is an infix of a run beginning and ending with a configuration.

A common abstraction of the semantics of TAs is the *region automaton* [1], recalled in Appendix A.

2.2 Buffered observations

A timed word is a sequence of pairs made up of a label (observation, control or both) and a timestamp in $\mathbb{R}_{\geq 0}$, with the timestamps being non-decreasing over the sequence. Timed words over an alphabet Σ are therefore elements of $(\Sigma \times \mathbb{R}_{\geq 0})^*$.

We denote by $TW(\Sigma)$ the set of all finite timed words over the alphabet Σ. A timed word over a set $\Sigma^\times$ of actions in Σ_o with control in Σ_c induces two timed words, one over the set of actions Σ_o and one over the set of controls Σ_c, each in which the silent symbols ε and u have been removed. For such a timed word w in $TW(\Sigma^\times)$, we denote by $act(w)$ and $ctr(w)$ the action and control parts of w. We also denote the label and timestamp parts of the timed word (or timed sequence) w by $unt(w)$ and $time(w)$ respectively.

▶ **Example 4.** If $w = (\frac{a}{u}, 0.2)(\frac{\varepsilon}{\kappa_1}, 0.3)(\frac{b}{\kappa_2}, 0.5)$, then $act(w) = (a, 0.2)(b, 0.5)$ and $ctr(w) = (\kappa_1, 0.3)(\kappa_2, 0.5)$. We have $unt(w) = \frac{a}{u}\frac{\varepsilon}{\kappa_1}\frac{b}{\kappa_2}$, and $time(w) = 0.2 \cdot 0.3 \cdot 0.5$. The label parts of the action and control parts are $unt(act(w)) = ab$ and $unt(ctr(w)) = \kappa_1\kappa_2$.

A run (or a path) ρ of a TA $\mathcal{A}$ defines a timed word $tw(\rho)$ over $\Sigma^\times$: if ρ is of the form $(\ell_0, \mu_0), (d_0, e_0), (\ell_1, \mu_1), \dots, (\ell_n, \mu_n)$ where for each $i \in [\![0; n-1]\!]$, $e_i = (\ell_i, g_i, \frac{a_i}{\kappa_i}, R_i, \ell_{i+1})$ and $\frac{a_i}{\kappa_i} \in \Sigma^\times$, then it generates the timed word $tw(\rho) = (\frac{a_{j_0}}{\kappa_{j_0}}, \sum_{i=0}^{j_0} d_i)(\frac{a_{j_1}}{\kappa_{j_1}}, \sum_{i=0}^{j_1} d_i) \cdots (\frac{a_{j_m}}{\kappa_{j_m}}, \sum_{i=0}^{j_m} d_i)$, where $j_0 < j_1 < \cdots < j_m$ and $\{j_k \mid k \in [\![0; m]\!]\} = \left\{i \in [\![0; n-1]\!] \mid \frac{a_i}{\kappa_i} \neq \frac{\varepsilon}{u}\right\}$.

The set of timed words recognized by a TA $\mathcal{A}$ is the set of timed words generated by its runs, i.e., $tw(\Omega(\mathcal{A}))$, which we also denote $\mathcal{L}(\mathcal{A})$ and call *language* of $\mathcal{A}$.

A *time interval* is an interval either of the form $\{n\}$ or $(n; n+1)$ with $n \in \mathbb{N}$. The set of time intervals is denoted by TI, and for $d \in \mathbb{R}_{\geq 0}$ then we denote by $Ti(d)$ the time interval containing d. If $\mathcal{I}$ is a convex union of time intervals and w a timed word, we define the *restriction of w to $\mathcal{I}$* as the maximal subword of w whose timestamps are all included in $\mathcal{I}$. We denote it by $\langle w \rangle_{\mathcal{I}}$. Formally, $\langle(\frac{a_0}{\kappa_0}, \tau_0)(\frac{a_1}{\kappa_1}, \tau_1) \dots (\frac{a_N}{\kappa_N}, \tau_N)\rangle_{\mathcal{I}} = (\frac{a_k}{\kappa_k}, \tau_k)(\frac{a_{k+1}}{\kappa_{k+1}}, \tau_{k+1}) \dots (\frac{a_{k+n}}{\kappa_{k+n}}, \tau_{k+n})$ with for all i in $[\![0; n]\!]$, $\tau_{k+i} \in \mathcal{I}$; if $k > 0$ then $\tau_{k-1} \notin \mathcal{I}$ and if $k + n < N$ then $\tau_{k+n+1} \notin \mathcal{I}$. This restriction of a timed word to a convex union of time intervals is also extended to runs: if $\rho = (\ell_0, \mu_0), (d_0, e_0), (\ell_1, \mu_1), \dots (\ell_N, \mu_N)$ for some $N \in \mathbb{N}$, the *restriction of ρ to a convex union of regions $\mathcal{I}$* is the path $\langle \rho \rangle_{\mathcal{I}} = (\ell_k, \mu_k), (d_k, e_k), (\ell_{k+1}, \mu_{k+1}), \dots (\ell_{k+n}, \mu_{k+n})$ such that:

- if $k > 0$, $\sum_{j=0}^{k-1} d_j \notin \mathcal{I}$; and
- for all $i \in [\![0; n]\!]$, $\sum_{j=0}^{k+i} d_j \in \mathcal{I}$; and
- if $k + n < N$, $\sum_{j=0}^{k+n+1} d_j \notin \mathcal{I}$.

We now define *buffered observations*, which correspond to the exhaustive sequences (in the right order) of observable actions made during each time interval and revealed, without their timestamp, when the next time interval is reached. New actions ▶ and ▷ are introduced to identify the time interval corresponding to each part of the buffered observation: a ▷ is added in the buffered observation whenever the total execution time enters an open interval, and a ▶ is added when the next integer is reached. We also consider the end of an execution as an action and annotate it with the symbol ▽. If $w = (\frac{a_0}{\kappa_0}, \tau_0)(\frac{a_1}{\kappa_1}, \tau_1)\dots(\frac{a_n}{\kappa_n}, \tau_n)$ is a timed word over the set of controlled actions $\Sigma^{\times}$, the buffered observation of w, denoted by $\mathit{buff}(w)$, is the untimed word

$$\mathit{buff}(w) = \mathit{unt}(\langle \mathit{act}(w)\rangle_{\{0\}}) \triangleright \mathit{unt}(\langle \mathit{act}(w)\rangle_{(0;1)}) \blacktriangleright \mathit{unt}(\langle \mathit{act}(w)\rangle_{\{1\}}) \triangleright \dots \mathit{unt}(\langle \mathit{act}(w)\rangle_{Ti(\tau_n)}) \triangledown z$$

for some $z \in \{\blacktriangleright, \triangleright\}$.

Each ▶ (resp. ▷) punctuates the end of a time interval of the form $(n; n+1)$ (resp. $\{n\}$), distinguishing actions made during a time interval, or made at a precise integral time.

▶ **Example 5.** For a timed word $w = (\frac{a}{u}, 0.2)(\frac{\varepsilon}{\kappa_1}, 0.3)(\frac{b}{\kappa_2}, 0.5)$, the buffered observation is the untimed word $\triangleright\, ab \triangledown \blacktriangleright$. It retains that in w, a and b occur during the time interval $(0; 1)$. The buffered observation does not distinguish whether some time elapsed between those two actions. The symbol ▽ preceding ▶ means that the run ended before the time 1.

Throughout a path ρ, the controller or an attacker only has access to the buffered observation of this path, $\mathit{Tr}(\rho) = \mathit{buff}(\mathit{tw}(\rho))$, called the *trace* of ρ. We also use this notation for the traces of a set of runs. A TA can be easily modified, so that all runs automatically include the symbols ▶, ▷ and ▽, by adding a new clock reset every time unit, used to produce the ▶ and ▷ (see Appendix C). That is, from now on, for every path ρ, $\mathit{Tr}(\rho) = \mathit{unt}(\mathit{tw}(\rho))$.

2.3 Control

In [8], a form of control was introduced in order to enforce execution-time opacity ("ET-opacity"). There, a "meta-strategy" was restricting the set of transitions a run can take during each time interval, and their order. In particular, this strategy was relying exclusively on the total amount of time elapsed since the beginning of the run, and fixing at each time unit its approximate behaviour until the next time unit.

In order to extend this control to the much broader setting of buffered observations, our controller may have access to some part of the trace during the run. We define sequential strategies, extending the notion of meta-strategy from [8] to our setting. To do so, we first define the *partial trace* $\mathit{Tr}_{\mathcal{I}}(\rho)$ of a run ρ in the time interval $\mathcal{I}$ as the restriction of $\mathit{Tr}(\rho)$ to the convex union of all time intervals preceding $\mathcal{I}$, i.e., to the interval $[0; n)$ if $\mathcal{I} = \{n\}$ and to the interval $[0; n]$ if $\mathcal{I} = (n; n+1)$. This way, $\mathit{Tr}_{\mathcal{I}}(\rho)$ gathers all observations taken into account by the controller at any time in $\mathcal{I}$, when it announces its strategy for the time interval $\mathcal{I}$. The set of all partial traces of the set $\Omega(\mathcal{A})$ of runs is denoted $\mathit{PartialTr}(\Omega(\mathcal{A}))$.

A *sequential strategy* σ is a function mapping a partial trace to a finite sequence of subsets of the control alphabet Σ_c, such that if the time interval is a singleton, then the sequence returned by the strategy includes only one element. The intuition is as follows: at each time interval, the strategy announces a sequence of control alphabets that will be used during that region. If the current time interval is a singleton $\{n\}$ (i.e., an integer time), then only one set of enabled actions is needed, as the region has no duration. If the current time interval is an open interval $(n; n+1)$, then the strategy may change the set of enabled actions multiple times during that interval, returning a sequence $\Sigma_{n,1}\Sigma_{n,2}\dots\Sigma_{n,K}$: the first set $\Sigma_{n,1}$ is active from the beginning of the interval until the first change, $\Sigma_{n,2}$ from the first change until the second, and so on. Formally:

$$\begin{array}{rccl}\sigma: & PartialTr(\Omega(\mathcal{A})) & \longrightarrow & \mathcal{P}(\Sigma_c)^* \\ & Tr_{\{n\}}(\rho) & \longmapsto & \Sigma_{n,1} \text{ if } Ti(dur(\rho)) = \{n\} \\ & Tr_{(n;n+1)}(\rho) & \longmapsto & \Sigma_{n,1}\Sigma_{n,2}\dots\Sigma_{n,K}, \text{ if } Ti(dur(\rho)) = (n;n+1), \text{ for } 0 < K \in \mathbb{N}.\end{array}$$

▶ **Example 6.** In the TA of Figure 2, a sequential strategy could allow the action κ_1 followed by the action κ_2 during the time interval $(0;1)$, and then forbid all controllable actions at the singleton $\{1\}$, giving the sequence $\{\kappa_1\}\{\kappa_2\}$ for $(0;1)$ and the empty set $\emptyset$ for $\{1\}$.

An *implementation of a sequential strategy* σ extends σ with exact timestamps. That is, for each time interval $\mathcal{I} \in \{\{n\}, (n;n+1)\}$ and partial trace, given the sequence $\Sigma_{n,1}\Sigma_{n,2}\dots\Sigma_{n,K}$ defined by σ, an implementation ν produces a timed sequence $(\Sigma_{n,1},t_{n,1})(\Sigma_{n,2},t_{n,2})\dots(\Sigma_{n,K},t_{n,K})$ with $(t_{n,k})_{k\in[\![1;K]\!]}$ an increasing sequence with values in $\mathcal{I}$ and with $t_{n,K} = n$ if $\mathcal{I} = \{n\}$ and $t_{n,K} = n+1$ if $\mathcal{I} = (n;n+1)$. Formally, ν *implements* σ if and only if for every buffered observation w of a run ρ, the timed sequence $\nu(w)$ is such that $unt(\nu(w)) = \sigma(w)$ and $\langle\nu(w)\rangle_{Ti(dur(\rho))} = \nu(w)$.

Let ν be some implementation of a strategy σ. Let $t \in \mathbb{R}_{\geq 0}$. The set of control labels *enabled by* ν *at time* t *knowing the partial trace* w is

$$E_\nu(t,w) = \sigma(w)_i \cup \{u\} \quad \text{with } i = \min\{k \mid t \leq time(\nu(w))_k\}.$$

▶ **Example 7.** Let ν be some implementation of the sequential strategy from Example 6 such that $\nu(\triangleright) = (\{\kappa_1\}, 0.4)(\{\kappa_2\}, 1)$ and $\nu(\triangleright\ a\ \blacktriangleright) = (\emptyset, 1)$. At time 0.2, ν enables the edges with control label κ_1 or u in the TA from Figure 2; at time 1, only u is enabled.

The runs of $\mathcal{A}$ which transitions are all enabled by an implementation ν of the strategy σ are called σ-*compatible* runs. Their set is denoted $\Omega^\sigma(\mathcal{A})$. Formally:

$$\begin{aligned}\Omega^\sigma(\mathcal{A}) = \Big\{\rho \in \Omega(\mathcal{A}) \mid \rho = (\ell_0,\mu_0)(d_0,\frac{a_0}{\kappa_0})\dots(d_{n-1},\frac{a_{n-1}}{\kappa_{n-1}})(\ell_n,\mu_n) : \exists \nu \text{ implementing} \\ \sigma \text{ such that } \forall 0 \leq i < n, \kappa_i \text{ is enabled by } \nu \text{ at time } \sum_{k=0}^{i} d_k \text{ knowing} \\ Tr_{Ti(\sum_{k=0}^{i} d_k)}((\ell_0,\mu_0)(d_0,\frac{a_0}{\kappa_0})\dots(d_{i-1},\frac{a_{i-1}}{\kappa_{i-1}})(\ell_i,\mu_i))\Big\}\end{aligned}$$

The trace of a σ-compatible run is called *controlled trace*. The set of controlled traces $Tr(\Omega^{\mathcal{A}}(\sigma))$ of a TA $\mathcal{A}$ with a strategy σ is called the *controlled language* of $\mathcal{A}$ under σ.

We will consider two orthogonal restrictions of control with sequential strategies:

- A strategy for which every sequence of alphabets has length at most N, for a given $N \in \mathbb{N}$, is called an N-*sequential strategy* (N-SS for short). This restriction is natural, representing the inability of a control to make an arbitrarily high number of actions within a single time unit.
- All controllable actions are supposed observable and distinguishable from each other and from uncontrollable actions. This assumption, called *Observable Sequential Strategies* (OSS for short), directly concerns the $\mathcal{A}$ and the control alphabet: each control label κ is associated to an observable action a_κ such that a transition has observation a_κ if and only if it has the control label κ.

2.4 Opacity of a controlled TA

Given a TA $\mathcal{A}$ and a run $\rho = (\ell_0, \mu_0), (d_0, e_0), (\ell_1, \mu_1), \ldots, (\ell_n, \mu_n)$ of $\mathcal{A}$, we say that ρ *visits* the location $\ell \in L$ if there exists $m \in [\![0; n]\!]$ such that $\ell_m = \ell$. If ρ visits some $\ell_{priv} \in L_{priv}$, we say it is a *private run*; otherwise, it is a *public run*. We denote by $\Omega^{\sigma}_{priv}(\mathcal{A})$ (resp. $\Omega^{\sigma}_{pub}(\mathcal{A})$) the set of private runs (resp. public runs) under a strategy σ. We use $PrivTr^{\sigma}(\mathcal{A}) = Tr(\Omega^{\sigma}_{priv}(\mathcal{A}))$ to denote the set of traces of private runs of $\mathcal{A}$ under σ, and $PubTr^{\sigma}(\mathcal{A}) = Tr(\Omega^{\sigma}_{pub}(\mathcal{A}))$ for the set of traces of public runs of $\mathcal{A}$ under σ.

A TA can be modified to store within locations whether the current run visited a private location or not (one only needs to include a Boolean in the locations that is turned to *true* if a private location was visited on the way to this location, see [9], illustrated in Appendix B). As such, we can assume w.l.o.g. that the set of private locations L_{priv} is absorbing (*i.e.* no run can go from a location of L_{priv} to a location of $L \setminus L_{priv}$). We denote $L_{pub} = L \setminus L_{priv}$ the set of public states (that by construction can only be reached by public runs).

Opacity is a notion to describe that, despite all information gathered by an attacker observing a run, they cannot deduce a secret (here, whether a private location was visited). Thus we say that a trace is *opaque under* σ if it can be obtained from both a public and a private run, i.e., whenever it is in $PrivTr^{\sigma}(\mathcal{A}) \cap PubTr^{\sigma}(\mathcal{A})$. A TA $\mathcal{A}$ is *weakly opaque under strategy* σ if $PrivTr^{\sigma}(\mathcal{A}) \subseteq PubTr^{\sigma}(\mathcal{A})$ (all private traces are opaque under σ) and *fully opaque under* σ if $PrivTr^{\sigma}(\mathcal{A}) = PubTr^{\sigma}(\mathcal{A})$ (all traces are opaque under σ).

When considering a controlled TA, a first option consists in exhibiting a strategy simply ensuring opacity. A second option consists in requiring additionally that a final location remains reachable even with this restriction of the system's behaviour. We say that the strategy σ is *non-blocking* on $\mathcal{A}$ if $\Omega^{\sigma}(\mathcal{A})$ is not empty. This reachability condition is natural since in several cases, the best strategy achieving opacity is the one enabling as few actions as possible, or even entirely blocking the system. Indeed, a TA with no possible run is opaque, and such a control strategy which is not non-blocking is of little interest in practice.

We study the existence of a strategy ensuring opacity, both with and without requiring it to be non-blocking. We add a subscript "NB" to denote the version of problems with the non-blocking requirement.

Weak / Full $\exists$SO / $\exists$SO$_{\text{NB}}$: Existence of a (non-blocking) sequential strategy making the TA fully opaque (resp. weakly opaque) problem:
INPUT: A TA $\mathcal{A}$
PROBLEM: Does there exist a (non-blocking) sequential strategy making $\mathcal{A}$ weakly (resp. fully) opaque?

When limiting the search to N-sequential strategies or when in the observable sequential strategies settings, these problems are denoted: $\exists N$-SSO, $\exists N$-SSO$_{\text{NB}}$, $\exists$OSSO and $\exists$OSSO$_{\text{NB}}$.

▶ Remark 8. Opacity is worth investigating following two different approaches:

offline when the only traces we compare in order to decide opacity are those of runs reaching a final location;

online when *prefixes* are also considered, i.e., traces of all paths eventually leading (or not) to a final location.

Here, we focus on *offline opacity* and simply refer to it as *opacity*. However, note that online opacity can be seen as a special case of offline opacity, since adding a silent transition (taken without any delay) from all locations to a final location amounts to add all paths to the set of runs reaching a final location. As a consequence, solving online opacity problems can be done with the same algorithms as for offline opacity.

▶ **Example 9.** We denote by $\mathcal{A}$ the TA in Figure 2. To make it fully opaque, we must prevent some particular traces from appearing. A b occurring exactly at the same time as the last a reveals that the run is public. It is distinguishable by the attacker only if these actions occurs at a precise integral time, due to the imprecision of buffered observations. Since the attacker knows the chosen sequential strategy, enabling only κ_2 or only κ_1 during some time interval reveals from which side (public or private) the b has been made. Allowing both at the same time ($\{\kappa_1, \kappa_2\}$) would allow non-opaque traces with two a and no b. One could enable them one after the other, for instance with $\{\kappa_1\}\{\kappa_2\}$.

Blocking all control labels would ensure the full opacity of $\mathcal{A}$, since it prevents all runs from exiting the initial location, hence producing no complete trace.

Let σ be the sequential strategy such that for every integer n, if the buffered observation w ends in the time interval $\{n\}$ (the last symbol is ▶ or the trace is empty), then $\sigma(w) = \emptyset$, and if it ends in $(n; n+1)$ (last symbol: $\triangleright$) then $\sigma(w) = \{\kappa_1\}\{\kappa_2\}$. This 2-sequential strategy ensures the full opacity of $\mathcal{A}$, and it is non-blocking since the final location remains reachable. However, there is no non-blocking 1-sequential strategy ensuring the opacity of $\mathcal{A}$.

Moreover, if we remove the guard $x = 0$ in $\mathcal{A}$, then there is no non-blocking sequential strategy that ensures full opacity.

3 Inter-reducibility of weak and full opacity, with and without control

We first prove that full and weak opacity are inter-reducible in our setting, and thus that they have the same theoretical complexity. In the subsequent sections, we will therefore only establish the results for one of the two notions.

▶ **Theorem 10.** *The weak and full variants of the $\exists SO$, $\exists SO_{NB}$, $\exists N\text{-}SSO$, $\exists N\text{-}SSO_{NB}$, $\exists OSSO$ and $\exists OSSO_{NB}$ problems are inter-reducible.*

Proof sketch, see Appendix D for the full proof. In [9], weak and full ET-opacity were shown to be inter-reducible in the absence of control. The proof mainly relied on the ability to construct from a TA $\mathcal{A}$ another TA $\mathcal{A}'$ where private and public runs were swapped. This proof cannot carry over to the controlled setting since there is no guarantee that the same strategy can be applied in both TAs to make them *simultaneously* weakly opaque.

Reducing weak opacity problems to full opacity problems is relatively straightforward, by reusing TA fragments from the proofs in [9]. The opposite reduction is more technical: we use a single TA, and add a final letter ("a" or "b") pointing which inclusion is being tested: if the last letter of the trace is an "a", then the run participates in the test of the inclusion $PrivTr^{\sigma}(\mathcal{A}) \subseteq PubTr^{\sigma}(\mathcal{A})$; and if the last letter is a "b", then the run participates in the converse inclusion. ◀

4 Undecidability in the general case of buffered observations

While the buffering of the observations and the sequencing of the control weaken the required accuracy of the analysis, we show that the full $\exists$SO and $\exists\text{SO}_{\text{NB}}$ problems are undecidable. To do so, we proceed by reduction from the Post correspondence problem (PCP) to the problem of the existence of a sequential strategy ensuring full opacity, under buffered observations. Intuitively, we rely on the strategy having to plan an unbounded number of alphabets over the next time interval to have it choose the full word of the PCP, and then the following buffered observation reveals whether the choice was good (i.e., ensured opacity). Our reduction strongly relies on the absence of bounds on the number of control alphabets planned by the strategy in a single time interval (contrarily to the N-SS setting), and requires that the controllable actions do not appear in the buffered observation (contrarily to the OSS setting).

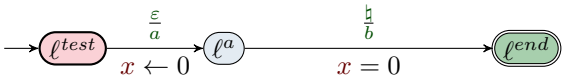


Figure 3 Gadget forbidding selecting two actions simultaneously, with a and b two actions in Σ_c with $b \neq a$ and $\natural$ an observation exclusive to this gadget.

▶ **Theorem 11.** *The weak and full $\exists SO$ and $\exists SO_{NB}$ problems are undecidable.*

Proof. Let $N \geq 2$ be an integer, and $V = (v^j)_{1 \leq j \leq N}$ and $W = (w^j)_{1 \leq j \leq N}$ be sequences of words over an alphabet Σ. The PCP asks whether there exists a sequence of indices $(i_k)_{1 \leq k \leq K}$, with $K \geq 1$ and for all k, $1 \leq i_k \leq N$, such that $v^{i_1} \cdot v^{i_2} \cdot \cdots \cdot v^{i_K} = w^{i_1} \cdot w^{i_2} \cdot \cdots \cdot w^{i_K}$. For $1 \leq i \leq N$, we denote by n_i and m_i the respective length of v^i and w^i.

We will build a TA $\mathcal{B}$ such that there exists a sequential strategy ensuring full opacity of $\mathcal{B}$ iff the instance (V, W) of the PCP has a solution starting with the pair (v_1, w_1) (this starting assumption is without loss of generality as there are finitely many starting options).

The TA $\mathcal{B}$ uses the control set $\Sigma_c = \{1, \ldots, N, e\}$, cannot take a transition labelled by a controllable action at time 0 and thanks to an invariant on the final locations every run reaches a final location before time 1. Hence, every sequential strategy on $\mathcal{B}$ is defined entirely by the sequence of control alphabets it selects for the interval $(0; 1)$. Moreover, $\mathcal{B}$ is made of three components reached at a time greater than 0 from the initial location. The first two limit the shape of the strategies that could ensure opacity, so that it can be associated to a word $i_1 e i_2 e \ldots i_k e$ where for each j, $1 \leq i_j \leq N$, and a third which tests whether the sequence $i_1 i_2 \ldots i_k$ is solution to the PCP instance. More precisely, $\mathcal{B}$ consists of:

1. a gadget ensuring that two actions cannot be selected at the same time (see Figure 3), hence a strategy achieving opacity is associated to a single sequence of alphabets, each of which contains a single action (the strategy can thus be represented by a word $a_1 \ldots a_k \in \Sigma_c^*$);
2. a gadget which forces that the strategy alternates between actions $i \in \{1, \ldots, N\}$ and the action e, starts with an action $i \in \{1, \ldots, N\}$ and ends with e, giving the form claimed (see Figure 4);
3. a part of the TA imitating the words of the PCP instance (see Figure 5): mainly, if the strategy is associated to a word $i_1 e i_2 e \ldots i_k e$, a private run will read $v^{i_1} \cdot v^{i_2} \cdot \cdots \cdot v^{i_K}$ and a public run will read $w^{i_1} \cdot w^{i_2} \cdot \cdots \cdot w^{i_K}$.

First component. Let us start by explaining the gadget of Figure 3. From an initial location ℓ^{test}, for every action a, if a is allowed, then a run can reach a location ℓ^a (there is thus one such location per controllable actions), while resetting a clock x. From ℓ^a, if another action b is allowed and if $x = 0$ (in other words, if b and a are allowed at the same time), the run can reach a final location while reading $\natural$. This thus produces a private run which trace is $\natural$, and as $\natural$ is exclusive to this gadget, this run violates opacity. Hence, any strategy that ensures opacity cannot allow two actions at the same time. As by the invariant mentioned earlier a run can only reach a final location before time 1, and as no action can be done in the second part of the construction (more details on this below) at time 0, a sequential strategy can be defined exclusively by the sequence of control alphabet it selects for the interval $(0; 1)$. Moreover, due to the above, if the strategy achieves opacity, only one action can be selected at a time. Thus there is a correspondence between sequential strategies that could achieve opacity, and sequences of actions.

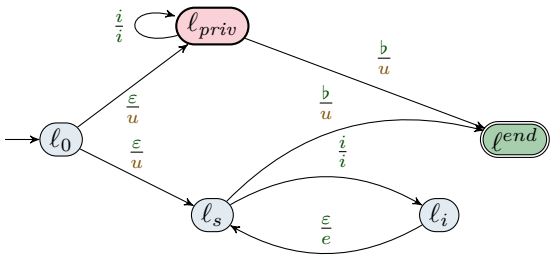


Figure 4 Gadget limiting the order of the actions selected by the strategy, with $1 \leq i \leq N$, and $\flat$ an observation exclusive to this gadget.

Second component. Let us explain the second gadget (Figure 4). Thanks to the first gadget, a sequential strategy can be represented by a single word $a_1 \dots a_k \in \Sigma_c^*$. In this gadget under this strategy, private runs will produce every word $w\sharp$ where w is a subword of $a_1 \dots a_k$ that does not contain e. On the public side however, if a run reads a letter associated to an integer, it goes to the location ℓ_i from which it can exit only if an e is allowed. Hence, to copy the private runs, public runs need every integer action to be followed by an e. As $\flat$ cannot be produced by any other component of $\mathcal{B}$, this ensures the alternation claimed in the strategy. While technically the strategy could start by allowing e, this will have no effect in any of the components of $\mathcal{B}$ and thus ignored without loss of generality. In the following, we thus assume the sequential strategy is associated to a word $i_1 e i_2 e \dots i_k e$.

Third component. We now explain the third component of the construction (Figure 5). This part of the TA consists in four branches reached at a time greater than 0.

First consider branch B. A run going through this branch will first read the word w^1 and reach a location with many exiting transitions, allowing to select which word to read. From there, either it takes the transition associated to i_1 and reads word w^{i_1} before coming back to the starting location by a transition labelled by an e, or it will stay in that location. This repeats for every action i_j, before the run decides to reach the final location (with an unobservable and uncontrollable action). As a consequence, this branch produces private runs with the buffered observation $w_1 w^{i_{j_1}} w^{i_{j_r}}$ where $j_1, \dots, j_r$ is an increasing subsequence of $1, \dots, k$. In other words, it starts by w_1 and then selects which of the words allowed by the strategy it reads. Action e ensures that, if an action i is allowed, no run can take twice the transition associated to it, as an e is necessary to come back to the location where the selection occurs.

Branch C is similar, producing public runs with the buffered observation $v^1 v^{i_{j_1}} v^{i_{j_r}}$ where $j_1, \dots, j_r$ is an increasing subsequence of $1, \dots, k$.

In particular, in order for full opacity to be ensured by the strategy, we see that, ignoring the other branches, the two longest words produced by runs in those branches ($w^1 w^{i_1} w^{i_k}$ and $v^1 v^{i_1} v^{i_k}$) must be equal as we want for the reduction. However, the other subwords need to be equal as well, which is not required in the PCP. This is the reason why the other two branches A and D exist. Branch A acts like C, except that it produces private runs, and the buffered observations it produces cannot be $v^1 v^{i_1} v^{i_k}$: in order to reach the final location, a run needs to take a transition with control i and the following transition of control e to go to the second line of the branch. In other words, it needs to "waste" one opportunity to read a word. It thus can produce every buffered observation except the full one. Branch D does the same for w / as branch B.

As a consequence, this component is fully opaque iff $w^1 w^{i_1} w^{i_k} = v^1 v^{i_1} v^{i_k}$, and thus iff the given PCP instance has a solution.

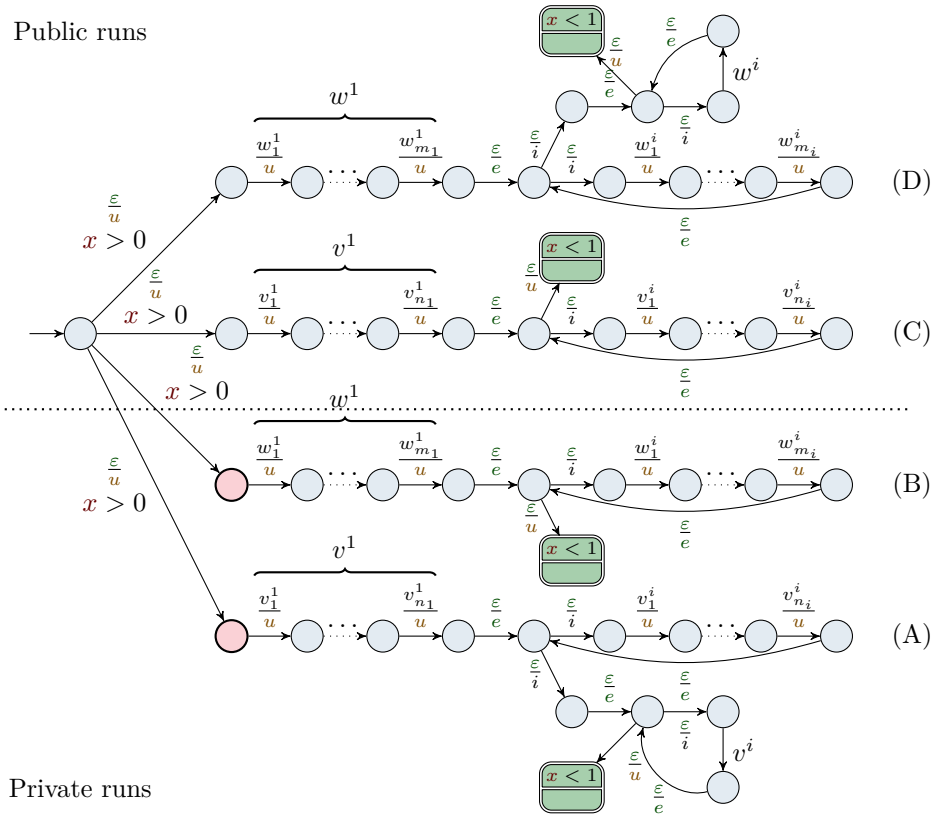


Figure 5 Reducing from PCP to ∃SO.

As a final location can be reached in the third component whatever the strategy does, this reduction applies to both the full ∃SO and full $\exists\text{SO}_{\text{NB}}$ problems. ◀

5 Ensuring opacity with N-Sequential Strategies (N-SS)

While the existence of a sequential strategy making the TA opaque under buffered observations is undecidable in general, we now prove that it becomes decidable when limiting the sequential strategies to *sequences of a predefined given length*.

▶ **Theorem 12.** *The decision problems $\exists N$-SSO and $\exists N$-SSO_{NB} are* 2EXPTIME*-complete.*

In the remaining of this section, we construct our *belief automaton*, an automaton that describes the set of possible regions the system can be in at each time under a given strategy. We establish formally that this belief automaton correctly represents the behaviour of the controlled TA. The $\exists N$-SSO and $\exists N$-SSO_{NB} problems can then be solved through a game on the belief automaton. We establish the 2EXPTIME-hardness of $\exists N$-SSO and $\exists N$-SSO_{NB} by reduction from the halting problem for alternating Turing machines with exponential memory. Detailed proofs of Theorem 12 are in [11].

We call *belief* a set of regions[1]. In particular, we call *natural belief* the set of regions in which the attacker *believes* to be according to their knowledge, i.e., the current buffered observation and the sequential strategy. For a TA $\mathcal{A}$ and a sequential strategy σ, we denote

[1] We follow the vocabulary from, e.g., [14]. This is also close to the concept of *estimator* (e.g., [18]).

by $\mathfrak{b}^\sigma_w$ the set of regions in which the system can be after a buffered observation w while following a strategy ν implementing σ in $\mathcal{A}$, i.e., $r \in \mathfrak{b}^\sigma_w$ iff there exists a run ρ in $\mathcal{A}$ such that ρ is σ-compatible, $last(\rho) \in r, r \in \mathsf{R}_\mathcal{A}$ and $tw(\rho) = w$. The set of all natural beliefs is denoted $\mathbb{B}^\sigma_\mathcal{A} = \{\mathfrak{b}^\sigma_w \mid w \in B\mathcal{L}(\mathcal{A}, \sigma)\}$ where $B\mathcal{L}(\mathcal{A}, \sigma)$ is the set of prefixes of controlled traces ending with a time interval symbol $\blacktriangleright$ or $\triangleright$.

Among those natural beliefs, we will be particularly interested in the ones showing *leaks* of information about the system. Intuitively, a *leaking* natural belief allows to discriminate private and public runs. For a given TA $\mathcal{A}$, we denote $\mathsf{Private}_\mathcal{A} = \left\{[(\ell, \mu)] \mid \ell \in L_{priv}, \mu \in \mathbb{R}^H_{\geq 0}\right\}$ the set of regions reachable after visiting ℓ_{priv} on a run in $\mathcal{A}$, and $\mathsf{Public}_\mathcal{A} = \left\{[(\ell, \mu)] \mid \ell \in L_{pub}, \mu \in \mathbb{R}^H_{\geq 0}\right\}$ the set of regions reachable on a run not visiting ℓ_{priv} in $\mathcal{A}$.

▶ **Definition 13.** *Given a TA $\mathcal{A}$, a natural belief B is said to be* leaking for full opacity *when exactly one of the following two conditions is satisfied:* **1)** $(B \cap \mathsf{R}^F_\mathcal{A} \cap \mathsf{Private}_\mathcal{A} \neq \emptyset)$, *or* **2)** $(B \cap \mathsf{R}^F_\mathcal{A} \cap \mathsf{Public}_\mathcal{A} \neq \emptyset)$.

This means that finishing in this belief leaks an information to the attacker: only one final state is possible (private or public, but not both).

By definition of leaking natural beliefs and full opacity, we immediately have the following result:

▶ **Lemma 14.** *Let $\mathcal{A}$ be a TA and σ a sequential strategy. $\mathcal{A}$ is fully opaque with σ iff there is no belief in $\mathbb{B}^\sigma_\mathcal{A}$ that is leaking for full opacity.*

We will now build an automaton, called *belief automaton*, constructing the natural beliefs from the observations and the choices of the sequential strategy. In the following, $\mathcal{P}(\Sigma_c)^{\leq N}$ denotes the set of sequences of length at most N of subsets of Σ_c. Let $S = \Sigma_1 \dots \Sigma_n$ be a finite sequence of alphabets. A path ρ is *locally S-compatible* if there exists a decomposition $\rho = \rho_1 \cdot (d_1, e_1) \cdot \rho_2 \cdot (d_2, e_2) \dots (d_{n-1}, e_{n-1}) \cdot \rho_n$ such that $ctr(tw(\rho_1)) \in (\Sigma_1 \cup \{u\})^*$; and for $i \in [\![1; n-1]\!]$, $ctr(tw((d_i, e_i) \cdot \rho_{i+1})) \in (\Sigma_{i+1} \cup \{u\})^*$ and $d_i > 0$. Let $S = \Sigma_1 \dots \Sigma_N$ and $S' = \Sigma_k \dots \Sigma_N$ be two sequences in $\mathcal{P}(\Sigma_c)^{\leq N}$, which we call *local strategies*, suffix one of the other for a certain index i. Then we denote by $S \sqcap S'$ the local strategy $\Sigma_1 \dots \Sigma_k$. Any path leading from a belief with local strategy S to another with local strategy S' must be locally $S \sqcap S'$-compatible.

Let r be a region, B be a set of regions, a an action and S, S' local strategies such that S' is a suffix of S. Then we define the following belief of r or of B, reading a and changing the strategy from S to S':

$$\begin{aligned} Next^{S'}_a(r, S) = \big\{ r' \mid \exists r \in B, \exists p \text{ a path in } \mathcal{A} \text{ with } first(p) \in r, last(p) \in r', \\ tw(p) = a \text{ and } p \text{ is locally } (S \sqcap S')\text{-compatible.} \big\} \\ Next^{S'}_a(B, S) = \bigcup_{r \in B} Next^{S'}_a(r, S) \end{aligned}$$

We extend the notation to a set H of pairs composed of a belief and a local strategy.

$$Succ_a(H) = \{(B', S') \mid B' = \bigcup_{(B,S) \in H \text{ s.t. } S' \text{ suffix of } S} Next^{S'}_a(B, S) \text{ with } a \in \{\blacktriangleright, \triangleright\} \implies S' = \emptyset\}.$$

▶ **Definition 15** (Belief Automaton). *For a given TA $\mathcal{A} = (\Sigma_o, \Sigma_c, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$, the belief automaton $\mathcal{B}_\mathcal{A}$ is given by the tuple $(Q_\mathcal{A}, \Gamma, B_0, \Delta)$ where:*

- *$Q_\mathcal{A} = \mathcal{P}(\left\{(B, S) \mid B \in \mathcal{P}(\mathsf{R}_\mathcal{A}) \wedge S \in \mathcal{P}(\Sigma_c)^{\leq N}\right\}) \times \{c, \overline{c}\}$, hence every state is associated to a set of pairs composed of one belief and a local strategy of length at most N, as well as a tag in $\{c, \overline{c}\}$;*

- $\Gamma = \mathcal{P}(\Sigma_c)^{\leq N} \bigcup \Sigma_o$; $B_0 = (\{(\{r_0\}, \emptyset)\}, c)$;
- $\Delta = \Delta_{strat} \cup \Delta_o \subset Q_{\mathcal{A}} \times \Gamma \times Q_{\mathcal{A}}$ *with*
  1. $\Delta_{strat} = \left\{((\{(B, \emptyset)\}, c), S, (\{(B, S)\}, \overline{c})) \mid B \in \mathcal{P}(\mathsf{R}_{\mathcal{A}}), S \in \mathcal{P}(\Sigma_c)^{\leq N}\right\}$,
  2. $\Delta_o = \{((H, \overline{c}), a, (Succ_a(H), z)) \mid H \in Q_{\mathcal{A}}, a$ *such that* $Succ_a(H) \neq \emptyset, z = c$ *if* $a \notin \{\blacktriangleright, \triangleright\}, z = \overline{c}$ *otherwise*$\}$

Intuitively, the belief automaton alternates between states that have a single belief and the tag c where a local strategy is selected, and states with the tag $\overline{c}$ associated to a set H with many pairs of beliefs and sequences where the observable actions are selected. A pair (B, S) contained in a state of the belief automaton H means that with the current observations, a run may be in the regions of B with the remaining local strategy S available for the runs. This represents the fact that throughout a time interval, we do not know the current implementation of the strategy, and thus when it moves to the next control alphabet. As the local strategy has at most N elements, a state H can keep up to N beliefs simultaneously depending on how far into the local strategy the runs gathered in this belief are.

▶ **Definition 16** (Controlled Belief Automaton)**.**
For a given TA $\mathcal{A} = (\Sigma_o, \Sigma_c, L, \ell_0, L_{priv}, L_f, \mathbb{X}, I, E)$ *and a sequential strategy* σ*, the controlled belief automaton* $\mathcal{B}^{\sigma}_{\mathcal{A}}$ *is given by the tuple* $(Q^{\sigma}_{\mathcal{A}}, \Gamma, B^{\sigma}_0, \Delta^{\sigma})$ *where:*

- $Q^{\sigma}_{\mathcal{A}} = Q_{\mathcal{A}} \times \Sigma_o^*$; $\Gamma = \mathcal{P}(\Sigma_c)^{\leq N} \bigcup \Sigma_o$; $B^{\sigma}_0 = (\{r_0\}, \emptyset, \varepsilon)$;
- $\Delta^{\sigma} = \Delta^{\sigma}_{strat} \cup \Delta^{\sigma}_o \subset Q^{\sigma}_{\mathcal{A}} \times \Gamma \times Q^{\sigma}_{\mathcal{A}}$ *with*
  1. $\Delta^{\sigma}_{strat} = \{((q, w), S, (q', w) \mid S = \sigma(w) \wedge (q, S, q') \in \Delta_{strat}\}$;
  2. $\Delta^{\sigma}_o = \{(q, w), a, (q', w \cdot a) \mid (q, a, q') \in \Delta_o\}$

We now aim to prove that the natural beliefs correspond to the beliefs of the form $(\{(B, \emptyset)\}, c, w)$, called *encountered beliefs*, by showing that these beliefs are reachable if and only if for all region r in B, there exists a run σ-compatible which trace is w, starting in the region r_0 and ending in r.

▶ **Lemma 17.** *The natural beliefs are exactly the encountered beliefs in the controlled belief automaton.*

The proof of this lemma is in [11]. Thanks to this result, the $\exists N$-SSO problem can be translated into a safety two-player game on the belief automaton where the first player selects the strategy and aims to avoid the leaking encountered belief, while the second player selects the buffered observation. The translation of the $\exists N$-SSO_{NB} problem is slightly more difficult, as we need to combine the safety condition and the reachability condition into a single Büchi condition for the game. In both cases, we translate the opacity problems into a doubly exponential game, that can be solved in polynomial time in the size of the game – hence the 2EXPTIME algorithm. The details of this game as well as the hardness proof are also in [11].

6 Ensuring opacity with Observable Sequential Strategies (OSS)

In a control setting, it is natural to assume that controllable events are observable, because the controller can only choose or enable events it can detect: if it could not observe them, it would not know when or whether its control decisions are being applied. This is the motivation behind the OSS setting. An unfortunate consequence of this assumption however is that allowing more controllable actions does not help make the system opaque, and thus there exists a strategy ensuring weak opacity iff the strategy $\sigma_{\emptyset}$ – that constantly blocks every controllable action – ensures weak opacity. The proofs of this section are in [11].

Table 1 Summary: impact of the attacker observation power on the decidability of the existence of a sequential strategy ensuring opacity.

Attacker observation power	Decidability
All observations	undecidable [15]
Execution time only	in EXPSPACE [8]
Buffered observations	undecidable (Theorem 11)

Table 2 Summary: decidability of the existence of a strategy ensuring opacity under buffered observations.

Type of strategy	General strategy	Non-blocking strategy
Sequential	undecidable (Th. 11)	
N-sequential	2EXPTIME-complete (Th. 12)	
Observable sequential	EXPSPACE-complete (Th. 19)	In 3EXPTIME (2EXPTIME-hard) (Th. 20)

▶ **Proposition 18.** *Given a TA $\mathcal{A}$ in the OSS setting, the existence of a sequential strategy making $\mathcal{A}$ weakly opaque implies that $\sigma_\emptyset$ makes $\mathcal{A}$ weakly opaque.*

The EXPSPACE-completeness of the opacity problem without control hence immediately gives:

▶ **Theorem 19.** *The ∃OSSO problem is* EXPSPACE*-complete.*

Requiring the sequential strategy to be non-blocking avoids – at least partially – the previous situation, as blocking every controllable action may not allow any run to reach a final location. Nonetheless, we can similarly limit the shape of the non-blocking sequential strategies that achieve opacity: such a strategy only needs to allow (at least) *one run* leading to a final location, and any deviation from that run only has to satisfy opacity (without necessarily satisfying the non-blocking assumption) and thus can be supervised by the strategy $\sigma_\emptyset$. Moreover, we can produce a doubly exponential bound for the length of the path leading to the final location. This bound can then be used to limit the sequences of control alphabet the sequential strategy needs to use, hence reducing the $\exists\text{OSSO}_{\text{NB}}$ problem to the $\exists N$-SSO problem.

▶ **Theorem 20.** *The $\exists OSSO_{NB}$ problem is in* 3EXPTIME *and* 2EXPTIME*-hard.*

7 Conclusion

We studied the problem of controlling timed automata to ensure opacity under *buffered observations*, a realistic attacker model in which the attacker observes all actions within each time unit, in the right order but without their precise timestamps. We first proved that full and weak opacity are inter-reducible, and that they are both undecidable in this controlled setting. We then identified two independent and practically motivated restrictions of the controller, both restoring decidability alone: when the strategy changes at most N times per time unit (N-sequential strategies), and when all controllable actions are observable and distinguishable by the attacker (observable sequential strategies), the problems become decidable, and in most cases 2EXPTIME-complete (with the exception of the ∃OSSO problem which is EXPSPACE-complete). We summarise our results in Tables 1 and 2.

Even though we identified realistic decidable classes of TAs, the theoretical complexity of solving opacity for these models is high and, due to our hardness results, it cannot be substantially improved. In order to be able to handle those problems in practice, one needs

either to find a relevant subclass of TAs with a lower complexity, or to design heuristics that – despite high theoretical complexities – remain practically efficient. The former could consist for instance in *bounding the observer's memory* across time units. The latter would require a completely different approach from ours, as our techniques strongly rely on the region automaton, which is known to be practically expensive. Moreover, the *zone automaton* (a usual structure to circumvent the region automaton) does not pair well with the determinisation step we need to construct the belief automaton. One could for instance adapt the notion of control over buffered observation in an *untimed* setting, hence removing entirely the need for the region automaton. In another direction, we will investigate *parametric variants*, where N or some elements of the TA are not part of the input, hence allowing to represent systems that are not fully defined.

References

1 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994. doi:10.1016/0304-3975(94)90010-8.

2 Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoretical Computer Science*, 211(1-2):253–273, 1999. doi:10.1016/S0304-3975(97)00173-4.

3 Ikhlass Ammar, Yamen El Touati, Moez Yeddes, and John Mullins. Bounded opacity for timed systems. *Journal of Information Security and Applications*, 61:1–13, September 2021. doi:10.1016/j.jisa.2021.102926.

4 Sjood Ammen Daje, Tareq Ahmad Al-Sarayrah, Ding Liu, and Zhiwu Li. Robust opacity in timed automata: notions and verification. *Reliability Engineering & System Safety*, 270:112093, 2026. URL: https://www.sciencedirect.com/science/article/pii/S095183202501292X, doi:10.1016/j.ress.2025.112093.

5 Jie An, Qiang Gao, Lingtai Wang, Naijun Zhan, and Ichiro Hasuo. The opacity of timed automata. In André Platzer, Kristin-Yvonne Rozier, Matteo Pradella, and Matteo Rossi, editors, *FM*, volume 14933 of *Lecture Notes in Computer Science*, pages 620–637. Springer, 2024. doi:10.1007/978-3-031-71162-6_32.

6 Étienne André, Johan Arcile, and Engel Lefaucheux. Execution-time opacity problems in one-clock parametric timed automata. In Siddharth Barman and Sławomir Lasota, editors, *FSTTCS*, volume 323 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, December 2024. doi:10.4230/LIPIcs.FSTTCS.2024.3.

7 Étienne André, Shapagat Bolat, Engel Lefaucheux, and Dylan Marinho. strategFTO: Untimed control for timed opacity. In Cyrille Artho and Peter Ölveczky, editors, *FTSCS*, pages 27–33. ACM, 2022. doi:10.1145/3563822.3568013.

8 Étienne André, Marie Duflot, Laetitia Laversa, and Engel Lefaucheux. Execution-time opacity control for timed automata. *Software and Systems Modeling*, 2026. To appear. doi:10.1007/s10270-026-01395-5.

9 Étienne André, Sarah Dépernet, and Engel Lefaucheux. The bright side of timed opacity. *Logical Methods in Computer Science*, 22:31:1–31:51, June 2026. doi:10.46298/lmcs-22(2:31)2026.

10 Étienne André, Engel Lefaucheux, Didier Lime, Dylan Marinho, and Jun Sun. Configuring timing parameters to ensure execution-time opacity in timed automata. In Maurice H. ter Beek and Clemens Dubslaff, editors, *TiCSA*, volume 392 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–26, 2023. Invited paper. doi:10.4204/EPTCS.392.1.

11 Étienne André, Sarah Dépernet, and Engel Lefaucheux. Buffered control for opacity in timed automata (extended version). Technical Report abs/2606.28170, arXiv, June 2026. arXiv:2606.28170.

12 Johan Arcile and Étienne André. Timed automata as a formalism for expressing security: A survey on theory and practice. *ACM Computing Surveys*, 55(6):1–36, July 2023. doi:10.1145/3534967.

13 Gilles Benattar, Franck Cassez, Didier Lime, and Olivier H. Roux. Control and synthesis of non-interferent timed systems. *International Journal of Control*, 88(2):217–236, 2015. doi:10.1080/00207179.2014.944356.

14 Nathalie Bertrand, Eric Fabre, Stefan Haar, Serge Haddad, and Loïc Hélouët. Active diagnosis for probabilistic systems. In Anca Muscholl, editor, *FoSSaCS*, volume 8412 of *Lecture Notes in Computer Science*, pages 29–42. Springer, 2014. doi:10.1007/978-3-642-54830-7_2.

15 Franck Cassez. The dark side of timed opacity. In Jong Hyuk Park, Hsiao-Hwa Chen, Mohammed Atiquzzaman, Changhoon Lee, Tai-Hoon Kim, and Sang-Soo Yeo, editors, *ISA*, volume 5576 of *Lecture Notes in Computer Science*, pages 21–30. Springer, 2009. doi:10.1007/978-3-642-02617-1_3.

16 Weilin Deng, Daowen Qiu, and Jingkai Yang. Initial-location opacity and infinite-step opacity of timed automata with integer resets. *IEEE Control Systems Letters*, 9:2031–2036, 2025. doi:10.1109/LCSYS.2025.3590422.

17 Weilin Deng, Daowen Qiu, and Jingkai Yang. New insights into opacity verification in timed discrete-event systems. *Automatica*, 186:112869, 2026. doi:10.1016/J.AUTOMATICA.2026.112869.

18 Julian Klein, Paul Kogel, and Sabine Glesner. Verifying opacity of discrete-timed automata. In Nico Plat, Stefania Gnesi, Carlo A. Furia, and Antónia Lopes, editors, *FormaliSE*, pages 55–65. ACM, 2024. doi:10.1145/3644033.3644376.

19 Jun Li, Dimitri Lefebvre, Christoforos N. Hadjicostis, and Zhiwu Li. Observers for a class of timed automata based on elapsed time graphs. *IEEE Transactions on Automatic Control*, 67(2):767–779, 2022. doi:10.1109/TAC.2021.3064542.

20 Jun Li, Dimitri Lefebvre, Christoforos N. Hadjicostis, and Zhiwu Li. Verification of state-based timed opacity for constant-time labeled automata. *IEEE Transactions on Automatic Control*, 70(1):503–509, 2025. doi:10.1109/TAC.2024.3432788.

21 Anthony Spriet, Didier Lime, and Olivier H. Roux. Timed non-interference under partial observability and bounded memory. In Laure Petrucci and Jeremy Sproston, editors, *FORMATS*, volume 14138 of *Lecture Notes in Computer Science*, pages 122–137. Springer, 2023. doi:10.1007/978-3-031-42626-1_8.

22 Lingtai Wang and Naijun Zhan. Decidability of the initial-state opacity of real-time automata. In Cliff B. Jones, Ji Wang, and Naijun Zhan, editors, *Symposium on Real-Time and Hybrid Systems - Essays Dedicated to Professor Chaochen Zhou on the Occasion of His 80th Birthday*, volume 11180 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2018. doi:10.1007/978-3-030-01461-2_3.

23 Lingtai Wang, Naijun Zhan, and Jie An. The opacity of real-time automata. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2845–2856, 2018. doi:10.1109/TCAD.2018.2857363.

24 Kuize Zhang. State-based opacity of labeled real-time automata. *Theoretical Computer Science*, 987:114373, 2024. doi:10.1016/J.TCS.2023.114373.

25 Yiwei Zheng, Aiwen Lai, Weiyao Lan, Xiao Yu, and Rong Su. Enforcement of opacity for interval weighted discrete-event system by supervisory control. *IEEE Transactions on Automatic Control*, 2025. doi:10.1109/TAC.2025.3611993.

A Recalling the region automaton

We recall the classic region automaton construction. Given a TA $\mathcal{A}$, for a clock x_i, we denote by c_i the largest constant to which x_i is compared within the guards and invariants of $\mathcal{A}$: formally: $c_i = \max_j\{d_j \mid x_i \bowtie d_j$ appears in a guard or invariant of $\mathcal{A}\}$.

Given a clock valuation μ and a clock x_i, $\lfloor \mu(x_i) \rfloor$ (resp. $\text{frac}(\mu(x_i))$) denotes the integral (resp. fractional) part of $\mu(x_i)$. We now recall the equivalence relation between clock valuations.

▶ **Definition 21** (Equivalence relation [1]). *Two clocks valuations μ, μ' are* equivalent, *denoted by $\mu \approx \mu'$, when the following three conditions hold for any clocks $x_i, x_j \in \mathbb{X}$:*

1. $\lfloor \mu(x_i) \rfloor = \lfloor \mu'(x_i) \rfloor$ *or* $\mu(x_i) > c_i$ *and* $\mu'(x_i) > c_i$;
2. *if* $\mu(x_i) \leq c_i$ *and* $\mu(x_j) \leq c_j$: $frac(\mu(x_i)) \leq frac(\mu(x_j))$ *iff* $frac(\mu'(x_i)) \leq frac(\mu'(x_j))$;
3. *if* $\mu(x_i) \leq c_i$: $frac(\mu(x_i)) = 0$ *iff* $frac(\mu'(x_i)) = 0$.

Given two states $s = (\ell, \mu), s' = (\ell', \mu')$ of $\mathfrak{T}_{\mathcal{A}}$, we write $s \approx s'$ iff $\ell = \ell'$ and $\mu \approx \mu'$. We denote by $[s]$ and call *region* the equivalence class of a state s for $\approx$. Then, $s' \in [s]$ when $s \approx s'$. The set of all regions of $\mathcal{A}$ is denoted $\mathsf{R}_{\mathcal{A}}$. A region $r = [(\ell, \mu)]$ is *final* whenever $\ell \in F$. The set of final regions is denoted by $\mathsf{R}^F_{\mathcal{A}}$. A region r is *reachable* when there exists a run ρ such that $last(\rho) \in r$ ($last(\rho)$ denoting the final configuration of ρ.

Given a state $s = (\ell, \mu)$, and $d \in \mathbb{R}_{\geq 0}$, we write $s + d$ to denote $(\ell, \mu + d)$. Given two regions r and r', we write $r \cup r'$ for $\{s \mid s \in r \text{ or } s \in r'\}$.

▶ **Definition 22** (Region Automaton). *For a given TA $\mathcal{A}$, the region automaton $\mathcal{R}_{\mathcal{A}}$ is given by the tuple $(\mathsf{R}_{\mathcal{A}}, \Sigma^{\times} \cup \{\varepsilon\}, \delta^{\mathsf{R}}, r_0 = (l_0, \vec{0}))$ where:* **1)** *$\mathsf{R}_{\mathcal{A}}$ is the set of states,* **2)** *given two regions $r, r' \in \mathsf{R}_{\mathcal{A}}$ and $a \in \Sigma^{\times} \cup \{\varepsilon\}$, we have $(r, a, r') \in \delta^{\mathsf{R}}$ if there exist $s = (\ell, \mu) \in r$ and $s' = (\ell', \mu') \in r'$ such that one of the following holds:* **i)** *$a \in \Sigma^{\times}$ and $(\ell, \mu) \overset{e}{\mapsto} (\ell', \mu') \in \to$ in $\mathfrak{T}_{\mathcal{A}}$ with $e = (\ell, g, a, R, \ell')$ for some g and R; or* **ii)** *$a = \varepsilon$ and $\exists d \in \mathbb{R}_{>0}$ such that $s \overset{d}{\mapsto} s'$, and $\forall 0 < d' < d$, $s + d' \in r \cup r'$.*

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

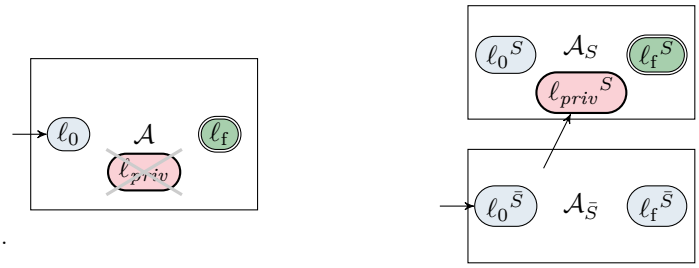
In this section, we give a quick intuition of the constructions $\mathcal{A}_{pub}$ and $\mathcal{A}_{priv}$ (defined earlier, e.g., in [9]), that recognize timed words produced respectively by public and private runs of a given TA $\mathcal{A}$. Those constructions are also illustrated in Figures 6 and 7. We refer to [9] for the formal definitions.

The public runs TA $\mathcal{A}_{pub}$ is the easiest to build: it suffices to remove the private locations from $\mathcal{A}$. The remaining runs are exactly the ones that do not visit the private locations, thus the public runs.

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

C Buffered automaton

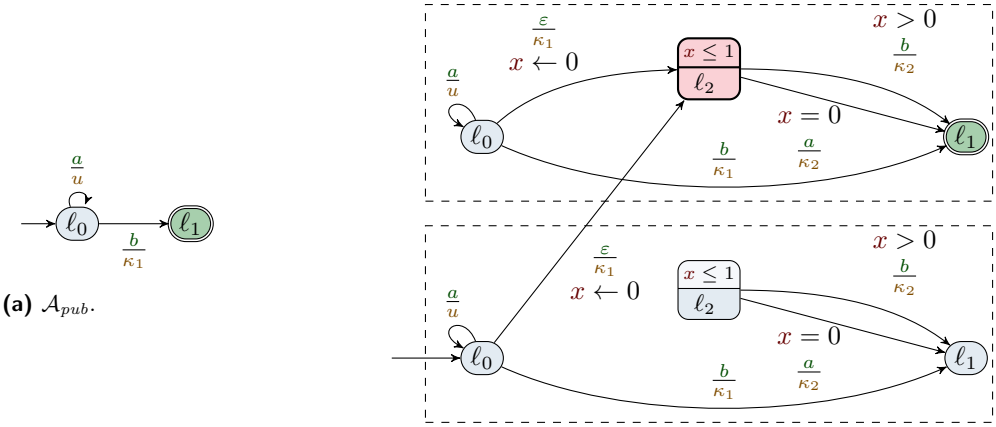
Given a TA $\mathcal{A}$, we give the construction of a TA $buff(\mathcal{A})$ whose untimed language is exactly the set of buffered observations of $\mathcal{A}$. Simply said, we make two copies of the TA, encoding the Boolean value whether the elapsed time is an integer or not, and add a clock z (encoding the fractional part of the global time) to switch between both parts at the right moment.



(a) $\mathcal{A}_{pub}$.

(b) $\mathcal{A}_{priv}$.

Figure 6 Illustrating $\mathcal{A}_{pub}$ and $\mathcal{A}_{priv}$.



(a) $\mathcal{A}_{pub}$.

(b) $\mathcal{A}_{priv}$.

Figure 7 $\mathcal{A}_{pub}$ and $\mathcal{A}_{priv}$ with the example from Figure 2.

Each location ℓ is duplicated in two locations ℓ^0 and ℓ^1. A transition (ℓ, a, g, R, ℓ') in $\mathcal{A}$ is decomposed into two new transitions $(\ell^i, a, g \wedge g_i, R, \ell'^i)$ in $\mathit{buff}(\mathcal{A})$, for $i \in \{0,1\}$, with $g_0 = (z = 0)$ and $g_1 = (0 < z) \wedge (z < 1)$.

To switch between both sets of locations, we add for each location ℓ in $\mathcal{A}$ two transitions in $\mathit{buff}(\mathcal{A})$: $(\ell^1, \frac{\blacktriangleright}{u}, (z = 1), \{0\}, \ell^0)$ (denoting that the global time has reached an integer value), and $(\ell^0, \frac{\triangleright}{u}, g_1, \emptyset, \ell^1)$ (denoting that some time smaller than 1 time unit has elapsed since the last integer global time). This way, the untimed trace of a timed word contains time marks allowing to know between which integer times an action took place but also whether an action took place exactly at an integer time.

If ℓ (in $\mathcal{A}$) was an initial location, then ℓ^0 is initial; however if ℓ was final, then none of ℓ^0 and ℓ^1 are final, for the following reason.

In the following proofs of the appendix, we restrict the opacity study to states reached by a transition labelled by a time interval symbol ($\blacktriangleright$ or $\triangleright$). Hence all relevant events of a run must happen before the last time interval symbol occurs, which we add artificially as explained below. To end with a time interval symbol without forgetting when the run ended, we first add four locations ℓ^0_{end}, ℓ^1_{end}, ℓ^0_{reg} and ℓ^1_{reg}, and a special observable symbol $\bigtriangledown$ that marks the end of the run. If ℓ was a final location, then for $i \in \{0,1\}$, ℓ^i leads uncontrollably and in zero time to ℓ^i_{end}, producing the symbol $\bigtriangledown$. To ensure that this is done in zero time, every transition leading to ℓ^i_{end} resets a clock y, then compared to zero. Finally, we add two new final locations ℓ^0_{reg} and ℓ^1_{reg} respectively reachable from ℓ^1_{end} and ℓ^0_{end} through a transition announcing the next time interval (with $\frac{\blacktriangleright}{u}$ or $\frac{\triangleright}{u}$ resp.).

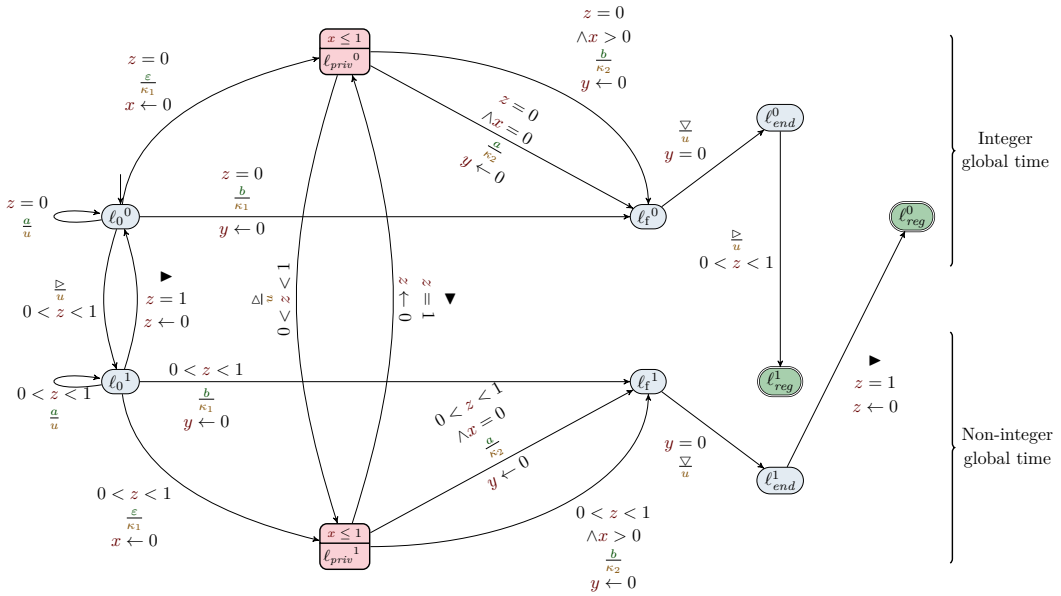


Figure 8 The buffered automaton of the TA from Figure 2.

With this construction, the set of durations of runs in $\mathcal{A}$ is exactly the set of times when a symbol $\bigtriangledown$ can be produced in $\mathit{buff}(\mathcal{A})$.

D Inter-reduction between weak and full opacity

▶ **Theorem 10.** *The weak and full variants of the $\exists SO$, $\exists SO_{NB}$, $\exists N\text{-}SSO$, $\exists N\text{-}SSO_{NB}$, $\exists OSSO$ and $\exists OSSO_{NB}$ problems are inter-reducible.*

Proof. In [9], the weak ET-opacity and full ET-opacity problems were shown to be inter-reducible in the absence of control. The proof mainly relied on the ability to construct from a TA $\mathcal{A}$ another TA $\mathcal{A}'$ where private and public runs were swapped. Hence, $\mathcal{A}$ is fully opaque iff both $\mathcal{A}$ and $\mathcal{A}'$ are weakly opaque. This proof cannot carry over to the controlled setting since there is no guarantee that the same strategy can be applied in both TAs to make them *simultaneously* weakly opaque: both $\mathcal{A}$ and $\mathcal{A}'$ could be deemed weakly ET-opaque, though relying on two different sequential strategies. While their proof cannot be adapted, we reuse some of their proof ingredients, mainly $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$, i.e., copies of the TA $\mathcal{A}$ encoding respectively all private and public behaviours (see details in Appendix B): given a sequential strategy σ, we have that $\mathit{PrivTr}^\sigma(\mathcal{A}) = \mathit{Tr}(\Omega^{\mathcal{A}_{priv}}(\sigma))$ and $\mathit{PubTr}^\sigma(\mathcal{A}) = \mathit{Tr}(\Omega^{\mathcal{A}_{pub}}(\sigma))$.

Let us first explain how to reduce weak opacity problems to full opacity problems. Given a TA $\mathcal{A}$, we first build the TA $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$ mentioned above, and then the TA $\mathcal{A}_{weak\to full}$ (shown on the left of Figure 9) such that the initial location of $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$ can be reached in 0 time while visiting a private location, and the initial location of $\mathcal{A}_{pub}$ can be visited in 0 time without visiting a private location. Thus, $\mathit{PrivTr}^\sigma(\mathcal{A}_{weak\to full}) = \mathit{Tr}(\Omega^{\mathcal{A}_{priv}}(\sigma)) \cup \mathit{Tr}(\Omega^{\mathcal{A}_{pub}}(\sigma)) = \mathit{Tr}(\Omega^{\mathcal{A}}(\sigma))$ and $\mathit{PubTr}^\sigma(\mathcal{A}_{weak\to full}) = \mathit{Tr}(\Omega^{\mathcal{A}_{pub}}(\sigma)) = \mathit{PubTr}^\sigma(\mathcal{A})$. Hence, $\mathit{PrivTr}^\sigma(\mathcal{A}) \subseteq \mathit{PubTr}^\sigma(\mathcal{A})$ if and only if $\mathit{PrivTr}^\sigma(\mathcal{A}_{weak\to full}) = \mathit{PubTr}^\sigma(\mathcal{A}_{weak\to full})$. Thus the strategy σ makes $\mathcal{A}$ weakly opaque if and only if it makes $\mathcal{A}_{weak\to full}$ fully opaque.

We now consider the opposite reduction. As mentioned previously, in [9] we had made this reduction by testing both inclusion $\mathit{PrivTr}^\sigma(\mathcal{A}) \subseteq \mathit{PubTr}^\sigma(\mathcal{A})$ and $\mathit{PrivTr}^\sigma(\mathcal{A}) \supseteq \mathit{PubTr}^\sigma(\mathcal{A})$ separately on two different TAs. Here, instead, we use a single TA, and add a final letter

("a" or "b") pointing which inclusion is being tested: if the last letter of the trace is an "a", then the run participates in the test of the inclusion $PrivTr^\sigma(\mathcal{A}) \subseteq PubTr^\sigma(\mathcal{A})$; and if the last letter is a "b", then the run participates in the converse inclusion. As a strategy is not aware before this last letter of which inclusion is being tested, it needs to ensure both simultaneously.

More precisely, we detail below the construction of the TA $\mathcal{A}_{full\to weak}$ (shown on the right of Figure 9), where we assume '$\sharp$' is a new letter (and thus cannot be produced through any transition besides the ones we add in the reduction):

- From a new initial location ℓ_0, one can go to the initial location of either $\mathcal{A}_{priv}$ or $\mathcal{A}_{pub}$ in 0 time;
- Within $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$, each transition leading to a final location of $\mathcal{A}_{priv}$ or $\mathcal{A}_{pub}$ resets x; and these final locations are not deemed final in $\mathcal{A}_{full\to weak}$;
- Once a final locations of $\mathcal{A}_{priv}$ is reached, a run can immediately (requiring $x = 0$) leave while reading a $\sharp$ and go either to ℓ^a (which is private) or to ℓ^b. Similarly, once a final locations of $\mathcal{A}_{pub}$ is reached, a run can immediately (requiring $x = 0$) leave while reading a $\sharp$ and go either to ℓ_a or to ℓ_b (which is private).
- From the locations ℓ^z and ℓ_z (with $z \in \{a, b\}$) the run can finally reach the final location ℓ_f of $\mathcal{A}_{full\to weak}$ by reading a z.

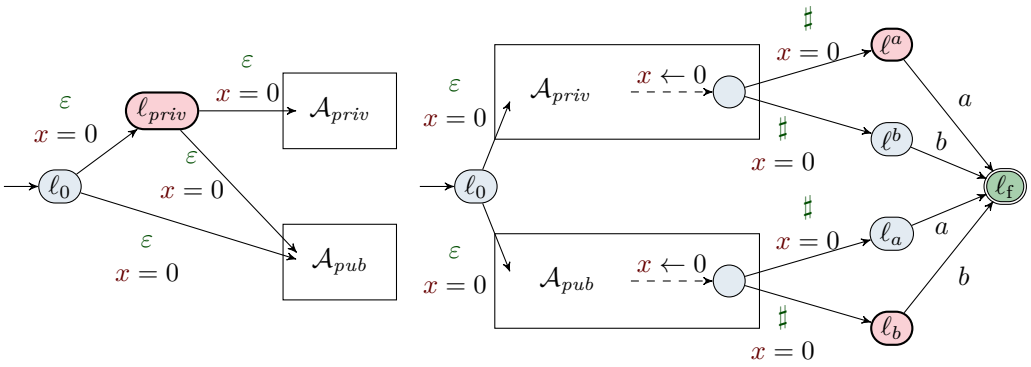


Figure 9 Left: $\mathcal{A}_{weak\to full}$, fully opaque iff $\mathcal{A}$ is weakly opaque; right: $\mathcal{A}_{full\to weak}$ weakly opaque iff $\mathcal{A}$ is fully opaque. Every transition outside $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$ is uncontrollable.

By construction, we have that a trace w is produced by a private run of $\mathcal{A}$ (and thus by a run of $\mathcal{A}_{priv}$) iff the trace $w\sharp a$ is a private trace of $\mathcal{A}_{full\to weak}$ and the trace $w\sharp b$ is a public trace of $\mathcal{A}_{full\to weak}$. And also that a trace w is produced by a public run of $\mathcal{A}$ (and thus by a run of $\mathcal{A}_{pub}$) iff the trace $w\sharp a$ is a public trace of $\mathcal{A}_{full\to weak}$ and the trace $w\sharp b$ is a private trace of $\mathcal{A}_{full\to weak}$.

As $\sharp$ can only be produced once by the runs exiting the TA $\mathcal{A}_{priv}$ and $\mathcal{A}_{pub}$, given a sequential strategy σ, in order to have $PrivTr^\sigma(\mathcal{A}_{full\to weak}) \subseteq PubTr^\sigma(\mathcal{A}_{full\to weak})$ we need that the traces of runs exiting $\mathcal{A}_{priv}$ (resp. $\mathcal{A}_{pub}$) and followed by $\sharp a$ (resp. $\sharp b$) are made opaque by traces of runs exiting $\mathcal{A}_{pub}$ (resp. $\mathcal{A}_{priv}$) and followed by $\sharp a$ (resp. $\sharp b$). Those two conditions thus require that $Tr(\Omega^{\mathcal{A}_{priv}}(\sigma)) \subseteq Tr(\Omega^{\mathcal{A}_{pub}}(\sigma))$ and $Tr(\Omega^{\mathcal{A}_{priv}}(\sigma)) \supseteq Tr(\Omega^{\mathcal{A}_{pub}}(\sigma))$ respectively, and thus that $Tr(\Omega^{\mathcal{A}_{priv}}(\sigma)) = Tr(\Omega^{\mathcal{A}_{pub}}(\sigma))$.

Hence, $PrivTr^\sigma(\mathcal{A}_{full\to weak}) \subseteq PubTr^\sigma(\mathcal{A}_{full\to weak})$ iff $PrivTr^\sigma(\mathcal{A}) = PubTr^\sigma(\mathcal{A})$.

As the N-SS setting only limits the strategy to consider, and we do not add any controllable action in our reduction, this inter-reducibility directly applies to both the N-SS setting and the OSS setting as well as the general one. ◀