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

Enforcement of (Timed) Properties with Uncontrollable Events

MATTHIEU RENARD[1], YLIÈS FALCONE[2], ANTOINE ROLLET[1],

THIERRY JÉRON[3], and HERVÉ MARCHAND[3]

[1] *LaBRI, Bordeaux INP, Université Bordeaux, Bordeaux, France.*
[2] *Univ. Grenoble Alpes, Inria, LIG, F-38000 Grenoble, France.*
[3] *Inria Rennes Bretagne-Atlantique, Rennes, France.*

Received April 25, 2017

This paper deals with runtime enforcement of untimed and timed properties with uncontrollable events. Runtime enforcement consists in defining and using mechanisms that modify the executions of a running system to ensure their correctness with respect to a desired property. We introduce a framework that takes as input any regular (timed) property described by a deterministic automaton over an alphabet of events, with some of these events being uncontrollable. An uncontrollable event cannot be delayed nor intercepted by an enforcement mechanism. Enforcement mechanisms should satisfy important properties, namely soundness, compliance, and optimality - meaning that enforcement mechanisms should output as soon as possible correct executions that are as close as possible to the input execution. We define the conditions for a property to be enforceable with uncontrollable events. Moreover, we synthesise sound, compliant, and optimal descriptions of runtime enforcement mechanisms at two levels of abstraction to facilitate their design and implementation.

1. Introduction

Runtime verification is a technique aiming at checking the conformance of the executions of a system under scrutiny w.r.t. some specification. It consists in running a mechanism that assigns verdicts to a sequence of events produced by the instrumented system w.r.t. a property formalising the specification. This paper focuses on *runtime enforcement* (cf. (Schneider, 2000; Ligatti et al., 2009; Falcone et al., 2011; Basin et al., 2013)) which goes beyond pure verification at runtime and studies how to react to a violation of specifications. In runtime enforcement, an enforcement mechanism (EM) takes a (possibly incorrect) execution sequence as input, and outputs a new sequence. Enforcement mechanisms should be *sound* and *transparent*, meaning that the output should satisfy the property under consideration and should be as close as possible to the input, respectively. When dealing with timed properties, EMs can act as *delayers* over the input sequence of events (Pinisetty et al., 2012; Pinisetty et al., 2014b; Pinisetty et al., 2014c). That is, whenever possible, EMs buffer input events for some time and then release them in such a way that the output sequence of events satisfies the property. The general scheme is given in Fig. 1.

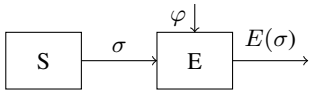


Figure 1: Schematic description of an enforcement mechanism E, modifying the execution σ of the system S to $E(\sigma)$, so that it satisfies the property φ.

Motivations. We focus on enforcement of properties with uncontrollable events. Introducing uncontrollable events is a step towards more realistic runtime enforcement. Uncontrollable events naturally occur in many applications scenarios where the EM has no control over certain input events. For instance, certain events from the environment may be out of the scope of the mechanism at hand. This situation arises for instance in avionic systems where a command of the pilot has consequences on a specific component. In this critical domain, it is usual to add control mechanisms in specific points of the architecture in order to verify that nothing wrong happens. Some events may only be observed by these mechanisms in order to decide if a situation is abnormal, but they cannot be acted upon, meaning that they are uncontrollable. For instance, the "spoiler activation" command triggered by the pilot is sent by the panel to a control flight system (to reduce the lift of an aircraft), and this leads finally to a specific event on the spoilers. Placing an EM directly on the spoiler permits to prevent events leading to an incoherent state by blocking them, according to the pilot commands. The pilot commands are out of the scope of the EM, i.e. observable but uncontrollable. In the timed setting, uncontrollable events may be urgent messages that cannot be delayed by an enforcement mechanism. Similarly, when a data-dependency exists between two events (e.g., between a *write* event that displays a value obtained from a previous *read* event), the first *read* event is somehow uncontrollable as it cannot be delayed by the enforcement mechanism without preventing the *write* event to occur in the monitored program.

Challenges. Considering uncontrollable events induces new challenges. Indeed, enforcement mechanisms may receive events that cannot be buffered and have to be output immediately. Since uncontrollable events influence the satisfaction of the property under scrutiny, the dates of the controllable events stored in memory have to be recomputed upon the reception of each uncontrollable event to guarantee that the property is still satisfied after outputting them. Moreover, it is necessary to prevent the system from reaching a bad state upon the reception of any sequence of uncontrollable events. Since uncontrollable events can occur at any time, the EM must take their potential reception into account when computing the sequence to be emitted. It turns out that a property may not be enforceable because of certain input sequences. Intuitively, enforceability issues arise because some sequences of uncontrollable events that lead the property to be violated cannot be avoided. Thus, new enforcement strategies are necessary for both untimed and timed properties.

Contributions. We introduce a framework for enforcement monitoring for regular untimed and timed properties with uncontrollable events, where properties are described by automata. We define enforcement mechanisms at two levels of abstraction. The synthesised enforcement mechanisms are sound, compliant and optimal. When considering uncontrollable events, it turns out that the usual notion of transparency has to be weakened. As we shall see, the initial order between

uncontrollable and controllable events can change in output, contrary to what is prescribed by transparency. Thus, we replace transparency with a new notion, namely *compliance*, prescribing that the order of controllable events is maintained while uncontrollable events are output as soon as they are received. We define a property to be enforceable with uncontrollable events when it is possible to obtain a sound and compliant enforcement mechanism for any input sequence. In the timed setting, the executions are associated with dates from which the property is enforceable.

This paper revisits and extends a first approach to the runtime enforcement with uncontrollable events proposed in (Renard et al., 2015). Most definitions have been refined in order to ensure optimality of the mechanism for any regular property. All the proofs of soundness, compliance, optimality and equivalence between the different descriptions of the enforcement mechanism are provided. This new framework can also be used without uncontrollable event. Finally, enforcement mechanisms output the longest possible words in terms of number of events.

Outline. Section 2 introduces preliminaries and notations. Sections 3 and 4 present the enforcement framework with uncontrollable events in the untimed and timed settings, respectively. In each setting, we define enforcement mechanisms at two levels of abstraction. Section 5 discusses related work. Section 6 concludes. Lemmas and proofs of the theorems are given in Appendix A.

2. Preliminaries and Notation

Untimed notions. An *alphabet* is a finite set of symbols. A *word* over an alphabet Σ is a sequence over Σ. The set of finite words over Σ is denoted Σ^*. The *length* of a finite word w is noted $|w|$, and the *empty word* is noted ϵ. Σ^+ stands for $\Sigma^* \setminus \{\epsilon\}$. A *language* over Σ is any subset $L \subseteq \Sigma^*$. The concatenation of two words w and w' is noted $w.w'$ (the dot is omitted when clear from the context). A word w' is a *prefix* of a word w, noted $w' \preccurlyeq w$ if there exists a word w'' such that $w = w'.w''$. The word w'' is called the *residual* of w after reading the prefix w', noted $w'' = {w'}^{-1}.w$. Note that $w'.w'' = w'.{w'}^{-1}.w = w$. These definitions are extended to languages in the natural way. A language $L \subseteq \Sigma^*$ is *extension-closed* if for any words $w \in L$ and $w' \in \Sigma^*$, $w.w' \in L$. Given a word w and an integer i such that $1 \leq i \leq |w|$, we note $w(i)$ the i-th element of w. Given a tuple $e = (e_1, e_2, \ldots, e_n)$ of size n, for an integer i such that $1 \leq i \leq n$, we note Π_i the projection on the i-th coordinate, i.e. $\Pi_i(e) = e_i$. The tuple $(e_1, e_2, \ldots, e_n)$ is sometimes noted $\langle e_1, e_2, \ldots, e_n \rangle$ in order to help reading. It can be used, for example, if a tuple contains a tuple. Given a word $w \in \Sigma^*$ and $\Sigma' \subseteq \Sigma$, we define the *restriction* of w to Σ', noted $w_{|\Sigma'}$, as the word $w' \in \Sigma'^*$ whose letters are the letters of w belonging to Σ' in the same order. Formally, $\epsilon_{|\Sigma'} = \epsilon$ and $\forall \sigma \in \Sigma^*, \forall a \in \Sigma$, $(w.a)_{|\Sigma'} = w_{|\Sigma'}.a$ if $a \in \Sigma'$, and $(w.a)_{|\Sigma'} = w_{|\Sigma'}$ otherwise. We also note $=_{\Sigma'}$ the equality of the restrictions of two words to Σ': if σ and σ' are two words, $\sigma =_{\Sigma'} \sigma'$ if $\sigma_{|\Sigma'} = \sigma'_{|\Sigma'}$. We define in the same way $\preccurlyeq_{\Sigma'}$: $\sigma \preccurlyeq_{\Sigma'} \sigma'$ if $\sigma_{|\Sigma'} \preccurlyeq \sigma'_{|\Sigma'}$.

Automata. An *automaton* is a tuple $\langle Q, q_0, \Sigma, \rightarrow, F \rangle$, where Q is the set of *states*, $q_0 \in Q$ is the initial state, Σ is the alphabet, $\rightarrow \subseteq Q \times \Sigma \times Q$ is the transition relation and $F \subseteq Q$ is the set of accepting states. Whenever there exists $(q, a, q') \in \rightarrow$, we note it $q \xrightarrow{a} q'$. Relation $\rightarrow$ is extended to words $\sigma \in \Sigma^*$ by noting $q \xrightarrow{\sigma.a} q'$ whenever there exists q'' such that $q \xrightarrow{\sigma} q''$ and $q'' \xrightarrow{a} q'$. Moreover, for any $q \in Q$, $q \xrightarrow{\epsilon} q$ always holds. An automaton $\mathcal{A} = \langle Q, q_0, \Sigma, \rightarrow, F \rangle$ is *deterministic* if $\forall q \in Q, \forall a \in \Sigma, (q \xrightarrow{a} q' \wedge q \xrightarrow{a} q'') \implies q' = q''$. $\mathcal{A}$ is *complete* if $\forall q \in$

$Q, \forall a \in \Sigma, \exists q' \in Q, q \xrightarrow{a} q'$. A word w is *accepted* by $\mathcal{A}$ if there exists $q \in F$ such that $q_0 \xrightarrow{w} q$. The language (i.e. set of all words) accepted by $\mathcal{A}$ is noted $\mathcal{L}(\mathcal{A})$. A *property* is a language over an alphabet Σ. A regular property is a language accepted by an automaton. In the sequel, we shall assume that a property φ is represented by a deterministic and complete automaton $\mathcal{A}_\varphi$. For example, in Fig. 2a, the set of states is $Q = \{q_0, q_1, q_2, q_3\}$, the initial state is q_0, the alphabet is $\Sigma = \{\texttt{Auth}, \texttt{LockOff}, \texttt{LockOn}, \texttt{Write}\}$, the set of accepting states is $F = \{q_1, q_2\}$, and the transitions relation $\rightarrow$ contains for instance $(q_0, \texttt{Auth}, q_1), (q_1, \texttt{LockOn}, q_2), (q_3, \texttt{LockOn}, q_3)$.

Timed languages. Let $\mathbb{R}_{\geq 0}$ be the set of non-negative real numbers, and Σ a finite alphabet of actions. An event is a pair $(t, a) \in \mathbb{R}_{\geq 0} \times \Sigma$. We define $\mathrm{date}(t, a) = t$ and $\mathrm{act}(t, a) = a$ the projections of events on dates and actions respectively. A *timed word* over Σ is a word over $\mathbb{R}_{\geq 0} \times \Sigma$ whose real parts are ascending, i.e. σ is a timed word if $\sigma \in (\mathbb{R}_{\geq 0} \times \Sigma)^*$ and $\forall i \in [1; |\sigma| - 1], \mathrm{date}(w(i)) \leq \mathrm{date}(w(i+1))$. $\mathrm{tw}(\Sigma)$ denotes the set of timed words over Σ. For a timed word $\sigma = (t_1, a_1).(t_2, a_2) \ldots (t_n, a_n)$ and an integer i such that $1 \leq i \leq n$, t_i is the time elapsed before action a_i occurs. We naturally extend the notions of *prefix* and *residual* to timed words. We note $\mathrm{time}(\sigma) = \mathrm{date}(\sigma(|\sigma|))$ for $\sigma \neq \epsilon$, and $\mathrm{time}(\epsilon) = 0$. We define the *observation* of σ at time t as the timed word $\mathrm{obs}(\sigma, t) = \max_{\preccurlyeq}(\{\sigma' \mid \sigma' \preccurlyeq \sigma \wedge \mathrm{time}(\sigma') \leq t\})$, corresponding to the word that would be observed at date t if events were received at the date they are associated with. We also define the remainder of the observation of σ at time t as $\mathrm{nobs}(\sigma, t) = (\mathrm{obs}(\sigma, t))^{-1}.\sigma$, which corresponds to the events that are to be received after date t. The *untimed projection* of σ is $\Pi_\Sigma(\sigma) = a_1.a_2 \ldots a_n$, it is the sequence of actions of σ with dates ignored. σ *delayed by* $t \in \mathbb{R}_{\geq 0}$ is the word noted $\sigma +_{\mathrm{t}} t$ such that t is added to all dates: $\sigma +_{\mathrm{t}} t = (t_1 + t, a_1).(t_2 + t, a_2) \ldots (t_n + t, a_n)$. Similarly, we define $\sigma -_{\mathrm{t}} t$, when $t_1 \geq t$, to be the word $(t_1 - t, a_1).(t_2 - t, a_2) \ldots (t_n - t, a_n)$. We also extend the definition of the restriction of σ to $\Sigma' \subseteq \Sigma$ to timed words, such that $\epsilon_{|\Sigma'} = \epsilon$, and for $\sigma \in \mathrm{tw}(\Sigma)$ and (t, a) such that $\sigma.(t, a) \in \mathrm{tw}(\Sigma)$, $(\sigma.(t, a))_{|\Sigma'} = \sigma_{|\Sigma'}.(t, a)$ if $a \in \Sigma'$, and $(\sigma.(t, a))_{|\Sigma'} = \sigma_{|\Sigma'}$ otherwise. The notations $=_{\Sigma'}$ and $\preccurlyeq_{\Sigma'}$ are then naturally extended to timed words. A *timed language* is any subset of $\mathrm{tw}(\Sigma)$. The notion of *extension-closed* languages is naturally extended to timed languages. We also extend the notion of extension-closed languages to sets of elements composed of a timed word and a date: a set $S \subseteq \mathrm{tw}(\Sigma) \times \mathbb{R}_{\geq 0}$ is *time-extension-closed* if for any $(\sigma, t) \in S$, for all $w \in \mathrm{tw}(\Sigma)$ such that $\sigma.w \in \mathrm{tw}(\Sigma)$, for all $t' \geq t$, $(\sigma.w, t') \in S$. In other words, S is time-extension-closed if for every $\sigma \in \mathrm{tw}(\Sigma)$, there exists a date t from which σ and all its extensions are in S, that is, associated with a date greater or equal to t. Moreover, we define an order on timed words: we say that σ' is a *delayed prefix* of σ, noted $\sigma \preccurlyeq_d \sigma'$, whenever $\Pi_\Sigma(\sigma') \preccurlyeq \Pi_\Sigma(\sigma)$ and $\forall i \in [1; |\sigma'| - 1], \mathrm{date}(\sigma(i)) \leq \mathrm{date}(\sigma'(i))$. Note that the order is not the same in the different constraints: $\Pi_\Sigma(\sigma')$ is a prefix of $\Pi_\Sigma(\sigma)$, but dates in σ' exceed dates in σ. As for the equality $=$ and the prefix order $\preccurlyeq$, we note $\sigma \preccurlyeq_{d\Sigma'} \sigma'$ whenever $\sigma_{|\Sigma'} \preccurlyeq_d \sigma'_{|\Sigma'}$. We also define a *lexical order* $\leq_{\mathrm{lex}}$ on timed words with identical untimed projections, such that $\epsilon \leq_{\mathrm{lex}} \epsilon$, and for two words σ and σ' such that $\Pi_\Sigma(\sigma) = \Pi_\Sigma(\sigma')$, and two events (t, a) and (t', a), $(t', a).\sigma' \leq_{\mathrm{lex}} (t, a).\sigma$ if $t' < t \vee (t = t' \wedge \sigma' \leq_{\mathrm{lex}} \sigma)$.

Consider for example the timed word $\sigma = (1, a).(2, b).(3, c).(4, a)$ over the alphabet $\Sigma = \{a, b, c\}$. Then, $\Pi_\Sigma(\sigma) = a.b.c.a$, $\mathrm{obs}(\sigma, 3) = (1, a).(2, b).(3, c)$, $\mathrm{nobs}(\sigma, 3) = (4, a)$, and if $\Sigma' = \{b,\ c\}$, $\sigma_{|\Sigma'} = (2,\ b)\ .\ (3,\ c)$, and for instance $\sigma \preccurlyeq_d (1,\ a)\ .\ (2,\ b)\ .\ (4,\ c)$, and $\sigma \leq_{\mathrm{lex}} (1, a).(3, b).(3, c).(3, a)$. Moreover, if $w = (1, a).(2, b)$, then $w^{-1}.\sigma = (3, c).(4, a)$.

Timed automata. Let $X = \{X_1, X_2, \ldots, X_n\}$ be a finite set of *clocks*, i.e. variables that increase regularly with time. A *clock valuation* is a function ν from X to $\mathbb{R}_{\geq 0}$. The set of clock valuations for the set of clocks X is noted $\mathcal{V}(X)$, i.e., $\mathcal{V}(X) = \{\nu \mid \nu : X \to \mathbb{R}_{\geq 0}\}$. We consider the following operations on valuations: for any valuation ν, $\nu + \delta$ is the valuation assigning $\nu(X_i) + \delta$ to every clock $X_i \in X$; for any subset $X' \subseteq X$, $\nu[X' \leftarrow 0]$ is the valuation assigning 0 to each clock in X', and $\nu(X_i)$ to any other clock X_i not in X'. $\mathcal{G}(X)$ denotes the set of guards consisting of boolean combinations of constraints of the form $X_i \bowtie c$ with $X_i \in X$, $c \in \mathbb{N}$, and $\bowtie \in \{<, \leq, =, \geq, >\}$. Given $g \in \mathcal{G}(X)$ and a valuation ν, we write $\nu \models g$ when for every constraint $X_i \bowtie c$ in g, $\nu(X_i) \bowtie c$ holds.

Definition 1 (Timed automaton (Alur and Dill, 1992)). A *timed automaton* (TA) is a tuple $\mathcal{A} = \langle L, l_0, X, \Sigma, \Delta, G\rangle$, such that L is a set of locations, $l_0 \in L$ is the initial location, X is a set of clocks, Σ is a finite set of events, $\Delta \subseteq L \times \mathcal{G}(X) \times \Sigma \times 2^X \times L$ is the transition relation, and $G \subseteq L$ is a set of accepting locations. A transition $(l, g, a, X', l') \in \Delta$ is a transition from l to l', labelled with event a, with guard g, and with the clocks in X' to be reset.

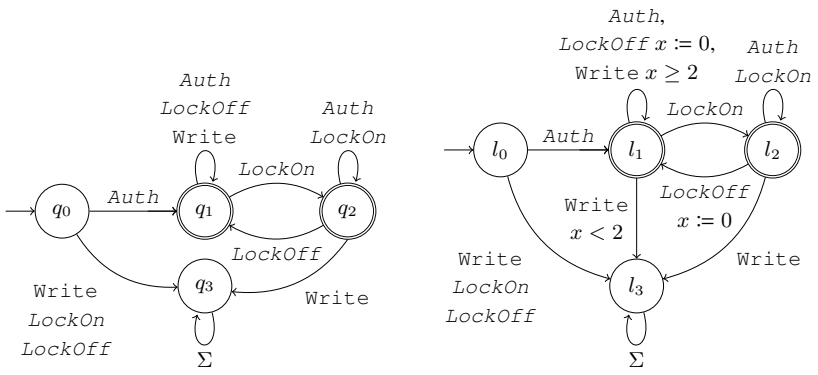
The semantics of a timed automaton $\mathcal{A}$ is a timed transition system $[\![\mathcal{A}]\!] = \langle Q, q_0, \Gamma, \to, F_G\rangle$ where $Q = L \times \mathcal{V}(X)$ is the (infinite) set of states, $q_0 = (l_0, \nu_0)$ is the initial state, with $\nu_0 = \nu[X \leftarrow 0]$, $F_G = G \times \mathcal{V}(X)$ is the set of accepting states, $\Gamma = \mathbb{R}_{\geq 0} \times \Sigma$ is the set of transition labels, each one composed of a delay and an action. The transition relation $\to \subseteq Q \times \Gamma \times Q$ is a set of transitions of the form $(l, \nu) \xrightarrow{(\delta, a)} (l', \nu')$ with $\nu' = (\nu + \delta)[Y \leftarrow 0]$ whenever there is a transition $(l, g, a, Y, l') \in \Delta$ such that $\nu + \delta \models g$, for $\delta \geq 0$.

A timed automaton $\mathcal{A} = \langle L, l_0, X, \Sigma, \Delta, G\rangle$ is *deterministic* if for any (l, g_1, a, Y_1, l'_1) and (l, g_2, a, Y_2, l'_2) in Δ, $g_1 \wedge g_2$ is unsatisfiable, meaning that only one transition can be fired at any time. $\mathcal{A}$ is *complete* if for any $l \in L$ and any $a \in \Sigma$, the disjunction of the guards of all the transitions leaving l and labelled by a is valid (i.e., it holds for any clock valuation). An example of timed automaton is given in Fig. 2b.

A *run* ρ from $q \in Q$ is a valid sequence of transitions in $[\![\mathcal{A}]\!]$ starting from q, of the form $\rho = q \xrightarrow{(\delta_1, a_1)} q_1 \xrightarrow{(\delta_2, a_2)} q_2 \ldots \xrightarrow{(\delta_n, a_n)} q_n$. The set of runs from q_0 is noted $\mathrm{Run}(\mathcal{A})$ and $\mathrm{Run}_{F_G}(\mathcal{A})$ denotes the subset of runs accepted by $\mathcal{A}$, i.e. ending in a state in F_G. The *trace* of the run ρ previously defined is the timed word $(t_1, a_1).(t_2, a_2)\ldots(t_n, a_n)$, with, for $1 \leq i \leq n$, $t_i = \sum_{k=1}^{i} \delta_k$. Thus, given the trace $\sigma = (t_1, a_1).(t_2, a_2)\ldots(t_n, a_n)$ of a run ρ from a state $q \in Q$ to $q' \in Q$, we can define $w = (\delta_1, a_1).(\delta_2, a_2)\ldots(\delta_n, a_n)$, with $\delta_1 = t_1$, and $\forall i \in [2; n], \delta_i = t_i - t_{i-1}$, and then $q \xrightarrow{w} q'$. To ease the notation, we will only consider traces and note $q \xrightarrow{\sigma} q'$ whenever $q \xrightarrow{w} q'$ for the previously defined w. Note that to concatenate two traces σ_1 and σ_2, it is needed to delay σ_2 to obtain a trace: the concatenation σ of σ_1 and σ_2 is the trace defined as $\sigma = \sigma_1.(\sigma_2 +_{\mathrm{t}} \mathrm{time}(\sigma_1))$. Thus, if $q \xrightarrow{\sigma_1} q' \xrightarrow{\sigma_2} q''$, then $q \xrightarrow{\sigma} q''$.

Timed properties. A *regular timed property* is a timed language $\varphi \subseteq \mathrm{tw}(\Sigma)$ accepted by a timed automaton. For a timed word σ, we say that σ *satisfies* φ, noted $\sigma \models \varphi$ whenever $\sigma \in \varphi$. We only consider regular timed properties whose associated automaton is complete and deterministic.

Given a deterministic automaton $\mathcal{A}$ such that Q is the set of states of $[\![\mathcal{A}]\!]$ and $\to$ its transition relation, and a word σ, for $q \in Q$, we note $q \,\mathrm{after}\, \sigma = q'$, where q' is such that $q \xrightarrow{\sigma} q'$. Since $\mathcal{A}$



(a) Property φ_{ex} (without time)

(b) Timed property φ_{t}

Figure 2: Examples of properties modelling writes on a shared storage device, one without time (Fig. 2a), the other with time (Fig. 2b)

is deterministic, there exists only one such q'. We note $\text{Reach}(\sigma) = q_0 \text{ after } \sigma$. These definitions are valid both in the untimed and timed cases. For the timed case, we also allow to add an extra parameter to after and Reach, that represents an observation time. For $q \in Q$, $t \in \mathbb{R}_{\geq 0}$, and $\sigma \in \text{tw}(\Sigma)$, $q \text{ after } (\sigma, t) = (l, \nu + t - \text{time}(\text{obs}(\sigma, t)))$, where $(l, \nu) = q \text{ after } (\text{obs}(\sigma, t))$, and $\text{Reach}(\sigma, t) = q_0 \text{after}(\sigma, t)$. This allows to consider states of the semantics that are reached after the last action of the input word, by letting time elapse. We extend these definitions to languages: if L is a language, $q \text{ after } L = \bigcup_{\sigma \in L} q \text{ after } \sigma$ and $\text{Reach}(L) = q_0 \text{ after } L$.

Example 1 (Timed shared data storage). Consider the property φ_{t} described in Fig. 2b and representing writings on a shared data storage. A more detailed description of this property is given in Section 4.3. This property is similar to φ_{ex}, but a clock has been added to impose that writes should not occur before two time units have elapsed since the reception of the last `LockOff` event. Thus, the set of locations of φ_{t} is $L = \{l_0, l_1, l_2, l_3\}$, the initial location is l_0, the set of clocks is $X = \{x\}$, the alphabet is $\Sigma = \{\texttt{Auth}, \texttt{LockOn}, \texttt{LockOff}, \texttt{Write}\}$, the set of accepting locations is $G = \{l_1, l_2\}$, and the set of transitions contains for instance transitions $(l_0, \top, \texttt{Auth}, \emptyset, l_1)$, $(l_1, \top, \texttt{LockOn}, \emptyset, l_2)$, $(l_2, \top, \texttt{Auth}, \emptyset, l_2)$, $(l_3, \top, \texttt{LockOn}, \emptyset, l_3)$, where $\top$ is the guard that holds for every clock valuation.

Let $Q = L \times \mathbb{R}_{\geq 0}$ be the set of states of the semantics of φ_{t}, where the clock valuations are replaced by the value of the unique clock x. Then, $\text{Reach}((2, \texttt{Auth})) = (l_0, 0)\text{after}(2, \texttt{Auth}) = (l_1, 2)$, and, for example, $(l_2, 3) \text{ after } ((2, \texttt{LockOff}), 4) = (l_1, 2)$, because the clock is reset when the `LockOff` action occurs, and then $4 - 2 = 2$ time units remain to reach date 4.

3. Enforcement Monitoring of Untimed Properties

In this section, φ is a regular property defined by a complete and deterministic automaton $\mathcal{A}_\varphi = \langle Q, q_0, \Sigma, \rightarrow, F \rangle$. Recall the general scheme of an *enforcement mechanism* (EM) given in Fig. 1. We consider uncontrollable events in the set $\Sigma_{\text{u}} \subseteq \Sigma$. These events cannot be modified by an EM, i.e. they cannot be suppressed nor buffered, so they must be output by the EM whenever

they are received. Let us note $\Sigma_c = \Sigma \setminus \Sigma_u$ the set of controllable events, which can be modified by the EM. An EM can decide to buffer them to delay their emission, but it cannot suppress them (nevertheless, it can delay them endlessly, keeping their order unchanged).Thus, an EM may interleave controllable and uncontrollable events.

3.1. *Enforcement Functions and their Requirements*

We consider an alphabet of actions Σ. An enforcement function is a description of the input/output behaviour of an EM. It is function that modifies an execution, and that cannot remove events it has already output. Formally, we define *enforcement functions* as follows:

Definition 2 (Enforcement function). An *enforcement function* is a function from Σ^* to Σ^*, that is increasing on Σ^* (with respect to $\preccurlyeq$).

In the sequel, we define the requirements on an EM and express them on enforcement functions. As stated previously, the usual purpose of an EM is to ensure that the executions of a running system satisfy a property, thus its enforcement function has to be *sound*, meaning that its output always satisfies the property:

Definition 3 (Soundness). An enforcement function $E : \Sigma^* \to \Sigma^*$ is *sound* with respect to φ in an extension-closed set $S \subseteq \Sigma^*$ if $\forall \sigma \in S, E(\sigma) \models \varphi$.

Since there are some uncontrollable events that are only observable by the enforcement mechanism, receiving uncontrollable events could lead to the property not being satisfied by the output of the enforcement mechanism. Moreover, some uncontrollable sequences could lead to a state of the property that would be a non-accepting sink state, leading to the enforcement mechanism not being able to satisfy the property any further. Consequently, in Definition 3, soundness is not defined for all words in Σ^*, but in a subset S, since it might happen that it is impossible to ensure it from the initial state. In practice, for an EM to be effective, S needs to be extension-closed to ensure that the property is always satisfied once it has been. If S were not extension-closed, soundness would only mean that the property is sometimes satisfied (in particular, the identity function would be sound in φ).

The usual notion of *transparency* in enforcement monitoring (cf. (Schneider, 2000; Ligatti et al., 2009)) states that the output of an enforcement function is the longest prefix of the input satisfying the property. The name "transparency" stems from the fact that correct executions are left unchanged. However, because of uncontrollable events, events may be released in a different order from the one they are received. Therefore, transparency can not be ensured, and we define the weaker notion of *compliance*.

Definition 4 (Compliance). E is *compliant* with respect to Σ_u and Σ_c, noted $\mathrm{compliant}(E, \Sigma_u, \Sigma_c)$, if $\forall \sigma \in \Sigma^*, E(\sigma) \preccurlyeq_{\Sigma_c} \sigma \wedge E(\sigma) =_{\Sigma_u} \sigma \wedge \forall u \in \Sigma_u, E(\sigma).u \preccurlyeq E(\sigma.u)$.

Intuitively, compliance states that the EM does not change the order of the controllable events and emits uncontrollable events simultaneously with their reception, possibly followed by stored controllable events. When clear from the context, the partition is not mentioned: E is said to be compliant, and we note it $\mathrm{compliant}(E)$.

We say that a property φ is *enforceable* whenever there exists a compliant function that is sound with respect to φ.

In addition, an enforcement mechanism should be optimal in the sense that its output sequences should be maximal while preserving soundness and compliance. In the same way we defined soundness in an extension-closed set, we define optimality as follows:

Definition 5 (Optimality). An enforcement function $E : \Sigma^* \to \Sigma^*$ is *optimal* in an extension-closed set $S \subseteq \Sigma^*$ if:

$$\forall E' : \Sigma^* \to \Sigma^*, \forall \sigma \in S, \forall a \in \Sigma,$$
$$(\text{compliant}(E') \wedge E'(\sigma) = E(\sigma) \wedge |E'(\sigma.a)| > |E(\sigma.a)|) \implies (\exists \sigma_u \in \Sigma_{\text{u}}^*, E'(\sigma.a.\sigma_u) \not\models \varphi).$$

Intuitively, optimality states that if there exists a compliant enforcement function that outputs a longer word than an optimal enforcement function, then there must exist a sequence of uncontrollable events that would lead the output of that enforcement function to violate φ. This would imply that this enforcement function is not sound because of $\sigma.a.\sigma_u$. Since it is not always possible to satisfy the property from the beginning, this condition is restrained to an extension-closed subset of Σ^*, as in the definition of soundness (see Definition 3).

Example 2 (Untimed shared storage device). After Authentication, a user can write a value only if the storage is unlocked. (Un)locking is decided by another entity, meaning that it is not controllable by the user. Property φ_{ex} (see Fig. 2a) formalises the above requirement (where uncontrollable events are emphasised in italics). φ_{ex} is not enforceable if the uncontrollable alphabet is $\{\texttt{LockOn}, \texttt{LockOff}, \texttt{Auth}\}$ since reading the word $\texttt{LockOn}$ from q_0 leads to q_3, which is not an accepting state. Yet the existence of such a word does not imply that it is impossible to enforce φ_{ex} for some other input words. If word $\texttt{Auth}$ is read then state q_1 is reached, and from this state it is possible to enforce φ_{ex} by emitting $\texttt{Write}$ only when in state q_1.

3.2. *Synthesising Enforcement Functions*

Example 2 shows that some input words cannot be corrected by the EM because of uncontrollable events. Nevertheless, since the received events may lead to a state from which it is possible to ensure that φ will be satisfied (meaning that for any events received as input, the enforcement mechanism can output a sequence that satisfies φ), it would then be possible to define a subset of Σ^* in which an enforcement function would be sound.

To define this set, we first define the predicate Safe which, given a state q_i and a sequence of controllable events σ_0, indicates whether it is always possible to reach an accepting state from q_i with a prefix of σ_0, whatever uncontrollable events are received.

Definition 6 (Safe**).** Given a state $q_i \in Q$, and a word $\sigma_0 \in \Sigma_{\text{c}}^*$, we have:

$$\text{Safe}(q_i, \sigma_0) = \text{Safe}_{\text{int}}(q_i, \sigma_0, \emptyset), \text{where:}$$

$$\text{Safe}_{\text{int}}(q, \epsilon, P) = \quad (q \text{ after } \Sigma_{\text{u}}^*) \subseteq F, \text{and,}$$
$$\text{for } \sigma \in \Sigma^* \setminus \{\epsilon\},$$
$$\text{Safe}_{\text{int}}(q, \sigma, P) = \begin{cases} \{q' \in Q \mid (q', \sigma) \in P\} \subseteq F) & \text{if } (q, \sigma) \in P, \\ \forall u \in \Sigma_{\text{u}}, (\exists w \in \Sigma^*, w \preccurlyeq \sigma \wedge (q \text{ after } (u.w)) \in F \wedge & \\ \text{Safe}_{\text{int}}(q \text{ after } u.w, w^{-1}.\sigma, P \cup \{(q, \sigma)\})), & \text{otherwise,} \end{cases}$$

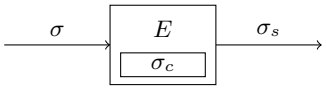


Figure 3: Enforcement function

for any $P \subseteq Q \times \Sigma^*$.

Intuitively, $\mathrm{Safe}(q, \sigma)$ indicates whether it is always possible to eventually reach an accepting state from q, and using only controllable events from σ, in the same order, whatever uncontrollable events are received. The third parameter of $\mathrm{Safe}_{\mathrm{int}}$, P, allows to define $\mathrm{Safe}(q, \sigma)$ properly, since there could be some loops in the inductive definition, meaning that the definition of $\mathrm{Safe}(q, \sigma)$ could depend on itself. P allows to avoid these loops, by simply ignoring recursive calls whenever a loop is detected. It is the set of arguments of all the calls to $\mathrm{Safe}_{\mathrm{int}}$ that are in the call stack when this call is made. $\mathrm{Safe}_{\mathrm{int}}$ is correctly defined whenever $\sigma = \epsilon$. If $\sigma \neq \epsilon$, then recursive calls are made, that could lead to a loop. A loop occurs when the computation of $\mathrm{Safe}_{\mathrm{int}}(q, \sigma, P)$ tries to compute $\mathrm{Safe}_{\mathrm{int}}(q, \sigma, P')$, for some subsets P and P' of $Q \times \Sigma^*$. If this happens, then on the call to $\mathrm{Safe}_{\mathrm{int}}(q, \sigma, P')$, $(q, \sigma) \in P'$, because the call to $\mathrm{Safe}_{\mathrm{int}}(q, \sigma, P')$ is a recursive call made by $\mathrm{Safe}_{\mathrm{int}}(q, \sigma, P)$, thus $P \cup \{(q, \sigma)\} \subseteq P'$. It follows that $\mathrm{Safe}_{\mathrm{int}}(q, \sigma, P')$ is well defined, meaning that $\mathrm{Safe}_{\mathrm{int}}(q, \sigma, P)$ is well defined too. Thus, since σ is finite, $\mathrm{Safe}_{\mathrm{int}}(q, \sigma, P)$ is correctly defined for all $q \in Q$ and $P \subseteq Q \times \Sigma^*$. This means that $\mathrm{Safe}(q_i, \sigma_0)$ is correctly defined for all $q_i \in Q$ and all $\sigma_0 \in \Sigma_c^*$.

For $q \in Q$ and $\sigma \in \Sigma_c^*$, we say that σ is safe to emit from q, or that q is safe with σ whenever $\mathrm{Safe}(q, \sigma)$ holds.

Now we define the functional behaviour of the enforcement mechanism.

Definition 7 (Functions store_φ, E_φ). † Function $\mathrm{store}_\varphi : \Sigma^* \to \Sigma^* \times \Sigma^*$ is defined as follows:

— $\mathrm{store}_\varphi(\epsilon) = (\epsilon, \epsilon)$;

— for $\sigma \in \Sigma^*$ and $a \in \Sigma$, let $(\sigma_s, \sigma_c) = \mathrm{store}_\varphi(\sigma)$, then:

$$\mathrm{store}_\varphi(\sigma.a) = \begin{cases} (\sigma_s.a.\sigma'_s, \sigma'_c) & \text{if } a \in \Sigma_{\mathrm{u}} \\ (\sigma_s.\sigma''_s, \sigma''_c) & \text{if } a \in \Sigma_{\mathrm{c}} \end{cases}, \text{ where:}$$

$$\kappa_\varphi(\sigma_1, \sigma_2) = \max_{\preccurlyeq}(\{w \preccurlyeq \sigma_2 \mid \sigma_1.w \models \varphi \wedge \mathrm{Safe}(\mathrm{Reach}(\sigma_1.w), w^{-1}.\sigma_2)\} \cup \{\epsilon\}),$$
$$\sigma'_s = \kappa_\varphi(\sigma_s.a, \sigma_c),$$
$$\sigma'_c = {\sigma'_s}^{-1}.\sigma_c,$$
$$\sigma''_s = \kappa_\varphi(\sigma_s, \sigma_c.a),$$
$$\sigma''_c = {\sigma''_s}^{-1}.(\sigma_c.a).$$

The enforcement function $\mathrm{E}_\varphi : \Sigma^* \to \Sigma^*$ is s.t. for $\sigma \in \Sigma^*$, $\mathrm{E}_\varphi(\sigma) = \Pi_1(\mathrm{store}_\varphi(\sigma))$.

Figure 3 gives a scheme of the behaviour of the enforcement function.

Intuitively, σ_s is the word that can be released as output, whereas σ_c is the buffer containing the events that are already read/received, but cannot be released as output yet because they lead to an

† E_φ and store_φ depend on Σ_{u} and Σ_{c}, but we did not write it in order to lighten the notations.

unsafe state from which it would be possible to violate the property reading only uncontrollable events. Upon receiving a new event a, the enforcement mechanism distinguishes two cases:

— If a belongs to Σ_{u}, then it is output, as required by compliance. Then, the longest prefix of σ_c that satisfies φ and is safe to be emitted is also output.
— If a is in Σ_c, then it is added to σ_c, and the longest prefix of this new buffer that satisfies φ and is safe is emitted, if it exists.

In both cases, κ_φ is used to compute the longest word that can be output, i.e. that satisfies φ and is safe. The first parameter of κ_φ corresponds to what has already been output, and the second parameter is the buffer of the enforcement mechanism, i.e. the sequence of controllable events that can be output by the enforcement mechanism.

As seen in Example 2, some properties are not enforceable, but receiving some events may lead to a state from which it is possible to enforce. Therefore, it is possible to define a set of words, called $\mathrm{Pre}(\varphi)$, such that E_φ is sound in $\mathrm{Pre}(\varphi)$, as stated in Proposition 2:

Definition 8 (Pre**).** $\mathrm{Pre}(\varphi) = \{w \in \Sigma^* \mid \exists \sigma \in \Sigma^*, w_{|\Sigma_{\mathrm{u}}} \preccurlyeq \sigma \wedge w =_{\Sigma_{\mathrm{u}}} \sigma \wedge \sigma \preccurlyeq_{\Sigma_{\mathrm{c}}} w \wedge \sigma \models \varphi \wedge \mathrm{Safe}(\mathrm{Reach}(\sigma), (\sigma_{|\Sigma_{\mathrm{c}}})^{-1}.w_{|\Sigma_{\mathrm{c}}})\}.\Sigma^*$.

Intuitively, $\mathrm{Pre}(\varphi)$ is the set of words for which E_φ would be sound. This set is extension-closed, as required by Definition 3. In E_φ, using Safe ensures that once the set $\mathrm{G} = \{w \preccurlyeq \sigma_2 \mid \sigma_1.w \models \varphi \wedge \mathrm{Safe}(\mathrm{Reach}(\sigma_1.w), w^{-1}.\sigma_2)\}$ is not empty, then it will never be afterwards, whatever events are received. Thus, $\mathrm{Pre}(\varphi)$ is the set of input words such that the output of E_φ would belong to G. Therefore, $\mathrm{Pre}(\varphi)$ considers words σ that are possible outputs of E_φ for input w. Since E_φ outputs only uncontrollable events until G is reached, $\mathrm{Pre}(\varphi)$ only considers σ as an extension of the uncontrollable events of w. The conditions $w =_{\Sigma_{\mathrm{u}}} \sigma$ and $\sigma \preccurlyeq_{\Sigma_{\mathrm{c}}} w$ are there to consider only words that can be output by a compliant enforcement mechanism. The last conditions $\sigma \models \varphi$ and $\mathrm{Safe}(\mathrm{Reach}(\sigma), (\sigma_{|\Sigma_{\mathrm{c}}})^{-1}.w_{|\Sigma_{\mathrm{c}}})$ ensure that G is not empty, meaning that the output of E_φ will satisfy φ from this point.

Example 3. Considering the property φ_{ex} as shown in Fig. 2a, with the uncontrollable alphabet $\Sigma_{\mathrm{u}} = \{\mathtt{Auth}, \mathtt{LockOff}, \mathtt{LockOn}\}$, $\mathrm{Pre}(\varphi_{\mathrm{ex}}) = \mathtt{Write}^*.\mathtt{Auth}.\Sigma^*$. Indeed, from the initial state q_0, if an uncontrollable event, say `LockOff`, is received, then q_3 is reached, which is a non-accepting sink state, and is thus not a safe state. In order to reach a safe state (i.e. q_1 or q_2), it is necessary to read `Auth`. Once `Auth` is read, q_1 is reached, and from there, all uncontrollable events lead to either q_1 or q_2. The same holds true from q_2. Thus, it is possible to stay in the accepting states q_1 and q_2, by delaying `Write` events when in q_2 until a `LockOff` event is received. Consequently, q_1 and q_2 are safe states, and thus $\mathrm{Pre}(\varphi_{\mathrm{ex}}) = \mathtt{Write}^*.\mathtt{Auth}.\Sigma^*$, since `Write` events can be buffered while in state q_0 until event `Auth` is received, leading to q_1.

E_φ as defined in Definition 7 is an enforcement function that is sound with respect to φ in $\mathrm{Pre}(\varphi)$, compliant with respect to Σ_{u} and Σ_{c}, and optimal in $\mathrm{Pre}(\varphi)$.

Proposition 1. E_φ as defined in Definition 7 is an enforcement function.

Sketch of proof. We have to show that for all σ and σ' in Σ^*, $\mathrm{E}_\varphi(\sigma) \preccurlyeq \mathrm{E}_\varphi(\sigma.\sigma')$. Following the definition of store_φ, this holds provided that $\sigma' \in \Sigma$ (i.e. σ' is a word of size 1). Since $\preccurlyeq$ is an order, it follows that the proposition holds for all $\sigma' \in \Sigma'$.

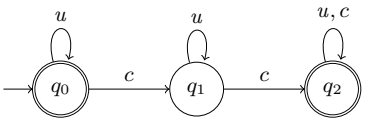


Figure 4: Property that can be enforced by blocking all controllable events c, thus outputting only the uncontrollable ones u.

Proposition 2. E_φ is sound with respect to φ in $\mathrm{Pre}(\varphi)$, as per Definition 3.

Sketch of proof. We have to show that if $\sigma \in \mathrm{Pre}(\varphi)$, then $\mathrm{E}_\varphi(\sigma) \models \varphi$. The proof is again made by induction on σ. In the induction step, considering $a \in \Sigma$, we distinguish three different cases:

1 $\sigma.a \notin \mathrm{Pre}(\varphi)$. Then the proposition holds.
2 $\sigma.a \in \mathrm{Pre}(\varphi)$, but $\sigma \notin \mathrm{Pre}(\varphi)$. Then the input reaches $\mathrm{Pre}(\varphi)$, and since it is extension-closed, all extensions of σ also are in $\mathrm{Pre}(\varphi)$, and we prove that the proposition holds considering the definition of $\mathrm{Pre}(\varphi)$.
3 $\sigma \in \mathrm{Pre}(\varphi)$ (and thus, $\sigma.a \in \mathrm{Pre}(\varphi)$ since it is extension-closed). Then, we prove that the proposition holds, based on the definition of store_φ, and more precisely on the definition of Safe, that ensures that there always exists a compliant output that satisfies φ.

Proposition 3. E_φ is compliant, as per Definition 4.

Sketch of proof. The proof is made by induction on the input $\sigma \in \Sigma^*$. Considering $\sigma \in \Sigma^*$ and $a \in \Sigma$, the proof is straightforward by considering the different values of $\mathrm{store}_\varphi(\sigma.a)$, $(\sigma.a)_{|\Sigma_\mathrm{u}}$, and $(\sigma.a)_{|\Sigma_\mathrm{c}}$ when $a \in \Sigma_\mathrm{c}$ and $a \in \Sigma_\mathrm{u}$.

Remark 1. Notice that for some properties, an enforcement function that would block all controllable events may still be sound and compliant. Consider for instance the property represented in Fig. 4, where c is a controllable event, and u an uncontrollable event. Then, outputting only the events u and buffering all the c events allows to stay in state q_0, which is accepting and safe for every word in c^*. This means that an enforcement mechanism which blocks all controllable events would be sound and compliant. Nevertheless, if two controllable events c are received, they can be output to reach state q_2, which is also accepting and safe for all possible sequences. Then it is possible to release more events. Therefore, an enforcement mechanism that would output two c events when they are received would be "better" than the first one blocking all of them, in the sense that its output would be longer.

For any input σ, $\mathrm{E}_\varphi(\sigma)$ is the longest possible word that ensures soundness and compliance, that is controllable events are blocked only when necessary. Thus, E_φ is also optimal in $\mathrm{Pre}(\varphi)$:

Proposition 4. E_φ is optimal in $\mathrm{Pre}(\varphi)$, as per Definition 5.

Sketch of proof. The proof is made by induction on the input $\sigma \in \Sigma^*$. Once $\sigma \in \mathrm{Pre}(\varphi)$, we know that $\mathrm{E}_\varphi(\sigma) \models \varphi$ since E_φ is sound in $\mathrm{Pre}(\varphi)$. E_φ is optimal because in store_φ, κ_φ provides the longest possible word. If a longer word were output, then either the output would not satisfy φ, or it would lead to a state that is not safe with the corresponding buffer, meaning that there

Table 1: Example of the evolution of $(\sigma_s, \sigma_c) = \mathrm{store}_{\varphi_{\mathrm{ex}}}(\sigma)$, with input `Auth.LockOn.Write.LockOff`

σ	σ_s	σ_c
ϵ	ϵ	ϵ
`Auth`	`Auth`	ϵ
`Auth.LockOn`	`Auth.LockOn`	ϵ
`Auth.LockOn.Write`	`Auth.LockOn`	`Write`
`Auth.LockOn.Write.LockOff`	`Auth.LockOn.LockOff.Write`	ϵ

would exist an uncontrollable word leading to a non-accepting state that would not be safe with the buffer. Then, the enforcement mechanism would have to output some controllable events from the buffer to reach an accepting state, but since the state is not safe with the buffer, there would exist again an uncontrollable word leading to a non-accepting state that is not safe with the updated buffer. By iterating, the buffer would become ϵ whereas the output of the enforcement mechanism would be leading to a non-accepting state. Therefore, outputting a longer word would mean that the function is not sound. This means that E_φ is optimal in $\mathrm{Pre}(\varphi)$, since it outputs the longest word that allows to be both sound and compliant.

Example 4. Consider property φ_{ex} (see Fig. 2a), we illustrate in Table 1 the enforcement mechanism by showing the evolution of σ_s and σ_c with input $\sigma =$ `Auth.LockOn.Write.LockOff`.

3.3. *Enforcement Monitors*

Enforcement monitors are operational descriptions of enforcement mechanisms. We give a representation of an enforcement mechanism as an input/output transition system. The input/output behaviour of the enforcement monitor is the same as the one of the enforcement function E_φ defined in Section 3.2. Enforcement monitors are purposed to ease the implementation of enforcement mechanisms, since this is a closer representation of a real enforcement mechanism.

Definition 9 (Enforcement monitor). An *enforcement monitor* $\mathcal{E}$ for φ is a transition system $\langle C^{\mathcal{E}}, c_0^{\mathcal{E}}, \Gamma^{\mathcal{E}}, \hookrightarrow_{\mathcal{E}} \rangle$ such that:

— $C^{\mathcal{E}} = Q \times \Sigma^*$ is the set of configurations.

— $c_0^{\mathcal{E}} = \langle q_0, \epsilon \rangle$ is the initial configuration.

— $\Gamma^{\mathcal{E}} = \Sigma^* \times \{\mathrm{dump}(.), \text{pass-uncont}(.), \text{store-cont}(.)\} \times \Sigma^*$ is the alphabet, where the first, second, and third members are an input sequence, an enforcement operation, and an output sequence, respectively.

— $\hookrightarrow_{\mathcal{E}} \subseteq C^{\mathcal{E}} \times \Gamma^{\mathcal{E}} \times C^{\mathcal{E}}$ is the transition relation, defined as the smallest relation obtained by applying the following rules in order (where $w/ \bowtie /w'$ stands for $(w, \bowtie, w') \in \Gamma^{\mathcal{E}}$):

  – **Dump:** $\langle q, \sigma.\sigma_c \rangle \xhookrightarrow{\epsilon/\,\mathrm{dump}(\sigma)/\sigma}_{\mathcal{E}} \langle q', \sigma_c \rangle$, where $\sigma \neq \epsilon$ and $q' = q$ after σ, with $q' \in F \wedge \mathrm{Safe}(q', \sigma_c)$,

  – **Pass-uncont:** $\langle q, \sigma_c \rangle \xhookrightarrow{a/\,\text{pass-uncont}(a)/a}_{\mathcal{E}} \langle q', \sigma_c \rangle$, with $a \in \Sigma_{\mathrm{u}}$ and $q' = q$ after a,

  – **Store-cont:** $\langle q, \sigma_c \rangle \xhookrightarrow{a/\,\text{store-cont}(a)/\epsilon}_{\mathcal{E}} \langle q, \sigma_c.a \rangle$.

In $\mathcal{E}$, a configuration $c = \langle q, \sigma \rangle$ represents the current state of the enforcement mechanism. The state q is the one reached so far in $\mathcal{A}_\varphi$ with the output of the monitor. The word of controllable events σ represents the buffer of the monitor, i.e. the controllable events of the input that it has not output yet. Rule **dump** outputs a prefix of the word in memory (the buffer) whenever it is possible to ensure soundness afterwards. Rule **pass-uncont** releases an uncontrollable event as soon as it is received. Rule **store-cont** simply adds a controllable event at the end of the buffer. Compared to Section 3.2, the second member of the configuration represents buffer σ_c in the definition of store_φ, whereas σ_s is here represented by state q which is the first member of the configuration, such that $q = \mathrm{Reach}(\sigma_s)$.

Proposition 5. The output of the enforcement monitor $\mathcal{E}$ for input σ is $\mathrm{E}_\varphi(\sigma)$.

In Proposition 5, the output of the enforcement monitor is the concatenation of all the outputs of the word labelling the path followed when reading σ. A more formal definition is given in the proof of this proposition, in appendix A.1.

Sketch of proof. By induction on the input $\sigma \in \Sigma^*$, we just consider the rules that can be applied when receiving a new event. If the event is controllable, then rule $\mathrm{store\text{-}cont}()$ can be applied, possibly followed by rule $\mathrm{dump}()$. Otherwise, the rule $\mathrm{pass\text{-}uncont}()$ can be applied, again possibly followed by rule $\mathrm{dump}()$. Since rule $\mathrm{dump}()$ applies only when reaching a safe accepting state with the controllable events in the buffer, it corresponds exactly to the computation of κ_φ in the definition of store_φ, and consequently the outputs of $\mathcal{E}$ and E_φ are the same.

Remark 2. Enforcement monitors in Definition 9 resemble the ones in (Falcone et al., 2011), except that we explicitly keep the memory as part of the configuration and get uniform definitions in the untimed and timed settings (see Section 4). Hence, enforcement monitors in Definition 9 can also equivalently be defined using a finite-state machine as in (Falcone et al., 2011).

4. Enforcement Monitoring of Timed Properties

In this section, we extend the framework presented in Section 3 to enforce timed properties. Enforcement mechanisms and their properties need to be redefined to fit with timed properties. Enforcement functions need an extra parameter representing the date at which the output is observed. Soundness needed to be weaken so that, at any time instant, the property is allowed not to hold, provided that it will hold in the future.

Considering uncontrollable events with timed properties raises several difficulties. First, as is the case in the untimed case, the order of events might be modified. Thus, previous definitions of transparency ((Pinisetty et al., 2012)), stating that the output of an enforcement function will eventually be a delayed prefix of the input, can not be used in this situation. Moreover, when delaying some events to have the property satisfied in the future, one must consider the fact that some uncontrollable events could occur at any moment (and cannot be delayed). Finally, some properties become not enforceable because of uncontrollable events, meaning that for these properties it is impossible to obtain sound enforcement mechanisms, as shown in Example 5.

In this section, φ is a timed property defined by a timed automaton $\mathcal{A}_\varphi = \langle L, l_0, X, \Sigma, \Delta, G \rangle$ with semantics $[\![\mathcal{A}_\varphi]\!] = \langle Q, q_0, \Gamma, \rightarrow, F_G \rangle$.

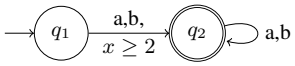


Figure 5: A timed property enforceable only if $\Sigma_{\mathrm{u}} = \emptyset$.

Example 5 (Non enforceable property). Consider the property defined by the automaton in Fig. 5 with $\Sigma = \{a, b\}$. If all actions are controllable ($\Sigma_{\mathrm{u}} = \emptyset$), the property is enforceable because an enforcement mechanism just needs to delay events until clock x exceeds 2. Otherwise, the property is not enforceable. For instance, if $\Sigma_{\mathrm{u}} = \{a\}$, word $(1, a)$ cannot be corrected.

4.1. *Enforcement Functions and their Properties*

Enforcement function take a timed word and the current time as input, and output a timed word:

Definition 10 (Enforcement Function). Given an alphabet of actions Σ, an *enforcement function* is a function $E : \mathrm{tw}(\Sigma) \times \mathbb{R}_{\geq 0} \to \mathrm{tw}(\Sigma)$ such that:

$$\forall \sigma \in \mathrm{tw}(\Sigma), \forall t \in \mathbb{R}_{\geq 0}, \forall t' \geq t,$$
$$(E(\sigma, t) \preccurlyeq E(\sigma, t')) \wedge (\sigma.(t, a) \in \mathrm{tw}(\Sigma) \implies E(\sigma, t) \preccurlyeq E(\sigma.(t, a), t)) .$$

The requirements in Definition 10 model physical constraints: an enforcement function can not remove something it has already output. The two conditions correspond to letting time elapse and reading a new event, respectively. In both cases, the new output must be an extension of what has been output so far.

Soundness states that the output of an enforcement function eventually satisfies the property:

Definition 11 (Soundness). An enforcement function E is *sound* with respect to φ in a time-extension-closed set $S \subseteq \mathrm{tw}(\Sigma) \times \mathbb{R}_{\geq 0}$ if $\forall (\sigma, t) \in S, \exists t' \geq t, \forall t'' \geq t', E(\sigma, t'') \models \varphi$.

An enforcement function is sound in a time-extension-closed set S if for any (σ, t) in S, the value of the enforcement function with input σ from date t satisfies the property in the future. As in the untimed setting, soundness is not defined for all words in $\mathrm{tw}(\Sigma)$, but in a set of words, this time associated with dates. This is again because the enforcement mechanism might not be able to ensure soundness from the beginning, because of bad uncontrollable sequences. Moreover, in the definition of soundness, the set S needs to be time-extension-closed in order to ensure that the property remains satisfied once the enforcement mechanism starts to operate.

Remark 3. Note that soundness could have been defined in the same way as in the untimed setting, but with this strong definition, where the output of the enforcement mechanism must always satisfy the property, less properties could be enforced. Weakening soundness allows to enforce more properties, and to let enforcement mechanisms produce longer outputs.

Compliance states that uncontrollable events should be emitted instantaneously upon reception, and that controllable events can be delayed, but their order must remain unchanged:

Definition 12 (Compliance). Given an enforcement function E defined on an alphabet Σ, we say that E is *compliant* with respect to Σ_{u} and Σ_{c}, noted $\mathrm{compliant}(E, \Sigma_{\mathrm{u}}, \Sigma_{\mathrm{c}})$, if $\forall \sigma \in \mathrm{tw}(\Sigma), \forall t \in \mathbb{R}_{\geq 0}, \sigma \preccurlyeq_{d_{\Sigma_{\mathrm{c}}}} E(\sigma, t) \wedge E(\sigma, t) =_{\Sigma_{\mathrm{u}}} \mathrm{obs}(\sigma, t) \wedge \forall u \in \Sigma_{\mathrm{u}}, E(\sigma, t).(t, u) \preccurlyeq E(\sigma.(t, u), t)$.

Compliance is similar to the one in the untimed setting except that the controllable events can be delayed. However, their order must not be modified by the enforcement mechanism, that is, when considering the projections on controllable events, the output should be a delayed prefix of the input. Regarding uncontrollable events, any uncontrollable event is released immediately when received, that is, when considering the projections on uncontrollable events, the output should be equal to the input.

We say that a property is *enforceable* whenever there exists a sound and compliant enforcement function for this property.

For a compliant enforcement function $E : \mathrm{tw}(\Sigma) \times \mathbb{R}_{\geq 0} \to \mathrm{tw}(\Sigma)$, and a timed word $\sigma \in \mathrm{tw}(\Sigma)$, we note $E(\sigma)$ the value of E with input σ at infinite time (i.e. when it has stabilised). More formally, $E(\sigma) = E(\sigma, t)$, where $t \in \mathbb{R}_{\geq 0}$ is such that for all $t' \geq t$, $E(\sigma, t') = E(\sigma, t)$. Since σ is finite, and E is compliant, $E(\sigma)$ is finite, thus such a t must exist.

As in the untimed setting, we define a notion of *optimality* in a set:

Definition 13 (Optimality). We say that an enforcement function $E : \mathrm{tw}(\Sigma) \times \mathbb{R}_{\geq 0} \to \mathrm{tw}(\Sigma)$ that is compliant with respect to Σ_c and Σ_u is *optimal* in a time-extension-closed set $S \subseteq \mathrm{tw}(\Sigma) \times \mathbb{R}_{\geq 0}$ if for all enforcement function $E' : \mathrm{tw}(\Sigma) \times \mathbb{R}_{\geq 0} \to \mathrm{tw}(\Sigma)$, for all $\sigma \in \mathrm{tw}(\Sigma)$, for all (t, a) such that $\sigma.(t, a) \in \mathrm{tw}(\Sigma)$,

$$\begin{aligned}&\mathrm{compliant}(E', \Sigma_\mathrm{u}, \Sigma_\mathrm{c}) \wedge (\sigma, t) \in S \wedge (E'(\sigma, t) = E(\sigma, t) \wedge E'(\sigma.(t, a)) \prec_d E(\sigma.(t, a))\\ &\quad \implies (\exists \sigma_u \in \mathrm{tw}(\Sigma_\mathrm{u}), E'(\sigma.(t, a).\sigma_u) \not\models \varphi).\end{aligned}$$

Optimality states that outputting a lower word (with respect to $\preccurlyeq_d$) than the output of an optimal enforcement function leads to either compliance or soundness not being guaranteed. This holds from the point where the input begins to belong to the set in which the function is optimal, and since it is time-extension-closed, the input will belong to this set ever after. In Definition 13, E is an optimal enforcement function, and E' is another compliant enforcement function, that we consider having a smaller output (with respect to $\preccurlyeq_d$) than E for some input word $\sigma.(t, a)$. Then, since E is optimal, E' is not sound, because there exists a word of uncontrollable events such that the output of E' after receiving it eventually violates φ.

An enforcement mechanism that delays events has to buffer them until it can output them. Being able to enforce φ depends on the possibility of computing a timed word with the events of the buffer, even when receiving some uncontrollable events, that leads to an accepting state from the current one. Thus, we define the predicate Safe which, given a state of the semantics of $\mathcal{A}_\varphi$, and a sequence of controllable events corresponding to the buffer, indicates if it is possible to compute a timed word leading to an accepting state, whatever uncontrollable events are received:

Definition 14 (Safe**).** Given a state q_i of $[\![\mathcal{A}_\varphi]\!]$, and a sequence of controllable events $\sigma_0 \in \Sigma_\mathrm{c}^*$, we define the predicate $\mathrm{Safe}(q_i, \sigma_0)$ as:

$$\mathrm{Safe}(q_i, \sigma_0) = \mathrm{Safe}_{\mathrm{int}}(q_i, \sigma_0, \emptyset), \text{where:}$$

$$\mathrm{Safe}_{\mathrm{int}}(q,\epsilon,P) = \quad (q \text{ after } \mathrm{tw}(\Sigma_{\mathrm{u}})) \subseteq F_G$$
and, for $\sigma \in \Sigma_{\mathrm{c}}^* \setminus \{\epsilon\}$,

$$\mathrm{Safe}_{\mathrm{int}}(q,\sigma,P) = \begin{cases} \{q' \in Q \mid (q',\sigma) \in P\} \subseteq F_G & \text{if } (q,\sigma) \in P \\ \forall u \in \Sigma_{\mathrm{u}}, \exists w \in \mathrm{tw}(\Sigma), \Pi_\Sigma(w) \preccurlyeq \sigma \wedge \forall t \in \mathbb{R}_{\geq 0}, & \\ ((q \text{ after } ((0,u).w,t)) \in F_G \wedge & \text{otherwise} \\ \mathrm{Safe}_{\mathrm{int}}(q \text{ after } ((0,u).w,t), \Pi_\Sigma(\mathrm{obs}(w,t))^{-1}.\sigma, P \cup \{(q,\sigma)\})) & \end{cases}$$

Intuitively, $\mathrm{Safe}(q_i,\sigma_0)$ indicates whether it is always possible to reach an accepting state from q_i with controllable events taken from σ_0, in the same order, whatever uncontrollable events are received. The last parameter of $\mathrm{Safe}_{\mathrm{int}}$, P, as in the untimed case, is used to avoid loops, that would make Safe undefined for some parameters (since they could depend on themselves). If $\sigma = \epsilon$, $\mathrm{Safe}_{\mathrm{int}}(q,\epsilon,P)$ is defined for all $q \in Q$, and for all $P \subseteq Q \times \Sigma_c^*$. If $\sigma \neq \epsilon$, then loops could occur if $w = \epsilon$ and thus if a state reached after an uncontrollable word leads to some state already reached. Therefore, this state would be in P, associated with the same σ, since $w = \epsilon$, meaning that the second parameter does not change in the recursive call. Thus, since this pair (q,σ) would be in P, the computation would end, its value depending on the acceptance of all the states reached in the loop. Thus, Safe is defined for all $q \in Q$ and for all $\sigma \in \Sigma_{\mathrm{c}}^*$. Moreover, considering regions instead of symbolic states, the size of the set of states is bounded by the number of regions (see (Alur and Dill, 1992)), and since the second parameter is always a prefix of σ in recursive calls, the size of P would be bounded by the length of σ times the number of regions. Since the size of P strictly increases in recursive calls, and having the pair composed of the first two parameters in P is sufficient to terminate the computation, the computation of $\mathrm{Safe}(q,\sigma)$ must terminate for all $q \in Q$ and for all $\sigma \in \mathrm{tw}(\Sigma)$.

Unlike in the untimed case, some delay between two consecutive events may be necessary to satisfy φ, thus it is possible for an uncontrollable event to happen while waiting for the duration of the delay. If this happens, the enforcement mechanism needs to compute again the dates for the events it has not output yet in order to reach F_G if possible. Safe is used to ensure that F_G is always reachable with the events that have not been output yet even if some uncontrollable events occur. We use Safe to define an enforcement function for φ, denoted as E_φ:

Definition 15 (store_φ, E_φ). Let store_φ be the function : $\mathrm{tw}(\Sigma) \times \mathbb{R}_{\geq 0} \to \mathrm{tw}(\Sigma) \times \mathrm{tw}(\Sigma_{\mathrm{c}}) \times \Sigma_{\mathrm{c}}^*$ defined inductively by:
$\forall t \in \mathbb{R}_{\geq 0}, \mathrm{store}_\varphi(\epsilon,t) = (\epsilon,\epsilon,\epsilon)$, and
for $\sigma \in \mathrm{tw}(\Sigma)$, (t',a) s.t. $\sigma.(t',a) \in \mathrm{tw}(\Sigma)$, and $t \geq t'$, if $(\sigma_s,\sigma_b,\sigma_c) = \mathrm{store}_\varphi(\sigma,t')$, then

$$\mathrm{store}_\varphi(\sigma.(t',a),t) = \begin{cases} (\sigma_s.(t',a).\,\mathrm{obs}(\sigma_b',t),\sigma_b',\sigma_c') & \text{if } a \in \Sigma_{\mathrm{u}} \\ (\sigma_s.\,\mathrm{obs}(\sigma_b'',t),\sigma_b'',\sigma_c'') & \text{if } a \in \Sigma_{\mathrm{c}} \end{cases}$$

with:

$$
\begin{aligned}
\mathrm{G}(q,\sigma_1) &= \{w \in \mathrm{tw}(\Sigma) \mid \Pi_\Sigma(w) \preccurlyeq \sigma_1 \wedge (q \text{ after } w) \in F_G \wedge \\
&\qquad \forall t'' \in \mathbb{R}_{\geq 0}, \mathrm{Safe}(q \text{ after } (w,t''), \Pi_\Sigma(\mathrm{obs}(w,t''))^{-1}.\sigma_1)\}, \\
\kappa_\varphi(q,\sigma_1) &= \min_{\text{lex}}(\max_{\preccurlyeq}(\mathrm{G}(q,\sigma_1) \cup \{\epsilon\})) \\
\mathit{buffer}_c &= \Pi_\Sigma(\mathrm{nobs}(\sigma_b,t')).\sigma_c \\
t_1 &= \begin{array}{l}\min(\{t'' \in \mathbb{R}_{\geq 0} \mid t'' \geq t' \wedge \\ \mathrm{G}(\mathrm{Reach}(\sigma_s.(t',a),t''), \mathit{buffer}_c) \neq \emptyset\} \cup \{+\infty\}),\end{array} \\
\sigma'_b &= \kappa_\varphi(\mathrm{Reach}(\sigma_s.(t',a), \min(\{t,t_1\})), \mathit{buffer}_c) +_{\mathrm{t}} \min(\{t,t_1\}), \\
\sigma'_c &= \Pi_\Sigma(\sigma'_b)^{-1}.\mathit{buffer}_c, \\
t_2 &= \begin{array}{l}\min(\{t'' \in \mathbb{R}_{\geq 0} \mid t'' \geq t' \wedge \\ \mathrm{G}(\mathrm{Reach}(\sigma_s,t''), \mathit{buffer}_c.a) \neq \emptyset\} \cup \{+\infty\}),\end{array} \\
\sigma''_b &= \kappa_\varphi(\mathrm{Reach}(\sigma_s, \min(\{t,t_2\})), \mathit{buffer}_c.a) +_{\mathrm{t}} \min(\{t,t_2\}), \\
\sigma''_c &= \Pi_\Sigma(\sigma''_b)^{-1}.(\mathit{buffer}_c.a).
\end{aligned}
$$

For $\sigma \in \mathrm{tw}(\Sigma)$, and $t \in \mathbb{R}_{\geq 0}$, we define $\mathrm{E}_\varphi(\sigma,t) = (\Pi_1(\mathrm{store}_\varphi(\mathrm{obs}(\sigma,t),t)))$.

In the definition of store_φ, $\mathrm{G}(q,\sigma_1)$ is the set of prefixes of σ_1 that can be emitted safely from q. store_φ takes a timed word σ as input, and a date t, and outputs three words: σ_s, σ_b, and σ_c. σ_s is the output of the enforcement monitor at time t. σ_b is the timed word, composed of controllable events, that is to be output after the last event of the input, if no uncontrollable event is received. σ_c is the untimed word composed of the remaining controllable actions of the buffer. When time elapses, after the last event of the input, σ_s is modified to output the events of σ_b when the dates are reached. Since letting time elapse can disable some transitions, it is possible to reach a safe state without emitting any event, and thus σ_b can change at this moment, changing from ϵ to a safe word. This change of σ_b when letting time elapse can only happen once, since G will not be empty anymore once it has become non-empty. t_1 and t_2 are used for this purpose, they both represent the time at which G becomes non-empty, if $a \in \Sigma_{\mathrm{u}}$ or $a \in \Sigma_{\mathrm{c}}$ respectively. Words are thus calculated from this point whenever G has become non-empty, to ensure that what has already been output is not modified. If G is still empty, then $\min(\{t,t_1\})$ (or $\min(\{t,t_2\})$, depending on whether $a \in \Sigma_{\mathrm{c}}$ or $a \in \Sigma_{\mathrm{u}}$) equals to t, meaning that $\sigma_b = \epsilon$. Most of the time, t_1, or t_2 is equal to t', it is not the case only when G was still empty at time t', but if G was not empty at date t', then t_1 (or t_2) is equal to t'. σ_c contains the controllable actions of the input that have not been output and do not belong to σ_b. It is used to compute the new value of σ_b when possible. When receiving a new event in the input, it is appended to σ_s if it is an uncontrollable event, or the action is appended to the buffer if it is a controllable one. Then, σ_b is computed again, from the new state reached if it was an uncontrollable event, or with the new buffer if it was controllable. Note that t_1 and t_2 may not exist if strict guards exist. In this case, one should consider the infimum instead of the minimum, and consider $t'' \in \mathbb{R}_{\geq 0} \setminus \{0\}$ in the definition of G, so that the behaviour would be as desired.

As mentioned previously, an enforcement mechanism may not be sound from the beginning of an execution, but some uncontrollable events may lead to a state from which it becomes possible

to be sound. Whenever σ_b is safe, then it will always be, meaning that the output of E_φ will eventually reach a state in F_G, i.e. it will eventually satisfy φ. Thus, E_φ eventually satisfies φ as soon as σ_b is safe from the state reached so far. This leads to the definition of $\mathrm{Pre}(\varphi, t)$, which is the set of timed words for which E_φ ensures soundness at time t. For $\sigma \in \mathrm{tw}(\Sigma)$, if $(\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t)$, then σ is in $\mathrm{Pre}(\varphi, t)$ if and only if σ_b is safe from state $\mathrm{Reach}(\sigma_s)$. If σ_b is safe from $\mathrm{Reach}(\sigma_s)$, then the output of store_φ will be safe with its buffer for any extension of σ as input, thus $\mathrm{Pre}(\varphi, t)$ is extension-closed. To define $\mathrm{Pre}(\varphi, t)$, we first define $\mathrm{E}_1(\sigma, t)$, which is the set of words that could be output by E_φ with input σ at date t in the case where only uncontrollable events were output previously (E_φ would output the minimum with respect to $\preccurlyeq_d$). Then, $\mathrm{Pre}(\varphi, t)$ is used to define $\mathrm{Pre}(\varphi)$, which is the set in which E_φ is sound:

Definition 16. $\mathrm{Pre}(\varphi)$

$$\mathrm{Pre}(\varphi) = \{(\sigma, t) \mid \sigma \in \mathrm{Pre}(\varphi, t)\},$$

where, for $\sigma \in \mathrm{tw}(\Sigma)$ and $t \in \mathbb{R}_{\geq 0}$:

$$\begin{aligned}
\mathrm{E}_1(\sigma, t) = \; & \{w \in \mathrm{tw}(\Sigma) \mid \sigma_{|\Sigma_\mathrm{u}} \preccurlyeq w \wedge \sigma =_{\Sigma_\mathrm{u}} w \wedge \sigma \preccurlyeq_{d\,\Sigma_\mathrm{c}} w \wedge \\
& (w_{|\Sigma_\mathrm{c}} \neq \epsilon \implies \mathrm{date}(w_{|\Sigma_\mathrm{c}}(1)) \geq t) \wedge \mathrm{Reach}(w) \in F_G \wedge \forall t' \geq t, \\
& \mathrm{Safe}(\mathrm{Reach}(w, t'), \Pi_\Sigma(\mathrm{obs}(w_{|\Sigma_\mathrm{c}}, t'))^{-1}.\Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}}))\} \\
\mathrm{Pre}(\varphi, t) = \; & \{\sigma \in \mathrm{tw}(\Sigma) \mid \exists t' \leq t, \mathrm{E}_1(\mathrm{obs}(\sigma, t'), t') \neq \emptyset\}
\end{aligned}$$

Note that $\mathrm{Pre}(\varphi)$ is time-extension-closed, meaning that once E_φ is sound, it will always be in the future.

In Definition 16, $\mathrm{E}_1(\sigma, t)$ is the set of words that can be output safely by our enforcement function, with input σ, and after time t. Considering that the output of our enforcement function was only the uncontrollable events so far, if $\mathrm{E}_1(\sigma, t)$ is not empty, this means that the enforcement function becomes sound with input σ from time t, since there is a word that is safe to emit. Thus, $\mathrm{Pre}(\varphi, t)$ is the set of inputs for which E_φ is sound after date t, and then E_φ is sound for any input in $\mathrm{Pre}(\varphi)$ after its associated date.

Proposition 6. E_φ as defined in Definition 15 is an enforcement function, as per Definition 10.

Sketch of proof. We have to show that for all $\sigma \in \mathrm{tw}(\Sigma)$, for all $t \in \mathbb{R}_{\geq 0}$ and $t' \geq t$, $\mathrm{E}_\varphi(\sigma, t) \preccurlyeq \mathrm{E}_\varphi(\sigma, t')$, and for all (t'', a) such that $\sigma.(t'', a) \in \mathrm{tw}(\Sigma)$, if $t \leq t''$, then $\mathrm{E}_\varphi(\sigma, t) \preccurlyeq \mathrm{E}_\varphi(\sigma.(t'', a), t')$. To prove this, we first show that $\mathrm{E}_\varphi(\sigma, t) \preccurlyeq \mathrm{E}_\varphi(\sigma, t')$. Considering (t'', a) such that $\sigma.(t'', a) \in \mathrm{tw}(\Sigma)$, we distinguish different cases according to the values of t'' compared to t and t':

Case 1: $t'' \leq t$. Then, in the definition of store_φ, t_1 (or t_2, if a is controllable) has the same value in $\mathrm{store}_\varphi(\sigma, t)$ and $\mathrm{store}_\varphi(\sigma.(t'', a), t')$. Then, comparing t to t_1 (or t_2), either $\mathrm{store}_\varphi(\sigma.(t'', a), t) = \epsilon$ if $t < t_1$, and then $\mathrm{store}_\varphi(\sigma.(t'', a), t) \preccurlyeq \mathrm{store}_\varphi(\sigma.(t'', a), t')$, or $t \geq t_1$, and then there exists σ_s and σ_b such that $\mathrm{store}_\varphi(\sigma.(t'', a), t) = \sigma_s.\,\mathrm{obs}(\sigma_b, t)$ and $\mathrm{store}_\varphi(\sigma.(t'', a), t') = \sigma_s.\,\mathrm{obs}(\sigma_b, t')$, meaning that $\mathrm{store}_\varphi(\sigma.(t'', a), t) \preccurlyeq \mathrm{store}_\varphi(\sigma.(t'', a), t')$.

Case 2: $t'' \geq t'$. Then the proposition holds because in the definition of E_φ, only the observation of the input word at the given time is considered, meaning that $\mathrm{E}_\varphi(\sigma.(t'', a), t) = \mathrm{E}_\varphi(\sigma, t)$ and $\mathrm{E}_\varphi(\sigma.(t'', a), t') = \mathrm{E}_\varphi(\sigma, t')$. By induction hypothesis, the proposition thus holds.

Case 3: $t < t'' < t'$. Then, $\mathrm{E}_\varphi(\sigma.(t'',a),t) = \mathrm{E}_\varphi(\sigma,t)$, and $\mathrm{E}_\varphi(\sigma.(t'',a),t') = \Pi_1(\mathrm{store}_\varphi(\sigma.(t'',a),t'))$, meaning that, looking at the definition of store_φ, $\mathrm{E}_\varphi(\sigma.(t'',a),t) \preccurlyeq \mathrm{E}_\varphi(\sigma.(t'',a),t')$.

Thus, $\mathrm{E}_\varphi(\sigma,t) \preccurlyeq \mathrm{E}_\varphi(\sigma,t')$. Then, what remains to show is that if $t \leq t''$, then $\mathrm{E}_\varphi(\sigma,t) \preccurlyeq \mathrm{E}_\varphi(\sigma.(t'',a),t')$. Then, there are only two cases:

Case 1: $t' < t''$. Then, $\mathrm{obs}(\sigma.(t'',a),t') = \mathrm{obs}(\sigma,t')$, thus $\mathrm{E}_\varphi(\sigma.(t'',a),t') = \mathrm{E}_\varphi(\sigma,t')$, thus $\mathrm{E}_\varphi(\sigma,t) \preccurlyeq \mathrm{E}_\varphi(\sigma,t') = \mathrm{E}_\varphi(\sigma.(t'',a),t')$.

Case 2: $t' \geq t''$. Then, $\mathrm{obs}(\sigma.(t'',a),t') = \sigma.(t'',a)$, and considering the definition of $\mathrm{store}_\varphi(\sigma.(t'',a),t')$, $\mathrm{E}_\varphi(\sigma,t'') \preccurlyeq \mathrm{store}_\varphi(\sigma.(t'',a),t') = \mathrm{E}_\varphi(\sigma.(t'',a),t')$. Thus, $\mathrm{E}_\varphi(\sigma,t) \preccurlyeq \mathrm{E}_\varphi(\sigma,t'') \preccurlyeq \mathrm{E}_\varphi(\sigma.(t'',a),t'') \preccurlyeq \mathrm{E}_\varphi(\sigma.(t'',a),t')$.

Proposition 7. E_φ is sound with respect to φ in $\mathrm{Pre}(\varphi)$ as per Definition 11.

Sketch of proof. As in the untimed setting, the proof is made by induction on the input $\sigma \in \mathrm{tw}(\Sigma)$. Similarly to the untimed setting, considering $\sigma \in \mathrm{tw}(\Sigma)$, $t \in \mathbb{R}_{\geq 0}$, and (t',a) such that $\sigma.(t',a) \in \mathrm{tw}(\Sigma)$, there are three possibilities:

1 $(\sigma.(t',a),t) \notin \mathrm{Pre}(\varphi)$. Then, the proposition holds.
2 $(\sigma.(t',a),t) \in \mathrm{Pre}(\varphi)$, but $(\sigma,t') \notin \mathrm{Pre}(\varphi)$. Then, this is when the input reaches $\mathrm{Pre}(\varphi)$. Considering the definition of $\mathrm{Pre}(\varphi)$, we then prove that it is possible to emit a word with the controllable events seen so far, leading to an accepting state which is safe with the remaining controllable events.
3 $(\sigma,t') \in \mathrm{Pre}(\varphi)$ (and thus $(\sigma.(t',a),t)$ too). Then, we prove that there exists a controllable word made with the events which have not been output yet leading to an accepting state that is safe with the remaining controllable events, but this time considering the definition of Safe.

Proposition 8. E_φ is compliant, as per Definition 12.

Sketch of proof. As in the untimed setting, the proof is made by induction on the input σ, considering the different cases where the new event is controllable or uncontrollable. The only difference with the untimed setting is that one should consider dates on top of actions.

Proposition 9. E_φ is optimal in $\mathrm{Pre}(\varphi)$ as per Definition 13.

Sketch of proof. This proof is made by induction on the input σ. Whenever $\sigma \in \mathrm{Pre}(\varphi)$, since E_φ is sound in $\mathrm{Pre}(\varphi)$, then $\mathrm{E}_\varphi(\sigma)$ is the minimal word (with respect to $\preccurlyeq_d$) that satisfies φ and is safe to output. It is minimal because in the definition of store_φ, κ_φ returns the longest word with lower delays (for lexicographic order), which corresponds to the minimum with respect to $\preccurlyeq_d$. Thus, outputting a lower word (with respect to $\preccurlyeq_d$) would lead either to not satisfy the property, or to not being safe. As in the untimed setting, not being safe means not being sound. Thus, E_φ is optimal in $\mathrm{Pre}(\varphi)$, since it outputs the minimal word with respect to $\preccurlyeq_d$ that allows to be sound and compliant.

4.2. *Enforcement Monitors*

As in the untimed setting, we define an operational description of an enforcement mechanism whose output is exactly the output of E_φ, as defined in Definition 15.

Definition 17. An *enforcement monitor* $\mathcal{E}$ for φ is a transition system $\langle C^{\mathcal{E}}, c_0^{\mathcal{E}}, \Gamma^{\mathcal{E}}, \hookrightarrow_{\mathcal{E}} \rangle$ s.t.:

— $C^{\mathcal{E}} = \mathrm{tw}(\Sigma) \times \Sigma_c^* \times Q \times \mathbb{R}_{\geq 0} \times \{\top, \bot\}$ is the set of configurations.

— $c_0^{\mathcal{E}} = \langle \epsilon, \epsilon, q_0, 0, \bot \rangle \in C^{\mathcal{E}}$ is the initial configuration.

— $\Gamma^{\mathcal{E}} = ((\mathbb{R}_{\geq 0} \times \Sigma) \cup \{\epsilon\}) \times Op \times ((\mathbb{R}_{\geq 0} \times \Sigma) \cup \{\epsilon\})$ is the alphabet, composed of an optional input, an operation and an optional output.
The set of operations is $\{\mathrm{compute}(.), \mathrm{dump}(.), \text{pass-uncont}(.), \text{store-cont}(.), \mathrm{delay}(.)\}$.
Whenever $(\sigma, \bowtie, \sigma') \in \Gamma^{\mathcal{E}}$, it will be noted $\sigma / \bowtie / \sigma'$.

— $\hookrightarrow_{\mathcal{E}}$ is the transition relation defined as the smallest relation obtained by applying the following rules given by their priority order:

– **Compute:** $\langle \epsilon, \sigma_c, q, t, \bot \rangle \xhookrightarrow{\epsilon / \text{ compute}() / \epsilon}_{\mathcal{E}} \langle \sigma_b', \sigma_c', q, t, \top \rangle$, if $\mathrm{G}(q, \sigma_c) \neq \emptyset$, with $\sigma_b' = \kappa_\varphi(q, \sigma_c) +_{\mathrm{t}} t$, and $\sigma_c' = \Pi_\Sigma(\sigma_b')^{-1}.\sigma_c$,

– **Dump:** $\langle (t_b, a).\sigma_b, \sigma_c, q, t_b, \top \rangle \xhookrightarrow{\epsilon / \text{ dump}(t_b,a) / (t_b,a)}_{\mathcal{E}} \langle \sigma_b, \sigma_c, q', t_b, \top \rangle$, with $q' = q \text{ after } (0, a)$,

– **Pass-uncont:** $\langle \sigma_b, \sigma_c, q, t, b \rangle \xhookrightarrow{(t,a) / \text{ pass-uncont}(t,a) / (t,a)}_{\mathcal{E}} \langle \epsilon, \Pi_\Sigma(\sigma_b).\sigma_c, q', t, \bot \rangle$, with $q' = q \text{ after } (0, a)$,

– **Store-cont:** $\langle \sigma_b, \sigma_c, q, t, b \rangle \xhookrightarrow{(t,c) / \text{ store-cont}((t,c)) / \epsilon}_{\mathcal{E}} \langle \epsilon, \Pi_\Sigma(\sigma_b).\sigma_c.c, q, t, \bot \rangle$,

– **Delay:** $\langle \sigma_b, \sigma_c, (l, v), t, b \rangle \xhookrightarrow{\epsilon / \text{ delay}(\delta) / \epsilon}_{\mathcal{E}} \langle \sigma_b, \sigma_c, (l, v + \delta), t + \delta, b \rangle$.

In a configuration $\langle \sigma_b, \sigma_c, q, t, b \rangle$, σ_b is the word to be output as time elapses; σ_c is the sequence of controllable actions from the input that are not used in σ_b; q is the state of the semantics reached after reading what has already been output; t is the current time instant, i.e., the time elapsed since the beginning of the run; and b indicates whether σ_b and σ_c should be computed (due to the reception of a new event for example).

The timed word σ_b corresponds to $\mathrm{nobs}(\sigma_b, t)$ from the definition of store_φ, whereas σ_c is the same as in the definition of store_φ. The state q represents σ_s, such that $q = \mathrm{Reach}(\sigma_s, t)$.

Proposition 10. The output of $\mathcal{E}$ for input σ is $\mathrm{E}_\varphi(\sigma)$.

As in the untimed setting, in Proposition 10, the output of the enforcement monitor is the concatenation of the outputs of the word labelling the path followed by the enforcement monitor when reading σ. A formal definition is given in the proof of this proposition, in appendix A.2.

Sketch of proof. The proof is done by induction on σ. When receiving a new event, rule $\text{store-cont}()$ can be applied if it is controllable, or rule $\text{pass-uncont}()$ if it is uncontrollable. Doing so, the last member of the configuration is set to $\bot$, meaning that the word to be emitted can be computed. If the input is in $\mathrm{Pre}(\varphi)$, then rule $\mathrm{compute}()$ can be applied, and then the second member of the configuration will have the same value as the second member of store_φ, and the same goes for the third members. Then, rule $\mathrm{delay}()$ can be applied, to reach the date of the first event in the second member of the current configuration, and then rule $\mathrm{dump}()$ can be applied to output it. This process can be repeated until the desired date is reached. Thus, when date t is reached, what has been emitted since the last rule $\text{store-cont}()$ or $\text{pass-uncont}()$ is $\mathrm{obs}(\sigma_b, t)$, where σ_b was computed by rule $\mathrm{compute}()$ as second member. Considering the definition of store_φ, it follows that the output of $\mathcal{E}$ with input σ at date t is $\mathrm{E}_\varphi(\sigma, t)$.

Table 2: Table showing the values of $(\sigma_s, \sigma_b, \sigma_c) = \text{store}_{\varphi_t}((1, \mathit{Auth}) \,.\, (2, \mathit{LockOn}) \,.\, (4, \texttt{Write}) \,.\, (5, \mathit{LockOff}) \,.\, (6, \mathit{LockOn}) \,.\, (7, \texttt{Write}) \,.\, (8, \mathit{LockOff}))$ over time.

t	σ_s	σ_b	σ_c
1	$(1, \mathit{Auth})$	ϵ	ϵ
2	$(1, \mathit{Auth}).(2, \mathit{LockOn})$	ϵ	ϵ
4	$(1, \mathit{Auth}).(2, \mathit{LockOn})$	ϵ	$\texttt{Write}$
5	$(1, \mathit{Auth}).(2, \mathit{LockOn}).(5, \mathit{LockOff})$	$(7, \texttt{Write})$	ϵ
6	$(1, \mathit{Auth}).(2, \mathit{LockOn}).(5, \mathit{LockOff}).$ $(6, \mathit{LockOn})$	ϵ	$\texttt{Write}$
7	$(1, \mathit{Auth}).(2, \mathit{LockOn}).(5, \mathit{LockOff}).$ $(6, \mathit{LockOn})$	ϵ	$\texttt{Write}\,.\,\texttt{Write}$
8	$(1, \mathit{Auth}).(2, \mathit{LockOn}).(5, \mathit{LockOff}).$ $(6, \mathit{LockOn}).(8, \mathit{LockOff})$	$(10, \texttt{Write}).(10, \texttt{Write})$	ϵ
10	$(1, \mathit{Auth}).(2, \mathit{LockOn}).(5, \mathit{LockOff}).$ $(6, \mathit{LockOn}).(8, \mathit{LockOff}).(10, \texttt{Write}).$ $(10, \texttt{Write})$	ϵ	ϵ

4.3. *Example*

Consider Fig. 2b, representing a property modelling the use of some shared writable device. We can get the status of a lock through the uncontrollable events LockOn and $\mathit{LockOff}$ indicating that the lock has been locked by someone else, and that it is unlocked, respectively. The uncontrollable event Auth is sent by the device to authorise writings. Once the Auth event is received, we are able to send the controllable event $\texttt{Write}$ after having waited some time for synchronisation. Each time the lock is taken and released, we must also wait before issuing a new $\texttt{Write}$ order. The sets of events are: $\Sigma_c = \{\texttt{Write}\}$ and $\Sigma_u = \{\mathit{Auth}, \mathit{LockOff}, \mathit{LockOn}\}$.

Now, let us follow the output of the store_φ function over time with the word $\sigma = (1, \mathit{Auth}) \,.\, (2, \mathit{LockOn}).(4, \texttt{Write}).(5, \mathit{LockOff}).(6, \mathit{LockOn}).(7, \texttt{Write}).(8, \mathit{LockOff})$ as input: let $(\sigma_s, \sigma_b, \sigma_c) = \text{store}_\varphi(\text{obs}(\sigma, t), t)$. Then the values taken by σ_s, σ_b and σ_c over time are given in Table 2. To calculate them, notice that for all valuation $\nu : \{x\} \to \mathbb{R}_{\geq 0}$, $\text{Safe}((l_1, \nu), \epsilon)$ and $\text{Safe}((l_2, \nu), \epsilon)$ hold, and so do $\text{Safe}((l_1, \nu), \texttt{Write})$ and $\text{Safe}((l_2, \nu), \texttt{Write})$, because whatever uncontrollable events we consider, it will be possible to delay the $\texttt{Write}$ event so that the current state remains in F_G.

We can also follow the execution of an enforcement monitor enforcing the property in Fig. 2b, watching the evolution of the configurations as semantic rules are applied. In a configuration, the input is on the right, the output on the left, and the middle is the current configuration of the enforcement monitor. The variable t defines the global time of the execution. Fig. 6 shows the execution of the enforcement monitor with input $(1, \mathit{Auth}).\ (2, \mathit{LockOn}).\ (4, \texttt{Write}).\ (5, \mathit{LockOff}).\ (6, \mathit{LockOn}).\ (7, \texttt{Write}).(8, \mathit{LockOff})$. In Fig. 6, valuations are represented as integers, giving the value of the only clock x of the property, $\mathit{LockOff}$ is abbreviated as $\texttt{off}$, LockOn as $\texttt{on}$, and $\texttt{Write}$ as $\texttt{w}$. First column depicts the dates of events, then red text is the current output (σ_s) of the enforcement mechanism, blue text shows the evolution of σ_b and green text depicts the remaining input word at this date. We can observe that the final output is the same as the one of the enforcement function : $(1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off}).(6, \texttt{on}).(8, \texttt{off}).(10, \texttt{w}).(10, \texttt{w})$

$$
\begin{array}{ll}
\text{t = 0} & \epsilon/(\epsilon, \epsilon, (l_0, 0), 0, \bot)/(1, \mathit{Auth}).(2, \texttt{on}).(4, \texttt{w}).(5, \texttt{off}).(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{delay}(1) \\
\text{t = 1} & \epsilon/(\epsilon, \epsilon, (l_0, 1), 1, \bot)/(1, \mathit{Auth}).(2, \texttt{on}).(4, \texttt{w}).(5, \texttt{off}).(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{pass-uncont}((1, \mathit{Auth})) \\
\text{t = 1} & (1, \mathit{Auth})/(\epsilon, \epsilon, (l_1, 1), 1, \bot)/(2, \texttt{on}).(4, \texttt{w}).(5, \texttt{off}).(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{compute}() \\
\text{t = 1} & (1, \mathit{Auth})/(\epsilon, \epsilon, (l_1, 1), 1, \top)/(2, \texttt{on}).(4, \texttt{w}).(5, \texttt{off}).(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{delay}(1) \\
\text{t = 2} & (1, \mathit{Auth})/(\epsilon, \epsilon, (l_1, 2), 2, \top)/(2, \texttt{on}).(4, \texttt{w}).(5, \texttt{off}).(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{pass-uncont}((2, \texttt{on})) \\
\text{t = 2} & (1, \mathit{Auth}).(2, \texttt{on})/(\epsilon, \epsilon, (l_2, 2), 2, \bot)/(4, \texttt{w}).(5, \texttt{off}).(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{compute}() \\
\text{t = 2} & (1, \mathit{Auth}).(2, \texttt{on})/(\epsilon, \epsilon, (l_2, 2), 2, \top)/(4, \texttt{w}).(5, \texttt{off}).(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{delay}(2) \\
\text{t = 4} & (1, \mathit{Auth}).(2, \texttt{on})/(\epsilon, \epsilon, (l_2, 4), 4, \top)/(4, \texttt{w}).(5, \texttt{off}).(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{store-cont}((4, \texttt{w})) \\
\text{t = 4} & (1, \mathit{Auth}).(2, \texttt{on})/(\epsilon, (4, \texttt{w}), (l_2, 4), 4, \bot)/(5, \texttt{off}).(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{compute}() \\
\text{t = 4} & (1, \mathit{Auth}).(2, \texttt{on})/(\epsilon, (4, \texttt{w}), (l_2, 4), 4, \top)/(5, \texttt{off}).(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{delay}(1) \\
\text{t = 5} & (1, \mathit{Auth}).(2, \texttt{on})/(\epsilon, (4, \texttt{w}), (l_2, 5), 5, \top)/(5, \texttt{off}).(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{pass-uncont}((5, \texttt{off})) \\
\text{t = 5} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off})/(\epsilon, (7, \texttt{w}), (l_1, 0), 5, \bot)/(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{compute}() \\
\text{t = 5} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off})/((7, \texttt{w}), \epsilon, (l_1, 0), 5, \top)/(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{delay}(1) \\
\text{t = 6} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off})/((7, \texttt{w}), \epsilon, (l_1, 1), 6, \top)/(6, \texttt{on}).(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{pass-uncont}((6, \texttt{on})) \\
\text{t = 6} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off}).(6, \texttt{on})/(\epsilon, (7, \texttt{w}), (l_2, 1), 6, \bot)/(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{compute}() \\
\text{t = 6} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off}).(6, \texttt{on})/(\epsilon, (7, \texttt{w}), (l_2, 1), 6, \top)/(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{delay}(1) \\
\text{t = 7} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off}).(6, \texttt{on})/(\epsilon, (7, \texttt{w}), (l_2, 2), 7, \top)/(7, \texttt{w}).(8, \texttt{off}) \\
& \downarrow \text{store-cont}((7, \texttt{w})) \\
\text{t = 7} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off}).(6, \texttt{on})/(\epsilon, (7, \texttt{w}).(7, \texttt{w}), (l_2, 2), 7, \bot)/(8, \texttt{off}) \\
& \downarrow \text{compute}() \\
\text{t = 7} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off}).(6, \texttt{on})/(\epsilon, (7, \texttt{w}).(7, \texttt{w}), (l_2, 2), 7, \top)/(8, \texttt{off}) \\
& \downarrow \text{delay}(1) \\
\text{t = 8} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off}).(6, \texttt{on})/(\epsilon, (7, \texttt{w}).(7, \texttt{w}), (l_2, 3), 8, \top)/(8, \texttt{off}) \\
& \downarrow \text{pass-uncont}((8, \texttt{off})) \\
\text{t = 8} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off}).(6, \texttt{on}).(8, \texttt{off})/(\epsilon, (10, \texttt{w}).(10, \texttt{w}), (l_1, 0), 8, \bot)/\epsilon \\
& \downarrow \text{compute}() \\
\text{t = 8} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off}).(6, \texttt{on}).(8, \texttt{off})/((10, \texttt{w}).(10, \texttt{w}), \epsilon, (l_1, 0), 8, \top)/\epsilon \\
& \downarrow \text{delay}(2) \\
\text{t = 10} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off}).(6, \texttt{on}).(8, \texttt{off})/((10, \texttt{w}).(10, \texttt{w}), \epsilon, (l_1, 2), 10, \top)/\epsilon \\
& \downarrow \text{dump}((10, \texttt{w})) \\
\text{t = 10} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off}).(6, \texttt{on}).(8, \texttt{off}).(10, \texttt{w})/((10, \texttt{w}), \epsilon, (l_1, 2), 10, \top)/\epsilon \\
& \downarrow \text{dump}((10, \texttt{w})) \\
\text{t = 10} & (1, \mathit{Auth}).(2, \texttt{on}).(5, \texttt{off}).(6, \texttt{on}).(8, \texttt{off}).(10, \texttt{w}).(10, \texttt{w})/(\epsilon, \epsilon, (l_1, 2), 10, \top)/\epsilon
\end{array}
$$

Figure 6: Execution of an enforcement monitor with input $(1, \mathit{Auth})$. $(2, \mathit{LockOn})$. $(4, \texttt{Write})$. $(5, \mathit{LockOff})$. $(6, \mathit{LockOn})$. $(7, \texttt{Write})$. $(8, \mathit{LockOff})$

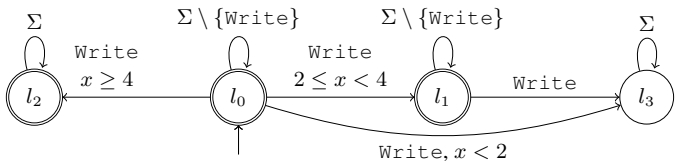


Figure 7: Example of Property without uncontrollable events

Remark 4. The enforcement mechanism defined in Definition 15 may provide in some situations longer sequences (i.e. longer timed words) than the approach of (Pinisetty et al., 2012) and (Pinisetty et al., 2014c) when applied only with controllable events. Indeed, our approach optimises the length of the output timed word. Consider the property described in Fig. 7 defined on the controllable set of actions Σ containing the action `Write` and some other ones, and the input timed word $(1, \texttt{Write}).(1.5, \texttt{Write})$ applied on the enforcement mechanism. The output obtained with our approach at date $t = 4$ is $(4, \texttt{Write}).(4, \texttt{Write})$ whereas the output obtained in (Pinisetty et al., 2012) would be $(2, \texttt{Write})$.

5. Related Work

Runtime enforcement was pioneered by the work of Schneider with security automata (Schneider, 2000), a runtime mechanism for enforcing safety properties. In (Schneider, 2000), monitors are able to stop the execution of the system once a deviation of the property has been detected. Later, Ligatti et al. proposed edit-automata, a more powerful model of enforcement monitors able to insert and suppress events from the execution. Later, more general models were proposed where the monitors can be synthesised from regular properties (Falcone et al., 2011). More recently, Bloem et al. (Bloem et al., 2015) presented a framework to synthesise enforcement monitors for reactive systems, called as *shields*, from a set of safety properties. A shield acts instantaneously and cannot buffer actions. Whenever a property violation is unavoidable, the shield allows to deviate from the property for k consecutive steps (as in (Charafeddine et al., 2015)). Whenever a second violation occurs within k steps, then the shield enters into a *fail-safe* mode, where it ensures only correctness. Another recent approach by Dolzehnko et al. (Dolzhenko et al., 2015) introduces Mandatory Result Automata (MRAs). MRAs extend edit-automata by refining the input-output relationship of an enforcement mechanism and thus allowing a more precise description of the enforcement abilities of an enforcement mechanism in concrete application scenarios. All the previously mentioned approaches considered untimed specifications, and do not consider uncontrollable events.

In the timed setting, several monitoring tools for timed specifications have been proposed. RT-Mac (Sammapun et al., 2005) permits to verify at runtime timeliness and reliability correctness. LARVA (Colombo et al., 2009a; Colombo et al., 2009b) takes as input safety properties expressed with DATEs (Dynamic Automata with Times and Events), a timed model similar to timed automata.

In previous work, we introduced *runtime enforcement for timed properties* (Pinisetty et al., 2012) specified by timed automata (Alur and Dill, 1992). We proposed a model of enforcement

monitors that work as *delayers*, that is, mechanisms that are able to delay the input sequence of timed events to correct it. While (Pinisetty et al., 2012) proposed synthesis techniques only for safety and co-safety properties, we then generalised the framework to synthesise an enforcement monitor for any regular timed property (Pinisetty et al., 2014b; Pinisetty et al., 2014c). In (Pinisetty et al., 2014a), we considered parametric timed properties, that is timed properties with data-events containing information from the execution of the monitored system. In our approach, the optimality of the enforcement mechanism is based on the maximisation of the length of the output sequence. When applied in the case of controllable events only, this improves the preceding results.

Basin et al. (Basin et al., 2011) introduced uncontrollable events for security automata (Schneider, 2000). The approach in (Basin et al., 2011) allows to enforce safety properties where some of the events in the specification are uncontrollable. More recently, they proposed a more general approach (Basin et al., 2013) related to enforcement of security policies with controllable and uncontrollable events. They presented several complexity results and how to synthesise enforcement mechanisms. In case of violation of the property, the system stops the execution. They handle discrete time, and clock ticks are considered as uncontrollable events. In our approach, we consider dense time using the expressiveness of timed automata, any regular properties, and our monitor are more flexible since they block the system only when delaying events cannot prevent violating the property, thus offering the possibility to correct many violations.

6. Conclusion and Future Work

This paper extends previous work on enforcement monitoring with the use of uncontrollable events, which are only observable by an enforcement device. We present a framework for enforcement monitoring for both untimed and timed regular properties, described with (timed) automata. We provide a functional and an operational description of the enforcement mechanism, and show their equivalence. Adding uncontrollable events leads to the necessity of changing the order between controllable and uncontrollable events, which requires some existing notions to be adapted. Therefore, we replace transparency with compliance to take this into account, and then give enforcement mechanisms, i.e. enforcement functions and enforcement monitors, for regular properties and regular timed properties. Since not every property can be enforced, we also give a condition, depending on the property and the input word, indicating whether the enforcement device is sound with respect to the property under scrutiny or not. The enforcement mechanisms output all the uncontrollable events received, and store the controllable ones, until soundness can be guaranteed. Then, they output events only when they can ensure that soundness will be satisfied. The proposed enforcement mechanisms are then sound and compliant. They are also optimal in the sense that they output the longest possible word, with the least possible dates in the timed setting, that allows them to be sound and compliant, even considering the reception of some uncontrollable events that they have to output.

One possible extension of this work would be to take some risks, outputting events even if some uncontrollable events could lead to a bad state, and introducing for example some probabilities. Implementing the given enforcement devices for the untimed setting is pretty straightforward, whereas implementation in the timed setting needs more attention due to computing in timed models. This is currently in progress.

References

Alur, R. and Dill, D. (1992). The theory of timed automata. In de Bakker, J., Huizing, C., de Roever, W., and Rozenberg, G., editors, *Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, pages 45–73. Springer Berlin Heidelberg.

Basin, D., Jugé, V., Klaedtke, F., and Zălinescu, E. (2013). Enforceable security policies revisited. *ACM Trans. Inf. Syst. Secur.*, 16(1):3:1–3:26.

Basin, D., Klaedtke, F., and Zalinescu, E. (2011). Algorithms for monitoring real-time properties. In Khurshid, S. and Sen, K., editors, *Proceedings of the 2nd International Conference on Runtime Verification (RV 2011)*, volume 7186 of *Lecture Notes in Computer Science*, pages 260–275. Springer-Verlag.

Bloem, R., Könighofer, B., Könighofer, R., and Wang, C. (2015). Shield synthesis: - runtime enforcement for reactive systems. In *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, pages 533–548.

Charafeddine, H., El-Harake, K., Falcone, Y., and Jaber, M. (2015). Runtime enforcement for component-based systems. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing, 2015*, pages 1789–1796.

Colombo, C., Pace, G. J., and Schneider, G. (2009a). LARVA — safer monitoring of real-time Java programs (tool paper). In Hung, D. V. and Krishnan, P., editors, *Proceedings of the 7th IEEE International Conference on Software Engineering and Formal Methods (SEFM 2009)*, pages 33–37. IEEE Computer Society.

Colombo, C., Pace, G. J., and Schneider, G. (2009b). Safe runtime verification of real-time properties. In Ouaknine, J. and Vaandrager, F. W., editors, *Proceedings of the 7th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2009)*, volume 5813 of *Lecture Notes in Computer Science*, pages 103–117. Springer.

Dolzhenko, E., Ligatti, J., and Reddy, S. (2015). Modeling runtime enforcement with mandatory results automata. *International Journal of Information Security*, 14(1):47–60.

Falcone, Y., Mounier, L., Fernandez, J., and Richier, J. (2011). Runtime enforcement monitors: composition, synthesis, and enforcement abilities. *Formal Methods in System Design*, 38(3):223–262.

Ligatti, J., Bauer, L., and Walker, D. (2009). Run-time enforcement of nonsafety policies. *ACM Trans. Inf. Syst. Secur.*, 12(3):19:1–19:41.

Pinisetty, S., Falcone, Y., Jéron, T., and Marchand, H. (2014a). Runtime enforcement of parametric timed properties with practical applications. In Lesage, J., Faure, J., Cury, J. E. R., and Lennartson, B., editors, *12th International Workshop on Discrete Event Systems, WODES 2014, Cachan, France, May 14-16, 2014.*, pages 420–427. International Federation of Automatic Control.

Pinisetty, S., Falcone, Y., Jéron, T., and Marchand, H. (2014b). Runtime enforcement of regular timed properties. In Cho, Y., Shin, S. Y., Kim, S., Hung, C., and Hong, J., editors, *Symposium on Applied Computing, SAC 2014, Gyeongju, Republic of Korea - March 24 - 28, 2014*, pages 1279–1286. ACM.

Pinisetty, S., Falcone, Y., Jéron, T., Marchand, H., Rollet, A., and Nguena-Timo, O. (2014c). Runtime enforcement of timed properties revisited. *Formal Methods in System Design*, 45(3):381–422.

Pinisetty, S., Falcone, Y., Jéron, T., Marchand, H., Rollet, A., and Nguena-Timo, O. L. (2012). Runtime enforcement of timed properties. In Qadeer, S. and Tasiran, S., editors, *Runtime Verification, Third International Conference, RV 2012, Istanbul, Turkey, September 25-28, 2012, Revised Selected Papers*, volume 7687 of *Lecture Notes in Computer Science*, pages 229–244. Springer.

Renard, M., Falcone, Y., Rollet, A., Pinisetty, S., Jéron, T., and Marchand, H. (2015). Enforcement of (timed) properties with uncontrollable events. In Leucker, M., Rueda, C., and Valencia, F. D., editors, *Theoretical Aspects of Computing - ICTAC 2015*, volume 9399 of *Lecture Notes in Computer Science*, pages 542–560. Springer International Publishing.

Sammapun, U., Lee, I., and Sokolsky, O. (2005). RT-MaC: Runtime monitoring and checking of quantitative and probabilistic properties. *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications*, 0:147–153.

Schneider, F. B. (2000). Enforceable security policies. *ACM Trans. Inf. Syst. Secur.*, 3(1):30–50.

Appendix A. Proofs

A.1. *Proofs for the Untimed Setting*

In all this section, we will use the notations from Section 3, meaning that φ is a property whose associated automaton is $\mathcal{A}_\varphi = \langle Q, q_0, \Sigma, \rightarrow, F\rangle$. In some proofs, we also use notations from Definition 7.

Proposition 1. E_φ as defined in Definition 7 is an enforcement function.

Proof. Let us consider $\sigma \in \Sigma^*$, and $\sigma' \in \Sigma^*$. If $\sigma' = \epsilon$, then $\mathrm{E}_\varphi(\sigma) = \mathrm{E}_\varphi(\sigma.\sigma') \preccurlyeq \mathrm{E}_\varphi(\sigma.\sigma')$. Otherwise, let $(\sigma_s, \sigma_c) = \mathrm{store}_\varphi(\sigma)$, $a = \sigma'(1)$, and $(\sigma_t, \sigma_d) = \mathrm{store}_\varphi(\sigma.a)$. Then, if $a \in \Sigma_\mathrm{u}$, $\sigma_t = \sigma_s.a.\sigma'_s$, where σ'_s is defined in Definition 7, meaning that $\sigma_s \preccurlyeq \sigma_t$. If $a \in \Sigma_\mathrm{c}$, then $\sigma_t = \sigma_s.\sigma''_s$, where σ''_s is defined in Definition 7, thus again, $\sigma_s \preccurlyeq \sigma_t$. In both cases, $\mathrm{E}_\varphi(\sigma) = \sigma_s \preccurlyeq \sigma_t = \mathrm{E}_\varphi(\sigma.a)$. Since the order $\preccurlyeq$ is transitive, this means that $\mathrm{E}_\varphi(\sigma) \preccurlyeq \mathrm{E}_\varphi(\sigma.a) \preccurlyeq \mathrm{E}_\varphi(\sigma.a.\sigma'(2)) \preccurlyeq \ldots \preccurlyeq \mathrm{E}_\varphi(\sigma.\sigma')$. Thus E_φ is an enforcement function. □

Lemma 1. For $q \in Q$, $\sigma \in \Sigma_\mathrm{c}^*$, $u \in \Sigma_\mathrm{u}$, and $w \preccurlyeq \sigma$,
$(\mathrm{Safe}_\mathrm{int}(q, \sigma, \emptyset) \wedge \mathrm{Safe}_\mathrm{int}(q \text{ after } (u.w), w^{-1}.\sigma, \{(q, \sigma)\})) \implies \mathrm{Safe}_\mathrm{int}(q \text{ after } (u.w), w^{-1}.\sigma, \emptyset)$.

Proof. Let us consider $q \in Q, \sigma \in \Sigma_\mathrm{c}^*, u \in \Sigma_\mathrm{u}$, and $w \preccurlyeq \sigma$ such that $\mathrm{Safe}_\mathrm{int}(q, \sigma, \emptyset)$ and $\mathrm{Safe}_\mathrm{int}(q\mathrm{after}$ $(u.w), w^{-1}.\sigma, \{(q, \sigma)\})$ hold.

—*Case 1:* $w \neq \epsilon$. Then, $w^{-1}.\sigma \neq \sigma$, thus for all inductive calls in the computation of $\mathrm{Safe}_\mathrm{int}(q \text{ after } (u.w), w^{-1}.\sigma, \{(q, \sigma)\})$, there exists $q' \in Q$, $\sigma' \in \Sigma_\mathrm{c}^*$, and $Q' \subseteq Q$ such that the inductive call is $\mathrm{Safe}_\mathrm{int}(q', \sigma', \{(q, \sigma)\} \cup Q')$, with $\sigma' \prec \sigma$. Since $\sigma' \prec \sigma$, $(q', \sigma') \neq (q, \sigma)$, meaning that the condition $(q', \sigma') \in \{(q, \sigma)\} \cup Q'$ is equivalent to $(q', \sigma') \in Q'$. Thus, the inductive call is equivalent to $\mathrm{Safe}_\mathrm{int}(q', \sigma', Q')$ It follows that $\mathrm{Safe}_\mathrm{int}(q \text{ after } (u.w), w^{-1}.\sigma, \emptyset)$ holds, since $\mathrm{Safe}_\mathrm{int}(q\,\mathrm{after}\,(u.w), w^{-1}.\sigma, \{(q, \sigma)\})$ holds, and whether (q, σ) belongs to the third parameter or not does not change the value.

—*Case 2:* $w = \epsilon$. Then, $\mathrm{Safe}_\mathrm{int}(q, \sigma, \emptyset)$ and $\mathrm{Safe}_\mathrm{int}(q \text{ after } u, \sigma, \{(q, \sigma)\})$ hold. Let us suppose that $\mathrm{Safe}_\mathrm{int}(q\mathrm{after}u, \sigma, \emptyset)$ does not hold. Then, there exists $u' \in \Sigma_\mathrm{u}$ such that for all $w' \preccurlyeq \sigma$, $\mathrm{Safe}_\mathrm{int}(q\mathrm{after}$ $(u.u'.w'), w'^{-1}.\sigma, \{(q\mathrm{after}u, \sigma)\})$ does not hold. Now, considering that $\mathrm{Safe}_\mathrm{int}(q\mathrm{after}u, \sigma, \{(q, \sigma)\})$ holds, and $u' \in \Sigma_\mathrm{u}$, this means that there exists $w_1 \preccurlyeq \sigma$ such that $\mathrm{Safe}_\mathrm{int}(q\mathrm{after}(u.u'.w_1), w_1^{-1}.\sigma, \{(q, \sigma), (q\mathrm{after}$ $u, \sigma)\})$ holds. Following the same reasoning as in the first case, if $w_1 \neq \epsilon$, then $\mathrm{Safe}_\mathrm{int}(q \text{ after } (u.u'.w_1), w_1^{-1}.\sigma, \{(q, \sigma), (q \text{ after } u, \sigma)\})$ is equivalent to $\mathrm{Safe}_\mathrm{int}(q \text{ after } (u.u'.w_1), w_1^{-1}.\sigma, \emptyset)$, and also to $\mathrm{Safe}_\mathrm{int}(q \text{ after } (u.u'.w_1), w_1^{-1}.\sigma, \{(q \text{ after } u, \sigma)\})$. This is absurd because $\mathrm{Safe}_\mathrm{int}(q \text{ after } (u.u'.w_1), w_1^{-1}.\sigma, \{(q \text{ after } u, \sigma)\})$ does not hold. This means that $w_1 = \epsilon$. Thus, $\mathrm{Safe}_\mathrm{int}(q \text{ after } (u.u'), \sigma, \{(q, \sigma), (q \text{ after } u, \sigma)\})$ holds. Repeating this process, we can find $\sigma_u \in \Sigma_\mathrm{u}^*$ such that $\mathrm{Safe}_\mathrm{int}(q \text{ after } \sigma_u, \sigma, Q')$ does not hold, and $q \text{ after } \sigma_u \in Q'$, with $Q' = \{(q', \sigma) \mid \exists \sigma'_u \preccurlyeq \sigma_u, q' = q \text{ after } \sigma'_u\}$. Since $q \text{ after } \sigma_u \in Q'$, and $\mathrm{Safe}_\mathrm{int}(q \text{ after } \sigma_u, \sigma, Q')$ does not hold, this means that there exists $q_b \in Q$ such that $(q_b, \sigma) \in Q'$ and $q_b \notin F$. If $(q, \sigma) \notin Q'$, then $\mathrm{Safe}_\mathrm{int}(q \text{ after } u, \sigma, \{(q, \sigma)\})$ is equivalent to $\mathrm{Safe}_\mathrm{int}(q \text{ after } u, \sigma, \emptyset)$, which is absurd. Thus, $(q, \sigma) \in Q'$, so let $\sigma_{u1} \preccurlyeq \sigma_u$ be such that $q \text{ after } \sigma_{u1} = q$. Then, considering that $\mathrm{Safe}_\mathrm{int}(q \text{ after } u, \sigma, \{(q, \sigma)\})$ holds, this means that $\mathrm{Safe}_\mathrm{int}(q, \sigma, \{q, \sigma\} \cup Q_1)$ holds, with $Q_1 = \{(q', \sigma) \mid \exists \sigma'_u \preccurlyeq \sigma_{u1}, q' = q \text{ after } \sigma'_u\}$. Thus,

for all q' such that $(q', \sigma) \in Q_1$, $q' \in F$. Since we supposed that $\mathrm{Safe}_{\mathrm{int}}(q, \sigma, \emptyset)$ holds, it follows that $\mathrm{Safe}_{\mathrm{int}}(q \text{ after } \sigma_{u2}, \sigma, Q_2)$ holds, with $\sigma_{u2} = \sigma_{u1}^{-1}.\sigma_u$, and $Q_2 = \{(q', \sigma) \mid \exists \sigma'_u \preccurlyeq \sigma_{u2}, q' = q \text{ after } \sigma'_u\}$ (following the same reasoning as before, we can show that $w = \epsilon$). If $q \text{ after } \sigma_{u2} = q \text{ after } \sigma_u \in Q_2$, then for all $(q', \sigma) \in Q_2$, $q' \in F$ since $\mathrm{Safe}_{\mathrm{int}}(q \text{ after } \sigma_{u2}, \sigma, Q_2)$ holds. Since $Q' = Q_1 \cup Q_2$, it is absurd, thus $\mathrm{Safe}_{\mathrm{int}}(q \text{after} u, \sigma, \emptyset)$ holds. Otherwise, $q \text{after} \sigma_u \in Q_1$ (because q after $\sigma_u \in Q'$). Then, there exists $\sigma_{u3} \preccurlyeq \sigma_{u2}$ such that q after $\sigma_{u3} = q$ after σ_u. Let us consider σ_{u4} as the smallest word satisfying $\sigma_{u4} \preccurlyeq \sigma_{u3}^{-1}.\sigma_{u2}$, and $(q \text{after} \sigma_u.\sigma_{u4}, \sigma) \in Q_2 \cup Q_3$, with $Q_3 = \{(q', \sigma) \mid \exists \sigma'_u \preccurlyeq \sigma_{u3}^{-1}.\sigma_{u2}, q' = q \text{after} \sigma_u.\sigma'_u\}$. σ_{u4} must exist because $q \text{after} \sigma_u.(\sigma_{u3}^{-1}.\sigma_{u2}) = q \in Q_2$, thus $\sigma_{u3}^{-1}.\sigma_{u2}$ satisfies the property. Then, $\mathrm{Safe}_{\mathrm{int}}(q \text{ after } \sigma_u.\sigma_{u4}, \sigma, Q_2 \cup Q_4)$ holds, with $Q_4 = \{(q', \sigma) \mid \exists \sigma'_u \preccurlyeq \sigma_{u4}, q' = q \text{ after } \sigma_u.\sigma'_u\}$, and q after $\sigma_u.\sigma_{u4} \in Q_2 \cup Q_4$. This means that for all $(q', \sigma) \in Q_2 \cup Q_4$, $q' \in F$. Thus, for all $(q', \sigma) \in Q_2$, $q' \in F$. This means that for all $(q', \sigma) \in Q_1 \cup Q_2 = Q'$, $q' \in F$, which is absurd, meaning that $\mathrm{Safe}_{\mathrm{int}}(q \text{ after } u, \sigma, \emptyset)$ must hold.

In all cases, $\mathrm{Safe}_{\mathrm{int}}(q \text{ after } (u.w), w^{-1}.\sigma, \emptyset)$ holds. Thus:
$(\mathrm{Safe}_{\mathrm{int}}(q, \sigma, \emptyset) \wedge \mathrm{Safe}_{\mathrm{int}}(q \text{ after } (u.w), w^{-1}.\sigma, \{(q, \sigma)\})) \implies \mathrm{Safe}_{\mathrm{int}}(q \text{ after } (u.w), w^{-1}.\sigma, \emptyset)$. □

Proposition 2. E_φ is sound with respect to φ in $\mathrm{Pre}(\varphi)$, as per Definition 3.

Proof. Let $\mathrm{P}(\sigma)$ be the predicate: "$\sigma \in \mathrm{Pre}(\varphi) \implies ((\mathrm{E}_\varphi(\sigma) \models \varphi) \wedge ((\sigma_s, \sigma_b) = \mathrm{store}_\varphi(\sigma)) \implies \mathrm{Safe}(\mathrm{Reach}(\sigma_s), \sigma_b))$". Let us prove that $\forall \sigma \in \Sigma^*, \mathrm{P}(\sigma)$ holds.

—*Induction basis*: $\mathrm{E}_\varphi(\epsilon) = \epsilon$. If $\epsilon \in \mathrm{Pre}(\varphi)$, then, following the definition of $\mathrm{Pre}(\varphi)$, $\epsilon \models \varphi$ and $\mathrm{Safe}(\mathrm{Reach}(\epsilon), \epsilon)$. Since $\mathrm{store}_\varphi(\epsilon) = (\epsilon, \epsilon)$, this means that $\mathrm{P}(\epsilon)$ holds.

—*Induction step*: Suppose now that, for some $\sigma \in \Sigma^*$, $\mathrm{P}(\sigma)$ holds. Let us consider $a \in \Sigma$. Let us prove that $\mathrm{P}(\sigma.a)$ holds. Let us consider $(\sigma_s, \sigma_b) = \mathrm{store}_\varphi(\sigma)$, and $(\sigma_t, \sigma_c) = \mathrm{store}_\varphi(\sigma.a)$.

– *Case 1:* $(\sigma.a) \notin \mathrm{Pre}(\varphi)$. Then $\mathrm{P}(\sigma.a)$ holds.

– *Case 2:* $(\sigma.a) \in \mathrm{Pre}(\varphi) \wedge \sigma \notin \mathrm{Pre}(\varphi)$. Then, there exists $\sigma' \in \Sigma^*, (\sigma.a)_{|\Sigma_\mathrm{u}} \preccurlyeq \sigma' \wedge (\sigma.a)_{|\Sigma_\mathrm{u}} = \sigma'_{|\Sigma_\mathrm{u}} \wedge \sigma'_{|\Sigma_\mathrm{c}} \preccurlyeq (\sigma.a)_{|\Sigma_\mathrm{c}} \wedge \sigma' \models \varphi \wedge \mathrm{Safe}(\mathrm{Reach}(\sigma'), (\sigma'_{|\Sigma_\mathrm{c}})^{-1}.\sigma_{|\Sigma_\mathrm{c}})$. Since $\sigma \notin \mathrm{Pre}(\varphi), \sigma_s = \sigma_{|\Sigma_\mathrm{u}}$, thus $\sigma_t \succcurlyeq \sigma_{|\Sigma_\mathrm{u}}$, and $\sigma_b = \sigma_{|\Sigma_\mathrm{c}}$.

• If $a \in \Sigma_\mathrm{u}$, then $(\sigma.a)_{|\Sigma_\mathrm{u}} = \sigma_s.a$, and $(\sigma.a)_{|\Sigma_\mathrm{c}} = \sigma_{|\Sigma_\mathrm{c}} = \sigma_b$, thus $((\sigma.a)_{|\Sigma_\mathrm{u}})^{-1}.\sigma' \in \{w \preccurlyeq \sigma_b \mid \sigma_s.a.w \models \varphi \wedge \mathrm{Safe}(\mathrm{Reach}(\sigma_s.a.w), w^{-1}.\sigma_b)\}$. Thus, following the construction of σ_t, $\mathrm{E}_\varphi(\sigma.a) = \sigma_t \models \varphi \wedge \mathrm{Safe}(\mathrm{Reach}(\sigma_t), \sigma_c)$. This means that $\mathrm{P}(\sigma.a)$ holds.

• If $a \in \Sigma_\mathrm{c}$, then $(\sigma.a)_{|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}} = \sigma_s$, and $(\sigma.a)_{|\Sigma_\mathrm{c}} = \sigma_{|\Sigma_\mathrm{c}}.a = \sigma_b.a$. Thus, $((\sigma.a)_{|\Sigma_\mathrm{u}})^{-1}.\sigma' \in \{w \preccurlyeq \sigma_b.a \mid \sigma_s.w \models \varphi \wedge \mathrm{Safe}(\mathrm{Reach}(\sigma_s.w), w^{-1}.(\sigma_b.a))\}$. Thus, following the definition of σ_t, $\sigma_t = \mathrm{E}_\varphi(\sigma.a) \models \varphi$ and $\mathrm{Safe}(\mathrm{Reach}(\sigma_t), \sigma_c)$ holds. This means that $\mathrm{P}(\sigma.a)$ holds.

– *Case 3:* $\sigma \in \mathrm{Pre}(\varphi)$ (and then $(\sigma.a) \in \mathrm{Pre}(\varphi)$ since $\mathrm{Pre}(\varphi)$ is extension-closed). Moreover, since $\mathrm{P}(\sigma)$ holds, $\mathrm{Safe}(\mathrm{Reach}(\sigma_s), \sigma_b)$ holds.

• If $a \in \Sigma_\mathrm{u}$, then, since $\mathrm{Safe}(\mathrm{Reach}(\sigma_s), \sigma_b)$ holds, there exists $w \preccurlyeq \sigma_b$ such that $\mathrm{Safe}_{\mathrm{int}}(\mathrm{Reach}(\sigma_s.a.w), w^{-1}.\sigma_b, \{(\mathrm{Reach}(\sigma_s), \sigma_b)\})$ holds, and $\mathrm{Reach}(\sigma_s.a.w) \in F$. By induction hypothesis, $\mathrm{Safe}_{\mathrm{int}}(\mathrm{Reach}(\sigma_s), \sigma_b, \emptyset)$ holds. Following lemma 1, since $a \in \Sigma_\mathrm{u}$, this means that $\mathrm{Safe}_{\mathrm{int}}(\mathrm{Reach}(\sigma_s.a.w), w^{-1}.\sigma_b, \emptyset)$ holds, i.e. $\mathrm{Safe}(\mathrm{Reach}(\sigma_s.a.w), w^{-1}.\sigma_b)$ holds. Thus, $w \in \mathrm{G} = \{w' \preccurlyeq \sigma_b \mid \sigma_s.a.w' \models \varphi \wedge \mathrm{Safe}(\mathrm{Reach}(\sigma_s.a.w'), \sigma_b)\}$. . Since $\mathrm{G} \neq \emptyset$, σ'_s from Definition 7 is in G. This means that $\mathrm{E}_\varphi(\sigma.a) = \sigma_t = \sigma_s.a.\sigma'_s \models \varphi$, and that $\mathrm{Safe}(\mathrm{Reach}(\sigma_t), \sigma_c)$ holds, since $\sigma_c = \sigma_b$. Thus, $\mathrm{P}(\sigma.a)$ holds.

• If $a \in \Sigma_\mathrm{c}$, then, by induction hypothesis, $\sigma_s \models \varphi$, and $\mathrm{Safe}(\mathrm{Reach}(\sigma_s), \sigma_b)$ holds. Thus, $\mathrm{Safe}(\mathrm{Reach}(\sigma_s), \sigma_b.a)$ also holds. Thus, $\epsilon \in E_2 = \{w' \preccurlyeq \sigma_b.a \mid \sigma_s.w' \models \varphi \wedge \mathrm{Safe}(\mathrm{Reach}(\sigma_s.w'), w'^{-1}.\sigma_b.a)\}$. Since $E_2 \neq \emptyset$, this means that σ''_s as defined in Definition 7 is in E_2. It follows that $\mathrm{E}_\varphi(\sigma.a) =$

$\sigma_t = \sigma_s.\sigma''_s \models \varphi$, and that $\mathrm{Safe}(\mathrm{Reach}(\sigma_s.\sigma''_s), {\sigma''_s}^{-1}.\sigma_b) = \mathrm{Safe}(\mathrm{Reach}(\sigma_t), \sigma_c)$ holds. Thus, $\mathrm{P}(\sigma.a)$ holds.

In all cases, $\mathrm{P}(\sigma.a)$ holds. Thus, $\mathrm{P}(\sigma) \implies \mathrm{P}(\sigma.a)$.

By induction on σ, $\forall \sigma \in \Sigma^*, (\sigma \in \mathrm{Pre}(\varphi)) \implies (\mathrm{E}_\varphi(\sigma) \models \varphi \wedge (((\sigma_s, \sigma_b) = \mathrm{store}_\varphi(\sigma)) \implies \mathrm{Safe}(\mathrm{Reach}(\sigma_s), \sigma_b)$. In particular, for all $\sigma \in \Sigma^*$, $(\sigma \in \mathrm{Pre}(\varphi)) \implies (\mathrm{E}_\varphi(\sigma) \models \varphi)$. This means that E_φ is sound with respect to φ in $\mathrm{Pre}(\varphi)$. □

Proposition 3. E_φ is compliant, as per Definition 4.

Proof. For $\sigma \in \Sigma^*$, let $\mathrm{P}(\sigma)$ be the predicate: "$((\sigma_s, \sigma_c) = \mathrm{store}_\varphi(\sigma)) \implies (\sigma_{s|\Sigma_\mathrm{c}}.\sigma_c = \sigma_{|\Sigma_\mathrm{c}} \wedge \sigma_{s|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}})$". Let us prove that for all $\sigma \in \Sigma^*$, $\mathrm{P}(\sigma)$ holds.

—*Induction basis*: $\mathrm{store}_\varphi(\epsilon) = (\epsilon, \epsilon)$, and $\epsilon_{|\Sigma_\mathrm{c}}.\epsilon$, and $\epsilon_{|\Sigma_\mathrm{u}} = \epsilon_{|\Sigma_\mathrm{u}}$. Thus $\mathrm{P}(\epsilon)$ holds.

—*Induction step*: Let us suppose that for $\sigma \in \Sigma^*$, $\mathrm{P}(\sigma)$ holds. Let us consider $(\sigma_s, \sigma_c) = \mathrm{store}_\varphi(\sigma)$, $a \in \Sigma$, and $(\sigma_t, \sigma_d) = \mathrm{store}_\varphi(\sigma.a)$. Let us prove that $\mathrm{P}(\sigma.a)$ holds.

– *Case 1:* $a \in \Sigma_\mathrm{u}$. Then, $\sigma_t = \sigma_s.a.\sigma'_s$, where σ'_s is defined in Definition 7, and $\sigma_t.\sigma_d = \sigma_s.a.\sigma_c$. Therefore, $\sigma_{t|\Sigma_\mathrm{c}}.\sigma_d = (\sigma_t.\sigma_d)_{|\Sigma_\mathrm{c}}$, since $\sigma_d \in \Sigma_\mathrm{c}^*$. Thus, $\sigma_{t|\Sigma_\mathrm{c}}.\sigma_d = \sigma_{s|\Sigma_\mathrm{c}}.\sigma_c$. Since $\mathrm{P}(\sigma)$ holds, $\sigma_{t|\Sigma_\mathrm{c}}.\sigma_d = \sigma_{|\Sigma_\mathrm{c}} = (\sigma.a)_{|\Sigma_\mathrm{c}}$.
Moreover, since $\sigma'_s \in \Sigma_\mathrm{c}^*$, $\sigma_{t|\Sigma_\mathrm{u}} = \sigma_{s|\Sigma_\mathrm{u}}.a$. Since $\mathrm{P}(\sigma)$ holds, this means that $\sigma_{t|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}}.a = (\sigma.a)_{|\Sigma_\mathrm{u}}$.
Thus $\mathrm{P}(\sigma.a)$ holds.

– *Case 2:* $a \in \Sigma_\mathrm{c}$. Then $\sigma_t = \sigma_s.\sigma''_s$, where σ''_s is defined in Definition 7, and $\sigma_t.\sigma_d = \sigma_s.\sigma_c.a$. Therefore, $\sigma_{t|\Sigma_\mathrm{c}}.\sigma_d = (\sigma_t.\sigma_d)_{|\Sigma_\mathrm{c}} = (\sigma_s.\sigma_c.a)_{|\Sigma_\mathrm{c}} = \sigma_{s|\Sigma_\mathrm{c}}.\sigma_c.a$. Since $\mathrm{P}(\sigma)$ holds, this means that $\sigma_{t|\Sigma_\mathrm{c}}.\sigma_d = \sigma_{\Sigma_\mathrm{c}}.a = (\sigma.a)_{|\Sigma_\mathrm{c}}$.
Moreover, since $\sigma''_s \in \Sigma_\mathrm{c}^*$, $\sigma_{t|\Sigma_\mathrm{u}} = \sigma_{s|\Sigma_\mathrm{u}}$. Since $\mathrm{P}(\sigma)$ holds, this means that $\sigma_{t|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}} = (\sigma.a)_{|\Sigma_\mathrm{u}}$.
Thus $\mathrm{P}(\sigma.a)$ holds.

In both cases, $\mathrm{P}(\sigma.a)$ holds.
Thus, for all $\sigma \in \Sigma^*$, for all $a \in \Sigma$, $\mathrm{P}(\sigma) \implies \mathrm{P}(\sigma.a)$.

By induction on σ, for all $\sigma \in \Sigma^*$, $((\sigma_s, \sigma_c) = \mathrm{store}_\varphi(\sigma)) \implies (\sigma_{s|\Sigma_\mathrm{c}}.\sigma_c = \sigma_{|\Sigma_\mathrm{c}} \wedge \sigma_{s|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}})$.
Moreover, if $\sigma \in \Sigma^*$, $u \in \Sigma_\mathrm{u}$, $(\sigma_s, \sigma_c) = \mathrm{store}_\varphi(\sigma)$, and $(\sigma_t, \sigma_d) = \mathrm{store}_\varphi(\sigma.u)$, then $\sigma_t = \sigma_s.u.\sigma'_s$, where σ'_s is defined in Definition 7. Thus $\sigma_s.u \preccurlyeq \sigma_t$, and since $\sigma_s = \mathrm{E}_\varphi(\sigma)$, and $\sigma_t = \mathrm{E}_\varphi(\sigma.u)$, it follows that $\mathrm{E}_\varphi(\sigma).u \preccurlyeq \mathrm{E}_\varphi(\sigma.u)$.
Thus, for all $\sigma \in \Sigma^*$, $\mathrm{E}_\varphi(\sigma)_{|\Sigma_\mathrm{c}} \preccurlyeq \sigma_{|\Sigma_\mathrm{c}} \wedge \mathrm{E}_\varphi(\sigma)_{|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}} \wedge \forall u \in \Sigma_\mathrm{u}, \mathrm{E}_\varphi(\sigma).u \preccurlyeq \mathrm{E}_\varphi(\sigma.u)$, meaning that E_φ is compliant. □

Proposition 4. E_φ is optimal in $\mathrm{Pre}(\varphi)$, as per Definition 5.

Proof. Let E be an enforcement function such that $\mathrm{compliant}(E, \Sigma_\mathrm{c}, \Sigma_\mathrm{u})$, and let $\sigma \in \mathrm{Pre}(\varphi)$ and $a \in \Sigma$ be such that $E(\sigma) = \mathrm{E}_\varphi(\sigma) \wedge |E(\sigma.a)| > |\mathrm{E}_\varphi(\sigma)|$. Let us consider $(\sigma_s, \sigma_c) = \mathrm{store}_\varphi(\sigma)$.

—*Case 1:* $a \in \Sigma_\mathrm{u}$. Then, since E is compliant, and $E(\sigma) = \mathrm{E}_\varphi(\sigma) = \sigma_s$, there exists $\sigma'_{s1} \preccurlyeq \sigma_c$ such that $E(\sigma.a) = E(\sigma).a.\sigma'_{s1} = \sigma_s.a.\sigma'_{s1}$. Moreover, there exists $\sigma'_s \preccurlyeq \sigma_c$ such that $\mathrm{E}_\varphi(\sigma.a) = \mathrm{E}_\varphi(\sigma).a.\sigma'_s = \sigma_s.a.\sigma'_s$. Since $|E(\sigma.a)| > |\mathrm{E}_\varphi(\sigma.a)|$, $|\sigma'_{s1}| > |\sigma'_s|$. Considering that $\sigma'_s = \max_\preccurlyeq(S \cup \{\epsilon\})$, with $S = \{w \preccurlyeq \sigma_b \mid \sigma_s.a.w \models \varphi \wedge \mathrm{Safe}(\mathrm{Reach}(\sigma_s.a.w), w^{-1}.\sigma_c)\}$, it follows that $\sigma'_{s1} \notin S$. This means that either $\sigma_s.a.\sigma'_{s1} = E(\sigma.a) \not\models \varphi$, or $\mathrm{Safe}(\mathrm{Reach}(\sigma_s.a.\sigma'_{s1}), {\sigma'_{s1}}^{-1}.\sigma_c)$ does not hold.
If $\sigma_s.a.\sigma_{s1} \not\models \varphi$, then $\epsilon \in \Sigma_\mathrm{u}^*$ is such that $\mathrm{E}_\varphi(\sigma.a.\epsilon) \not\models \varphi$. Otherwise, $\mathrm{Safe}(\mathrm{Reach}(\mathrm{E}_\varphi(\sigma).a.\sigma'_{s1}), {\sigma'_{s1}}^{-1}.\sigma_c)$ does not hold. Then, following the definition of Safe and $\mathrm{Safe}_\mathrm{int}$, this means that there exists $u_1 \in \Sigma_\mathrm{u}$ such that for all $w \preccurlyeq \sigma'^{-1}_{s1}.\sigma_b$, $\mathrm{Safe}_\mathrm{int}(\mathrm{Reach}(\sigma_s.a.\sigma'_{s1}.u_1.w), w^{-1}.(\sigma'^{-1}_{s1}.\sigma_b), \{(\mathrm{Reach}(\sigma_s.a.\sigma_{s1}), \sigma'^{-1}_{s1}.\sigma_b)\})$

does not hold, or $\mathrm{Reach}(\sigma_s.a.\sigma'_{s1}.u_1.w) \notin F$. Then, let us consider $E(\sigma.a.u_1)$. Since E is compliant, and $u_1 \in \Sigma_\mathrm{u}$, there exists $\sigma'_{s2} \preccurlyeq \sigma'^{-1}_{s1}.\sigma_b$ such that $E(\sigma.a.u_1) = \sigma_s.a.\sigma'_{s1}.u_1.\sigma'_{s2}$. Thus, either $\mathrm{Reach}(E(\sigma.a.u_1)) \notin F$, meaning that $u_1 \in \Sigma_\mathrm{u}^*$ is such that $E(\sigma.a.u_1) \not\models \varphi$, or $\mathrm{Safe}_\mathrm{int}(\mathrm{Reach}(\sigma_s.a.\sigma'_{s1}.u_1.\sigma'_{s2}), \sigma'^{-1}_{s2}.(\sigma'^{-1}_{s1}.\sigma_b), \{(\mathrm{Reach}(\sigma_s.a.\sigma'_{s1}), \sigma'^{-1}_{s1}.\sigma_b)\})$ does not hold. Then, this can be iterated, constructing a word $\sigma_u \in \Sigma_\mathrm{u}^*$ such that $\mathrm{Reach}(E(\sigma.a.\sigma_u)) \notin F$, or such that there exists $\sigma'_{sn} \preccurlyeq \sigma_b$ such that $\mathrm{Safe}_\mathrm{int}(\mathrm{Reach}(E(\sigma.a.\sigma_u)), \sigma'^{-1}_{sn}.\sigma_b, Q')$ does not hold, where $Q' = \{(q, \sigma') \in Q \times \Sigma_\mathrm{c}^* \mid \exists \sigma''_u \preccurlyeq \sigma_u, q = \mathrm{Reach}(E(\sigma.a.\sigma''_u)) \wedge \sigma' = (E(\sigma.a.\sigma''_u)_{|\Sigma_\mathrm{c}})^{-1}.\sigma_{|\Sigma_\mathrm{c}}\}$, with $(\mathrm{Reach}(E(\sigma.a.\sigma_u)), \sigma'^{-1}_{sn}.\sigma_b) \in Q'$, or $\sigma'^{-1}_{sn}.\sigma_b = \epsilon$. If $\sigma'^{-1}_{sn}.\sigma_b = \epsilon$, then since $\mathrm{Safe}_\mathrm{int}(\mathrm{Reach}(E(\sigma.a.\sigma_u)), \epsilon, Q')$ does not hold, there exists $\sigma'_u \in \Sigma_\mathrm{u}^*$ such that $\mathrm{Reach}(E(\sigma.a.\sigma_u))$ after $\sigma'_u \notin F$. Since E is compliant, and $E(\sigma.a.\sigma_u)_{|\Sigma_\mathrm{c}} = \sigma_{|\Sigma_\mathrm{c}}$ (since $\sigma'^{-1}_{sn}.\sigma_b = \epsilon$), then $E(\sigma.a.\sigma_u.\sigma'_u) = E(\sigma.a.\sigma_u).\sigma'_u \notin F$, meaning that $E(\sigma.a.\sigma_u.\sigma'_u) \not\models \varphi$. Otherwise, $\mathrm{Reach}(E(\sigma.a.\sigma_u)) \in Q'$, and since $\mathrm{Safe}_\mathrm{int}(\mathrm{Reach}(E(\sigma.a.\sigma_u)), \sigma'^{-1}_{sn}.\sigma_b, Q')$ does not hold, this means that there exists $(q, \sigma'^{-1}_{sn}.\sigma_b) \in Q'$ such that $q \notin F$. Since $(q, \sigma'^{-1}_{sn}.\sigma_b) \in Q'$, there exists $\sigma'_u \preccurlyeq \sigma_u$ such that $\mathrm{Reach}(E(\sigma.a.\sigma'_u)) = q$. Thus, $\mathrm{Reach}(\sigma.a.\sigma'_u) \notin F$, meaning that $\sigma'_u \in \Sigma_\mathrm{u}^*$ is such that $E(\sigma.a.\sigma'_u) \not\models \varphi$.

Thus, in call cases, there exists $\sigma_u \in \Sigma_\mathrm{u}^*$ such that $E(\sigma.a.\sigma_u) \not\models \varphi$.

—*Case 2:* $a \in \Sigma_\mathrm{c}$. The proof is the same as in the case where $a \in \Sigma_\mathrm{u}$, but with $S = \{w \preccurlyeq \sigma_b.a \mid \sigma_s.w \models \varphi \wedge \mathrm{Safe}(\mathrm{Reach}(\sigma_s.w), w^{-1}.(\sigma_b.a))\}$, and replacing occurrences of "$\sigma_s.a$" by "σ_s", and occurrences of "σ_b" by "$\sigma_b.a$".

Thus, if E is an enforcement function such that there exists $\sigma \in \mathrm{Pre}(\varphi)$, and $a \in \Sigma$ so that $\mathrm{compliant}(E, \Sigma_\mathrm{u}, \Sigma_\mathrm{c}) \wedge E(\sigma) = \mathrm{E}_\varphi(\sigma) \wedge |E(\sigma.a)| > |\mathrm{E}_\varphi(\sigma.a)|$, then there exists $\sigma_u \in \Sigma_\mathrm{u}^*$ such that $E(\sigma.a.\sigma_u) \not\models \varphi$.

This means that E_φ is optimal in $\mathrm{Pre}(\varphi)$. □

Proposition 5. The output of the enforcement monitor $\mathcal{E}$ for input σ is $\mathrm{E}_\varphi(\sigma)$.

Proof. Let us introduce some notation for this proof: for a word $w \in \Gamma^{\mathcal{E}*}$, we note $\mathrm{input}(w) = \Pi_1(w(1)).\Pi_1(w(2)) \ldots \Pi_1(w(|w|))$, the word obtained by concatenating the first members (the inputs) of w. In a similar way, we note $\mathrm{output}(w) = \Pi_3(w(1)).\Pi_3(w(2)) \ldots \Pi_3(w(|w|))$, the word obtained by concatenating all the third members (outputs) of w. Since all configurations are not reachable from $c_0^\mathcal{E}$, for $w \in \Gamma^{\mathcal{E}*}$, we note $\mathrm{Reach}(w) = c$ whenever $c_0^\mathcal{E} \xhookrightarrow{w}_\mathcal{E} c$, and $\mathrm{Reach}(w) = \bot$ if such a c does not exist. We also define the Rules function as follows:

$$\mathrm{Rules} : \begin{cases} \Sigma^* & \to & \Gamma^{\mathcal{E}*} \\ \sigma & \mapsto & \max_\preccurlyeq(\{w \in \Gamma^{\mathcal{E}*} \mid \mathrm{input}(w) = \sigma \wedge \mathrm{Reach}(w) \neq \bot\}) \end{cases}$$

For a word $\sigma \in \Sigma^*$, $\mathrm{Rules}(\sigma)$ is the trace of the longest valid run in $\mathcal{E}$, i.e. the sequence of all the rules that can be applied with input σ. We then extend the definition of output to words in Σ^*: for $\sigma \in \Sigma^*$, $\mathrm{output}(\sigma) = \mathrm{output}(\mathrm{Rules}(\sigma))$. We also note ϵ the empty word of Σ^*, and $\epsilon^\mathcal{E}$ the empty word of $\Gamma^{\mathcal{E}*}$.

For $\sigma \in \Sigma^*$, let $\mathrm{P}(\sigma)$ be the predicate: "$\mathrm{E}_\varphi(\sigma) = \mathrm{output}(\sigma) \wedge ((\sigma_s, \sigma_c) = \mathrm{store}_\varphi(\sigma) \wedge \mathrm{Reach}(\mathrm{Rules}(\sigma)) = \langle q, \sigma_c^\mathcal{E}\rangle) \implies (q = \mathrm{Reach}(\sigma_s) \wedge \sigma_c = \sigma_c^\mathcal{E})$".

Let us prove that for all $\sigma \in \Sigma^*$, $\mathrm{P}(\sigma)$ holds.

—*Induction basis*: $\mathrm{E}_\varphi(\epsilon) = \epsilon = \mathrm{output}(\epsilon)$. Moreover, $\mathrm{store}_\varphi(\epsilon) = (\epsilon, \epsilon)$, and $\mathrm{Reach}(\epsilon^\mathcal{E}) = c_0^\mathcal{E}$. Therefore, as $c_0^\mathcal{E} = \langle q_0, \epsilon\rangle$, $\mathrm{P}(\epsilon)$ holds, because $\mathrm{Reach}(\epsilon) = q_0$.

—*Induction step*: Let us suppose now that for some $\sigma \in \Sigma^*$, $\mathrm{P}(\sigma)$ holds. Let us consider $(\sigma_s, \sigma_c) = \mathrm{store}_\varphi(\sigma)$, $q = \mathrm{Reach}(\sigma_s)$, $a \in \Sigma$, and $(\sigma_t, \sigma_d) = \mathrm{store}_\varphi(\sigma.a)$. Let us prove that $\mathrm{P}(\sigma.a)$ holds.

Since $\mathrm{P}(\sigma)$ holds, $\mathrm{Reach}(\mathrm{Rules}(w)) = \langle q, \sigma_c\rangle$ and $\sigma_s = \mathrm{output}(\sigma)$.

–*Case 1:* $a \in \Sigma_\mathrm{u}$. Then, considering σ'_s as defined in Definition 7, $\sigma_t = \sigma_s.a.\sigma'_s$. Moreover, $a \in \Sigma_\mathrm{u}$, thus rule pass-uncont can be applied: let us consider $q' = q\mathrm{after}a$. Then $\langle q, \sigma_c\rangle \xrightarrow{a/\text{ pass-uncont}(a)/a}_\mathcal{E} \langle q', \sigma_c\rangle$. If $\sigma'_s \neq \epsilon$, then $\sigma_t = \sigma_s.a.\sigma'_s \models \varphi$, and if $q'' = \mathrm{Reach}(\sigma_t)$, then $\mathrm{Safe}(q'', \sigma'^{-1}_s.\sigma_c)$

holds, meaning that rule $\mathrm{dump}(\sigma'_s)$ can be applied, leading to $\langle q', \sigma_c\rangle$ after $\epsilon/\,\mathrm{dump}(\sigma'_s)/\sigma'_s = \langle q'', {\sigma'_s}^{-1}.\sigma_c\rangle = \langle \mathrm{Reach}(\sigma_t), \sigma_d\rangle$. Moreover, $\mathrm{output}(\sigma.a) = \mathrm{output}(\sigma).a.\sigma'_s$. Since $\mathrm{P}(\sigma)$ holds, $\mathrm{output}(\sigma) = \sigma_s$, thus $\mathrm{output}(\sigma.a) = \sigma_s.a.\sigma'_s = \sigma_t$. Thus, if $\sigma'_s \neq \epsilon$, $\mathrm{P}(\sigma.a)$ holds. Otherwise, $\sigma'_s = \epsilon$, and then it is impossible to apply dump rule, because it is impossible to reach a safe state with a prefix of σ_c that is not ϵ. Thus, $\mathrm{output}(\sigma.a) = \mathrm{output}(\sigma).a = \sigma_s.a = \sigma_t$, and $\mathrm{Reach}(\mathrm{Rules}(\sigma.a)) = \langle q', \sigma_c\rangle$, with $q' = \mathrm{Reach}(\sigma_t)$, and $\sigma_d = \sigma_c$. Thus, $\mathrm{P}(\sigma.a)$ holds.
Thus, if $a \in \Sigma_\mathrm{u}$, $\mathrm{P}(\sigma.a)$ holds.

– *Case 2:* $a \in \Sigma_\mathrm{c}$. Then, considering σ''_s as defined in Definition 7, $\sigma_t = \sigma_s.\sigma''_s$. Since $a \in \Sigma_\mathrm{c}$, it is possible to apply the store-cont rule, and $\langle q, \sigma_c\rangle$ after $a/\,\mathrm{store\text{-}cont}(a)/\epsilon = \langle q, \sigma_c.a\rangle$. Then, if $\sigma''_s \neq \epsilon$, $q' = q\mathrm{after}\sigma''_s$ is such that $q' \in F \wedge \mathrm{Safe}(q', {\sigma''_s}^{-1}.(\sigma_c.a))$, meaning that rule $\mathrm{dump}(\sigma''_s)$ can be applied. Let us consider $\langle q', {\sigma''_s}^{-1}.(\sigma_c.a)\rangle = \langle q, \sigma_c.a\rangle$ after $\epsilon/\,\mathrm{dump}(\sigma''_s)/\sigma''_s$. Then, $q' = \mathrm{Reach}(\sigma_t)$, and ${\sigma''_s}^{-1}.(\sigma_c.a) = \sigma_d$. Moreover, $\mathrm{output}(\sigma.a) = \mathrm{output}(\sigma).\sigma''_s = \sigma_s.\sigma''_s = \sigma_t$. Thus, if $\sigma''_s \neq \epsilon$, $\mathrm{P}(\sigma.a)$ holds.
Otherwise, $\sigma''_s = \epsilon$, and no rule can be applied anymore, thus $\mathrm{Reach}(\mathrm{Rules}(\sigma.a)) = \langle q, \sigma_c.a\rangle$, where $q = \mathrm{Reach}(\sigma_t)$, and $\sigma_c.a = {\sigma''_s}^{-1}.(\sigma_c.a) = \sigma_d$, and $\mathrm{output}(\sigma.a) = \mathrm{output}(\sigma) = \sigma_s = \sigma_t$. Thus, if $\sigma''_s = \epsilon$, $\mathrm{P}(\sigma.a)$ holds. Thus, if $a \in \Sigma_\mathrm{c}$, $\mathrm{P}(\sigma.a)$ holds.

This means that $\mathrm{P}(\sigma) \implies \mathrm{P}(\sigma.a)$.

Thus, by induction on σ, for all $\sigma \in \Sigma^*$, $\mathrm{P}(\sigma)$ holds. In particular, for all $\sigma \in \Sigma^*$, $\mathrm{E}_\varphi(\sigma) = \mathrm{output}(\sigma)$. □

A.2. *Proofs for the Timed Setting*

Proposition 6. E_φ as defined in Definition 15 is an enforcement function, as per Definition 10.

Proof. For $\sigma \in \mathrm{tw}(\Sigma)$, let $\mathrm{P}(\sigma)$ be the predicate: "$\forall t \in \mathbb{R}_{\geq 0}, \forall t' \geq t, \mathrm{E}_\varphi(\sigma, t) \preccurlyeq \mathrm{E}_\varphi(\sigma, t')$". Let us show by induction that for all $\sigma \in \mathrm{tw}(\Sigma)$, $\mathrm{P}(\sigma)$ holds.

— *Induction basis:* $\sigma = \epsilon$. Then, let us consider $t \in \mathbb{R}_{\geq 0}$, and $t' \geq t$. Then, $\mathrm{E}_\varphi(\epsilon, t) = \epsilon \preccurlyeq \epsilon = \mathrm{E}_\varphi(\epsilon, t')$. Thus, $\mathrm{P}(\epsilon)$ holds.

— *Induction step:* let us suppose that, for $\sigma \in \mathrm{tw}(\Sigma)$, $\mathrm{P}(\sigma)$ holds. Let us consider (t'', a) such that $\sigma.(t'', a) \in \mathrm{tw}(\Sigma)$, $t \in \mathbb{R}_{\geq 0}$, and $t' \geq t$.

– If $t \geq t''$, then let us consider $(\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t'')$, $(\sigma_{t1}, \sigma_{d1}, \sigma_{e1}) = \mathrm{store}_\varphi(\sigma.(t'', a), t)$, and $(\sigma_{t2}, \sigma_{d2}, \sigma_{e2}) = \mathrm{store}_\varphi(\sigma.(t'', a), t')$. Then, $\mathrm{E}_\varphi(\sigma.(t'', a), t) = \sigma_{t1}$ and $\mathrm{E}_\varphi(\sigma.(t'', a), t') = \sigma_{t2}$.

• If $a \in \Sigma_\mathrm{u}$, then considering t_1 as defined in Definition 15, $t_1 = \min(\{t_0 \in \mathbb{R}_{\geq 0} \mid t_0 \geq t'' \wedge \mathrm{G}(\mathrm{Reach}(\sigma_s.(t'', a), t_0), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t'')).\sigma_c) \neq \emptyset\})$. Then, $\sigma_{d1} = \min_{\mathrm{lex}}(\max_{\preccurlyeq}(\mathrm{G}(\mathrm{Reach}(\sigma_s.(t'', a), \min(\{t, t_1\})), \Pi_\Sigma(\mathrm{nob}$ $\{\epsilon\}))+_\mathrm{t} t'$, and $\sigma_{d2} = \min_{\mathrm{lex}}(\max_{\preccurlyeq}(\mathrm{G}(\mathrm{Reach}(\sigma_s.(t'', a), \min(\{t', t_1\})), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t'')).\sigma_c) \cup$ $\{\epsilon\})) +_\mathrm{t} t'$.

· *Case 1:* $t \geq t_1$. Since $t' \geq t$, then $t' \geq t_1$, thus $\min(\{t', t_1\}) = \min(\{t, t_1\}) = t_1$, thus $\sigma_{d1} = \sigma_{d2}$. It follows that:
$\sigma_{t1} = \sigma_s.(t'', a).\,\mathrm{obs}(\sigma_{d1}, t) \preccurlyeq \sigma_s.(t'', a).\,\mathrm{obs}(\sigma_{d1}, t') = \sigma_s.(t'', a).\,\mathrm{obs}(\sigma_{d2}, t') = \sigma_{t2}$.

· *Case 2:* $t < t_1$. Then, $\min(\{t, t_1\}) = t$. Since $t < t_1$, by definition of t_1, this means that $\mathrm{G}(\mathrm{Reach}(\sigma_s.(t'', a), t), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t'')).\sigma_c) = \emptyset$, and thus $\sigma_{d1} = \epsilon$. Since $\sigma_{d1} = \epsilon$, $\sigma_{t1} = \sigma_s.(t'', a) \preccurlyeq \sigma_s.(t'', a).\,\mathrm{obs}(\sigma_{d2}, t') = \sigma_{t2}$.

Thus, if $t' \geq t \geq t''$ and $a \in \Sigma_\mathrm{u}$, $\mathrm{P}(\sigma) \implies \mathrm{E}_\varphi(\sigma.(t'', a), t) \preccurlyeq \mathrm{E}_\varphi(\sigma.(t'', a), t')$.

• Otherwise, $a \in \Sigma_\mathrm{c}$, and then considering t_2 as defined in Definition 15, $t_2 = \min(\{t_0 \in \mathbb{R}_{\geq 0} \mid$

$t_0 \geq t'' \wedge \mathrm{G}(\mathrm{Reach}(\sigma_s, t_0), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t'')).\sigma_c.a) \neq \emptyset\}$). Then, $\sigma_{d1} = \min_{\mathrm{lex}}(\max_{\preccurlyeq}(\mathrm{G}(\mathrm{Reach}(\sigma_s,\ \min(\{t,\ t_2\})),\ \Pi_\Sigma(\mathrm{nobs}(\sigma_b,\ t'')) \,.\, \sigma_c \,.\, a) \cup \{\epsilon\}))+_{\mathrm{t}} \mathrm{time}(\sigma_s)$, and:
$\sigma_{d2} = \min_{\mathrm{lex}}(\max_{\preccurlyeq}(\mathrm{G}(\mathrm{Reach}(\sigma_s, \min(\{t', t_2\})), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t'')).\sigma_c.a) \cup \{\epsilon\}))+_{\mathrm{t}} \mathrm{time}(\sigma_s)$.

·*Case 1:* $t \geq t_2$. Since $t' \geq t$, $t' \geq t_2$, meaning that $\min(\{t, t2\}) = \min(\{t', t_2\}) = t_2$, and thus $\sigma_{d1} = \sigma_{d2}$. It follows that $\sigma_{t1} = \sigma_s.\mathrm{obs}(\sigma_{d1}, t)) \preccurlyeq \sigma_s.\mathrm{obs}(\sigma_{d1}, t') = \sigma_s.\mathrm{obs}(\sigma_{d2}, t') = \sigma_{t2}$.

·*Case 2:* $t < t_2$. Then, $\mathrm{G}(\mathrm{Reach}(\sigma_s, \min(\{t, t_2\})), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t'')).\sigma_c.a) = \emptyset$, meaning that $\sigma_{d1} = \epsilon$. Thus, $\sigma_{t1} = \sigma_s \preccurlyeq \sigma_s.\mathrm{obs}(\sigma_{d2}, t') = \sigma_{t2}$.

Thus, if $t' \geq t \geq t''$ and $a \in \Sigma_{\mathrm{c}}$, $\mathrm{P}(\sigma) \implies \mathrm{E}_\varphi(\sigma.(t'', a), t) \preccurlyeq \mathrm{E}_\varphi(\sigma.(t'', a), t')$.

Therefore, if $t' \geq t \geq t''$, for all $a \in \Sigma$, $\mathrm{P}(\sigma) \implies \mathrm{E}_\varphi(\sigma.(t'', a), t) \preccurlyeq \mathrm{E}_\varphi(\sigma.(t'', a), t')$.

–If $t' < t''$, then $t < t''$, and $\mathrm{obs}(\sigma.(t'', a), t) = \mathrm{obs}(\sigma, t)$, and $\mathrm{obs}(\sigma.(t'', a), t') = \mathrm{obs}(\sigma, t')$. Thus, $\mathrm{E}_\varphi(\sigma.(t'', a), t) = \mathrm{store}_\varphi(\mathrm{obs}(\sigma.(t'', a), t), t) = \mathrm{store}_\varphi(\mathrm{obs}(\sigma, t), t) = \mathrm{E}_\varphi(\sigma, t)$, and $\mathrm{E}_\varphi(\sigma.(t'', a), t') = \mathrm{store}_\varphi(\mathrm{obs}(\sigma.(t'', a), t'), t') = \mathrm{store}_\varphi(\mathrm{obs}(\sigma, t'), t') = \mathrm{E}_\varphi(\sigma, t')$. Since $\mathrm{P}(\sigma)$ holds, then $\mathrm{E}_\varphi(\sigma.(t'', a), t) = \mathrm{E}_\varphi(\sigma, t) \preccurlyeq \mathrm{E}_\varphi(\sigma, t') = \mathrm{E}_\varphi(\sigma.(t'', a), t')$.

–If $t < t'' \leq t'$, then $\mathrm{obs}(\sigma.(t'', a), t) = \mathrm{obs}(\sigma, t)$. Since $\mathrm{P}(\sigma)$ holds, then $\mathrm{E}_\varphi(\sigma, t) \preccurlyeq \mathrm{E}_\varphi(\sigma, t'')$. Let $(\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t'')$, and $(\sigma_t, \sigma_d, \sigma_e) = \mathrm{store}_\varphi(\sigma.(t'', a), t')$. Then, $\sigma_t = \sigma_s.(t'', a).\mathrm{obs}(\sigma_e, t')$ if $a \in \Sigma_{\mathrm{u}}$, and $\sigma_t = \sigma_s.\mathrm{obs}(\sigma_e, t')$ if $a \in \Sigma_{\mathrm{c}}$. In both cases, $\sigma_s \preccurlyeq \sigma_t$. This means that $\mathrm{E}_\varphi(\sigma, t'') \preccurlyeq \mathrm{E}_\varphi(\sigma.(t'', a), t')$. Thus, $\mathrm{E}_\varphi(\sigma.(t'', a), t) = \mathrm{E}_\varphi(\sigma, t) \preccurlyeq \mathrm{E}_\varphi(\sigma, t'') \preccurlyeq \mathrm{E}_\varphi(\sigma.(t'', a), t')$. Thus, if $t < t'' \leq t'$, then $\mathrm{P}(\sigma) \implies \mathrm{E}_\varphi(\sigma.(t'', a), t) \preccurlyeq \mathrm{E}_\varphi(\sigma.(t'', a), t')$.

Consequently, in all cases, if $t \leq t'$, then $\mathrm{P}(\sigma) \implies \mathrm{E}_\varphi(\sigma.(t'', a), t) \preccurlyeq \mathrm{E}_\varphi(\sigma.(t'', a), t')$. Finally, $\mathrm{P}(\sigma) \implies \mathrm{P}(\sigma.(t'', a))$.

By induction, for all $\sigma \in \mathrm{tw}(\Sigma)$, $\mathrm{P}(\sigma)$ holds. Thus, for all $\sigma \in \mathrm{tw}(\Sigma)$, for all $t \in \mathbb{R}_{\geq 0}$, for all $t' \geq t$, $\mathrm{E}_\varphi(\sigma, t) \preccurlyeq \mathrm{E}_\varphi(\sigma, t')$.

Now, let us consider $\sigma \in \mathrm{tw}(\Sigma)$, and (t, a) such that $\sigma.(t, a) \in \mathrm{tw}(\Sigma)$. Then, if $(\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t)$, and $(\sigma_t, \sigma_d, \sigma_e) = \mathrm{store}_\varphi(\sigma.(t, a), t)$, then either $\sigma_t = \sigma_s.(t, a).\sigma'_s$, or $\sigma_t = \sigma_s.\sigma''_s$, whether a is controllable or uncontrollable respectively, where σ'_s and σ''_s are defined in Definition 15. In both cases, $\sigma_s \preccurlyeq \sigma_t$. Thus, $\mathrm{E}_\varphi(\sigma, t) = \Pi_1(\mathrm{store}_\varphi(\mathrm{obs}(\sigma, t), t)) = \sigma_s \preccurlyeq \sigma_t = \Pi_1(\mathrm{store}_\varphi(\mathrm{obs}(\sigma.(t, a), t))) = \mathrm{E}_\varphi(\sigma.(t, a), t)$. This holds because, since $\sigma.(t, a) \in \mathrm{tw}(\Sigma)$, $\mathrm{time}(\sigma) \leq t$, thus $\mathrm{obs}(\sigma, t) = \sigma$.

Thus, for all $\sigma \in \mathrm{tw}(\Sigma)$, for all $t \in \mathbb{R}_{\geq 0}$ and $t' \geq t$, $\mathrm{E}_\varphi(\sigma, t) \preccurlyeq \mathrm{E}_\varphi(\sigma, t')$ and $\mathrm{E}_\varphi(\sigma, t) \preccurlyeq \mathrm{E}_\varphi(\sigma.(t, a), t)$.
This means that E_φ is an enforcement function. □

Lemma 2. $\forall t \in \mathbb{R}_{\geq 0}, \forall \sigma \in \mathrm{tw}(\Sigma), (\sigma \notin \mathrm{Pre}(\varphi, t) \wedge (\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t)) \implies (\sigma_s = \sigma_{|\Sigma_{\mathrm{u}}} \wedge \sigma_b = \epsilon \wedge \sigma_c = \Pi_\Sigma(\sigma_{|\Sigma_{\mathrm{c}}}))$.

Proof. For $\sigma \in \mathrm{tw}(\Sigma)$, let $\mathrm{P}(\sigma)$ be the predicate "$\forall t \geq \mathrm{time}(\sigma), (\sigma \notin \mathrm{Pre}(\varphi, t) \wedge (\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t)) \implies (\sigma_s = \sigma_{|\Sigma_{\mathrm{u}}} \wedge \sigma_b = \epsilon \wedge \sigma_c = \Pi_\Sigma(\sigma_{|\Sigma_{\mathrm{c}}}))$". Let us prove by induction that for all $\sigma \in \mathrm{tw}(\Sigma)$, $\mathrm{P}(\sigma)$ holds.

—*Induction basis:* for $\sigma = \epsilon$, let us consider $t \in \mathbb{R}_{\geq 0}$. Then, $\mathrm{store}_\varphi(\epsilon, t) = (\epsilon, \epsilon, \epsilon)$. Considering that $\epsilon \in \mathrm{tw}(\Sigma_{\mathrm{u}})$, and $\epsilon = \Pi_\Sigma(\epsilon_{|\Sigma_{\mathrm{c}}})$, $\mathrm{P}(\epsilon)$ trivially holds (whether $\epsilon \in \mathrm{P}(\varphi, t)$ or not).

—*Induction step:* suppose that for $\sigma \in \mathrm{tw}(\Sigma)$, $\mathrm{P}(\sigma)$ holds. Let us consider (t', a) such that $\sigma.(t', a) \in \mathrm{tw}(\Sigma)$, and $t \geq t'$. Let us also consider $(\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t')$ and $(\sigma_t, \sigma_d, \sigma_e) = \mathrm{store}_\varphi(\sigma.(t', a), t)$. Then, if $\sigma.(t', a) \in \mathrm{Pre}(\varphi, t)$, $\mathrm{P}(\sigma.(t', a))$ trivially holds. Thus, let us suppose that $\sigma.(t', a) \notin$

$\mathrm{Pre}(\varphi, t)$. Since $\sigma \preccurlyeq \sigma.(t', a)$ and $t \geq t'$, it follows that $\sigma \notin \mathrm{Pre}(\varphi, t')$. By induction hypothesis, this means that $\sigma_s \in \mathrm{tw}(\Sigma_\mathrm{u})$, $\sigma_b = \epsilon$, and $\sigma_c = \Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}})$. Then, since $\sigma.(t', a) \notin \mathrm{Pre}(\varphi, t)$, $\mathrm{E}_1(\sigma.(t', a), t) = \emptyset$, where E_1 is defined in Definition 16. Then, following the definition of E_1:

$$\mathrm{E}_1(\sigma.(t', a), t) = \begin{array}{l} \{w \in \mathrm{tw}(\Sigma) \mid (\sigma.(t', a))_{|\Sigma_\mathrm{u}} \preccurlyeq w \wedge \\ (\sigma.(t', a))_{|\Sigma_\mathrm{u}} = w_{|\Sigma_\mathrm{u}} \wedge (\sigma.(t', a))_{|\Sigma_\mathrm{c}} \preccurlyeq_d w_{|\Sigma_\mathrm{c}} \wedge \\ \mathrm{Reach}(w) \in F_G \wedge \\ (w_{|\Sigma_\mathrm{c}} \neq \epsilon \implies \mathrm{date}(w_{|\Sigma_\mathrm{c}}(1)) \geq t) \wedge \\ \forall t'' \geq t, \mathrm{Safe}(\mathrm{Reach}(w, t''), \Pi_\Sigma(\mathrm{obs}(w, t''))^{-1}.\Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}}))\}. \end{array} \quad (1)$$

– *Case 1:* $a \in \Sigma_\mathrm{u}$. Then, considering that $(\sigma.(t', a))_{|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}}.(t', a)$, $(\sigma.(t', a))_{|\Sigma_\mathrm{c}} = \sigma_{|\Sigma_\mathrm{c}}$, and replacing by σ_s, σ_b, and σ_c when possible, (1) becomes:

$$\mathrm{E}_1(\sigma.(t', a), t) = \begin{array}{l} \{w \in \mathrm{tw}(\Sigma) \mid \sigma_s.(t', a) \preccurlyeq w \wedge w_{|\Sigma_\mathrm{u}} = \sigma_s.(t', a) \wedge \\ \Pi_\Sigma(w_{|\Sigma_\mathrm{c}}) \preccurlyeq \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c \wedge \\ \mathrm{Reach}(w) \in F_G \wedge (w_{|\Sigma_\mathrm{c}} \neq \epsilon \implies \mathrm{date}(w_{|\Sigma_\mathrm{c}}(1)) \geq t) \wedge \forall t'' \geq t, \\ \mathrm{Safe}(\mathrm{Reach}(w, t''), \Pi_\Sigma(\mathrm{obs}(w_{|\Sigma_\mathrm{c}}, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c))\}. \end{array}$$

Now, $\mathrm{E}_1(\sigma.(t', a), t) = \emptyset$, and since $t \geq t'$, this means that the set $((\sigma_s.(t', a))^{-1}.\,\mathrm{E}_1(\sigma.(t', a), t)) -_\mathrm{t} t$, is empty too, thus:

$$\begin{array}{l} \{w \in \mathrm{tw}(\Sigma) \mid \Pi_\Sigma(w) \preccurlyeq \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c \wedge \\ (\mathrm{Reach}(\sigma_s.(t', a), t) \text{ after } w) \in F_G \wedge \forall t'' \in \mathbb{R}_{\geq 0}, \\ \mathrm{Safe}(\mathrm{Reach}(\sigma_s.(t', a), t) \text{ after } (w, t''), \Pi_\Sigma(\mathrm{obs}(w, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c))\} \end{array} = \emptyset.$$

This means that $\mathrm{G}(\mathrm{Reach}(\sigma_s.(t', a), t), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c) = \emptyset$, where G is defined in Definition 15. Thus, $\sigma_d = \epsilon$. It follows that $\sigma_t = \sigma_s.(t', a).\,\mathrm{obs}(\sigma_d, t) = \sigma_s.(t', a) = (\sigma.(t', a))_{|\Sigma_\mathrm{u}}$, and $\sigma_e = \Pi_\Sigma((\sigma.(t', a))_{|\Sigma_\mathrm{c}})$.
Thus, $\mathrm{P}(\sigma.(t', a))$ holds when $a \in \Sigma_\mathrm{u}$.

– *Case 2:* $a \in \Sigma_\mathrm{c}$. Then, $(\sigma.(t', a))_{|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}}$, and $(\sigma.(t', a))_{|\Sigma_\mathrm{c}} = \sigma_{|\Sigma_\mathrm{c}}.(t', a)$. In a similar way than previously, (1) becomes:

$$\mathrm{E}_1(\sigma.(t', a), t) = \begin{array}{l} \{w \in \mathrm{tw}(\sigma) \mid \sigma_s \preccurlyeq w \wedge w_{|\Sigma_\mathrm{u}} = \sigma_s \wedge \\ \Pi_\Sigma(w_{|\Sigma_\mathrm{c}}) \preccurlyeq \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c.a \wedge \\ \mathrm{Reach}(w) \in F_G \wedge (w_{|\Sigma_\mathrm{c}} \neq \epsilon \implies \mathrm{date}(w_{|\Sigma_\mathrm{c}}(1)) \geq t) \wedge \forall t'' \geq t, \\ \mathrm{Safe}(\mathrm{Reach}(w, t''), \Pi_\Sigma(\mathrm{obs}(w_{|\Sigma_\mathrm{c}}, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c.a))\}. \end{array}$$

Then, again, since $\mathrm{E}_1(\sigma.(t', a), t) = \emptyset$, the set $(\sigma_s^{-1}.\,\mathrm{E}_1(\sigma.(t', a), t)) -_\mathrm{t} t$ is empty too:

$$\begin{array}{l} \{w \in \mathrm{tw}\,\Sigma) \mid \Pi_\Sigma(w) \preccurlyeq \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c.a \wedge \\ \mathrm{Reach}(\sigma_s, t) \text{ after } w \in F_G \wedge \forall t'' \in \mathbb{R}_{\geq 0}, \\ \mathrm{Safe}(\mathrm{Reach}(\sigma_s, t) \text{ after } (w, t''), \Pi_\Sigma(\mathrm{obs}(w, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c.a))\} \end{array} = \emptyset.$$

Thus, $\mathrm{G}(\mathrm{Reach}(\sigma_s, t), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c.a) = \emptyset$. This means that $\sigma_d = \epsilon$, thus $\sigma_t = \sigma_s.\,\mathrm{obs}(\sigma_d, t) = \sigma_s = \sigma_{|\Sigma_\mathrm{u}} = (\sigma.(t', a))_{|\Sigma_\mathrm{u}}$, and $\sigma_e = \Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}}).a = \Pi_\Sigma((\sigma.(t', a))_{|\Sigma_\mathrm{c}})$. It follows that $\mathrm{P}(\sigma.(t', a))$ holds.

Thus, $\mathrm{P}(\sigma) \implies \mathrm{P}(\sigma.(t', a))$.

By induction, for all $\sigma \in \mathrm{tw}(\Sigma)$, $\mathrm{P}(\sigma)$ holds. Thus, for all $\sigma \in \mathrm{tw}(\Sigma)$, for all $t \in \mathbb{R}_{\geq 0}$, if $(\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t)$ and $(\sigma, t) \notin \mathrm{Pre}(\varphi)$, then $\sigma_s = \sigma_{|\Sigma_\mathrm{u}}$, $\sigma_b = \epsilon$, and $\sigma_c = \Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}})$. □

Lemma 3. For $q \in Q$, $\sigma \in \Sigma_\mathrm{c}^*$, $u \in \Sigma_\mathrm{u}$, $w \in \mathrm{tw}(\Sigma)$:
$(\mathrm{Safe}_\mathrm{int}(q, \sigma, \emptyset) \wedge \forall t \in \mathbb{R}_{\geq 0}, \mathrm{Safe}_\mathrm{int}(q \text{ after } ((0, u).w, t), \Pi_\Sigma(\mathrm{obs}(w, t))^{-1}.\sigma, \{(q, \sigma)\}))$
$\implies \forall t \in \mathbb{R}_{\geq 0}, \mathrm{Safe}_\mathrm{int}(q \text{ after } ((0, u).w, t), \Pi_\Sigma(\mathrm{obs}(w, t))^{-1}.\sigma, \emptyset)$.

Proof. The proof is the same as the one for lemma 1. The only difference is that it is necessary to consider the value of q after $((0,u).w)$ at each possible date. □

Proposition 7. E_φ is sound with respect to φ in $\mathrm{Pre}(\varphi)$ as per Definition 11.

Proof. Notation from Definition 15 is to be used in this proof:

$$
\begin{aligned}
\mathrm{G}(q,\sigma_1) &= \{w \in \mathrm{tw}(\Sigma) \mid \Pi_\Sigma(w) \preccurlyeq \sigma_1 \wedge (q \text{ after } w) \in F_G \wedge \\
&\qquad \forall t'' \in \mathbb{R}_{\geq 0}, \mathrm{Safe}(q \text{ after } (w,t''), \Pi_\Sigma(\mathrm{obs}(w,t''))^{-1}.\sigma_1)\}, \\
\kappa_\varphi(q,\sigma_1) &= \min_{\text{lex}}(\max_{\preccurlyeq}(\mathrm{G}(q,\sigma_1) \cup \{\epsilon\})) \\
\mathit{buffer}_c &= \Pi_\Sigma(\mathrm{nobs}(\sigma_b,t')).\sigma_c \\
t_1 &= \begin{array}{l}\min(\{t'' \in \mathbb{R}_{\geq 0} \mid t'' \geq t' \wedge \\ \mathrm{G}(\mathrm{Reach}(\sigma_s.(t',a),t''), \mathit{buffer}_c) \neq \emptyset\} \cup \{+\infty\}),\end{array} \\
\sigma'_b &= \kappa_\varphi(\mathrm{Reach}(\sigma_s.(t',a), \min(\{t,t_1\})), \mathit{buffer}_c) +_{\mathrm{t}} \min(\{t,t_1\}), \\
\sigma'_c &= \Pi_\Sigma(\sigma'_b)^{-1}.\mathit{buffer}_c, \\
t_2 &= \begin{array}{l}\min(\{t'' \in \mathbb{R}_{\geq 0} \mid t'' \geq t' \wedge \\ \mathrm{G}(\mathrm{Reach}(\sigma_s,t''), \mathit{buffer}_c.a) \neq \emptyset\} \cup \{+\infty\}),\end{array} \\
\sigma''_b &= \kappa_\varphi(\mathrm{Reach}(\sigma_s, \min(\{t,t_2\})), \mathit{buffer}_c.a) +_{\mathrm{t}} \min(\{t,t_2\}), \\
\sigma''_c &= \Pi_\Sigma(\sigma''_b)^{-1}.(\mathit{buffer}_c.a).
\end{aligned}
$$

For $\sigma \in \mathrm{tw}(\Sigma)$, and $t \geq \mathrm{time}(\sigma)$, let $\mathrm{P}(\sigma,t)$ be the predicate "$(\sigma \in \mathrm{Pre}(\varphi,t) \wedge (\sigma_s,\sigma_b,\sigma_c) = \mathrm{store}_\varphi(\sigma,t)) \implies (\mathrm{E}_\varphi(\sigma) \models \varphi \wedge \mathrm{nobs}(\sigma_b,t) -_{\mathrm{t}} t \in \mathrm{G}(\mathrm{Reach}(\sigma_s,t), \Pi_\Sigma(\mathrm{nobs}(\sigma_b,t)).\sigma_c))$. Let also $\mathrm{P}(\sigma)$ be the predicate: "$\forall t \geq \mathrm{time}(\sigma), \mathrm{P}(\sigma,t)$ holds". Let us show that for all $\sigma \in \mathrm{tw}(\Sigma)$, $\mathrm{P}(\sigma)$ holds.

—*Induction basis:* for $\sigma = \epsilon$, let us consider $t \in \mathbb{R}_{\geq 0}$.

– *Case 1:* $\epsilon \notin \mathrm{Pre}(\varphi,t)$. Then, $\mathrm{P}(\epsilon)$ trivially holds.

– *Case 2:* $\epsilon \in \mathrm{Pre}(\varphi,t)$. Then, there exists $t' \leq t$ such that $\mathrm{E}_1(\mathrm{obs}(\epsilon,t'),t') \neq \emptyset$, meaning that $\mathrm{E}_1(\epsilon,t') \neq \emptyset$. Thus, following the definition of $\mathrm{E}_1(\epsilon,t')$, $\mathrm{Reach}(\epsilon) \in F_G$, and for all $t'' \geq t'$, $\mathrm{Safe}(\mathrm{Reach}(\epsilon,t''),\epsilon)$ holds. Since $\mathrm{E}_\varphi(\epsilon) = \epsilon$, and $\mathrm{Reach}(\epsilon) \in F_G$, $\mathrm{E}_\varphi(\epsilon) \models \varphi$. Moreover, ϵ satisfies $\Pi_\Sigma(\epsilon) \preccurlyeq \epsilon \wedge (\mathrm{Reach}(\epsilon) \text{ after } \epsilon) \in F_G \wedge \forall t'' \geq t', \mathrm{Safe}(\mathrm{Reach}(\epsilon,t''),\epsilon)$, meaning that $\epsilon \in \mathrm{G}(\mathrm{Reach}(\epsilon,t'),\epsilon)$. Since $t \geq t'$, $\epsilon \in \mathrm{G}(\mathrm{Reach}(\epsilon,t),\epsilon)$. Moreover, $\mathrm{store}_\varphi(\epsilon,t) = (\epsilon,\epsilon,\epsilon)$, thus $\mathrm{P}(\epsilon,t)$ holds.

Thus, in both cases, $\mathrm{P}(\epsilon)$ holds.

—*Induction step:* suppose that for $\sigma \in \mathrm{tw}(\Sigma)$, $\mathrm{P}(\sigma)$ holds. Let us consider (t',a) such that $\sigma.(t',a) \in \mathrm{tw}(\Sigma)$, and $t \geq t' = \mathrm{time}(\sigma.(t',a))$. Let us also consider $(\sigma_s,\sigma_b,\sigma_c) = \mathrm{store}_\varphi(\sigma,t')$ and $(\sigma_t,\sigma_d,\sigma_e) = \mathrm{store}_\varphi(\sigma.(t',a),t)$.

– *Case 1:* $\sigma.(t',a) \notin \mathrm{Pre}(\varphi,t)$. Then, $\mathrm{P}(\sigma.(t',a),t)$ trivially holds.

– *Case 2:* $\sigma.(t',a) \in \mathrm{Pre}(\varphi,t) \wedge \sigma \notin \mathrm{Pre}(\varphi,t')$. Then, $\sigma \notin \mathrm{Pre}(\varphi,t')$, thus, following lemma 2, $\sigma_s = \sigma_{|\Sigma_{\mathrm{u}}}$, and $\sigma_b = \epsilon$. Since $\sigma.(t',a) \in \mathrm{Pre}(\varphi,t)$, and $\sigma \notin \mathrm{Pre}(\varphi,t')$, there exists $t'' \in \mathbb{R}_{\geq 0}$ such that $t' \leq t'' \leq t \wedge \mathrm{E}_1(\mathrm{obs}(\sigma.(t',a),t''),t'') \neq \emptyset$. Since $t'' > t' = \mathrm{time}(\sigma.(t',a))$, then $\mathrm{obs}(\sigma.(t',a),t'') = \sigma.(t',a)$. This means that $\mathrm{E}_1(\sigma.(t',a),t'') \neq \emptyset$. Thus, let us consider $w \in$

$\mathrm{E}_1(\sigma.(t',a),t'')$. Then, w satisfies:

$$
\begin{array}{l}
(\sigma.(t',a))_{|\Sigma_\mathrm{u}} \preccurlyeq w \wedge (\sigma.(t',a))_{|\Sigma_\mathrm{u}} = w_{|\Sigma_\mathrm{u}} \wedge \\
(\sigma.(t',a))_{|\Sigma_\mathrm{c}} \preccurlyeq_d w_{|\Sigma_\mathrm{c}} \wedge \\
\mathrm{Reach}(w) \in F_G \wedge \\
(w_{|\Sigma_\mathrm{c}} \neq \epsilon \implies \mathrm{date}(w_{|\Sigma_\mathrm{c}}(1)) \geq t'') \wedge \\
\forall t_0 \geq t'', \\
\mathrm{Safe}(\mathrm{Reach}(w,t_0), \Pi_\Sigma(\mathrm{obs}(w_{|\Sigma_\mathrm{c}},t_0))^{-1}.\Pi_\Sigma((\sigma.(t',a))_{|\Sigma_\mathrm{c}})).
\end{array}
$$

Let us consider $w_b = ((\sigma.(t',a))^{-1}_{|\Sigma_\mathrm{u}}.w) -_\mathrm{t} t''$. Then, w_b satisfies:

$$
\begin{array}{l}
\Pi_\Sigma(w_b) \preccurlyeq \Pi_\Sigma((\sigma.(t',a))_{|\Sigma_\mathrm{c}}) \wedge \\
\mathrm{Reach}((\sigma.(t',a))_{|\Sigma_\mathrm{u}}, t'') \text{ after } w_b \in F_G \wedge \\
\forall t_0 \in \mathbb{R}_{\geq 0}, \\
\mathrm{Safe}(\mathrm{Reach}((\sigma.(t',a))_{|\Sigma_\mathrm{u}}, t'') \text{ after } (w_b,t_0), \Pi_\Sigma(\mathrm{obs}(w_b,t_0))^{-1}.\Pi_\Sigma((\sigma.(t',a))_{|\Sigma_\mathrm{c}})).
\end{array}
$$

- If $a \in \Sigma_\mathrm{u}$, then considering that $(\sigma.(t',a))_{|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}}.(t',a) = \sigma_s.(t',a)$, $\sigma_b = \epsilon$, and thus $\sigma_c = \Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}})$, this means that w_b satisfies:

$$
\begin{array}{l}
\Pi_\Sigma(w_b) \preccurlyeq \Pi_\Sigma(\sigma_b).\sigma_c \wedge \\
(\mathrm{Reach}(\sigma_s.(t',a), t'') \text{ after } w_b) \in F_G \wedge \\
\forall t_0 \in \mathbb{R}_{\geq 0}, \\
\mathrm{Safe}(\mathrm{Reach}(\sigma_s.(t',a), t'') \text{ after } (w_b,t_0), \Pi_\Sigma(\mathrm{obs}(w_b,t_0))^{-1}.(\Pi_\Sigma(\sigma_b).\sigma_c)).
\end{array}
$$

  Thus, $w_b \in \mathrm{G}(\mathrm{Reach}(\sigma_s.(t',a),t''), \Pi_\Sigma(\sigma_b).\sigma_c)$.
  Since $\mathrm{G}(\mathrm{Reach}(\sigma_s.(t',a),t''), \Pi_\Sigma(\sigma_b).\sigma_c) \neq \emptyset$, and $t'' \leq t$, this means that $t_1 \leq t'' \leq t$, thus $\sigma_d -_\mathrm{t} t_1 \in \mathrm{G}(\mathrm{Reach}(\sigma_s.(t',a),t_1), \Pi_\Sigma(\sigma_b).\sigma_c)$. Thus, $\mathrm{nobs}(\sigma_d,t) -_\mathrm{t} t \in \mathrm{G}(\mathrm{Reach}(\sigma_s\,.\,(t',\,a)\,.\,\mathrm{obs}(\sigma_d,\,t),\,t),\,\Pi_\Sigma(\mathrm{obs}(\sigma_d$
  Moreover,
  $\Pi_\Sigma(\sigma_b).\sigma_c = \sigma_{|\Sigma_\mathrm{c}}$, thus $\Pi_\Sigma(\mathrm{obs}(\sigma_d,t))^{-1}.(\Pi_\Sigma(\sigma_b).\sigma_c) = \Pi_\Sigma(\mathrm{nobs}(\sigma_d,t)).\sigma_e$, meaning that $\mathrm{nobs}(\sigma_d,\ t)\ -_\mathrm{t}\ t \in \mathrm{G}(\mathrm{Reach}(\sigma_t,\ t),\ \Pi_\Sigma(\mathrm{nobs}(\sigma_d,\ t))\,.\,\sigma_e)$. Thus, $\mathrm{P}(\sigma.(t',a),t)$ holds.

- Otherwise, $a \in \Sigma_\mathrm{c}$. Then, $(\sigma.(t',a))_{|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}} = \sigma_s$, $\sigma_b = \epsilon$, and $\sigma_c = \Pi_\Sigma((\sigma.(t',a))_{|\Sigma_\mathrm{c}}) = \Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}}).a$. Therefore, w_b satisfies:

$$
\begin{array}{l}
\Pi_\Sigma(w_b) \preccurlyeq \Pi_\Sigma(\sigma_b).\sigma_c.a \wedge \\
(\mathrm{Reach}(\sigma_s, t'') \text{ after } w_b) \in F_G \wedge \\
\forall t_0 \in \mathbb{R}_{\geq 0}, \mathrm{Safe}(\mathrm{Reach}(\sigma_s, t'') \text{ after } (w_b,t_0), \Pi_\Sigma(\mathrm{obs}(w_b,t_0))^{-1}.(\Pi_\Sigma(\sigma_b).\sigma_c.a)).
\end{array}
$$

  This means that $w_b \in \mathrm{G}(\mathrm{Reach}(\sigma_s,t''), \Pi_\Sigma(\sigma_b).\sigma_c.a)$.
  Thus, $\mathrm{G}(\mathrm{Reach}(\sigma_s,t''), \Pi_\Sigma(\sigma_b).\sigma_c.a) \neq \emptyset$, meaning that $t_2 \leq t'' \leq t$, thus $\sigma_d -_\mathrm{t} t_2 \in \mathrm{G}(\mathrm{Reach}(\sigma_s,\ t_2),\ \Pi_\Sigma(\sigma_b)\,.\,\sigma_c\,.\,a)$. It follows that $\mathrm{nobs}(\sigma_d,\ t)\ -_\mathrm{t}\ t \in \mathrm{G}(\mathrm{Reach}(\sigma_s\,.\,\mathrm{obs}(\sigma_d,\ t),\ t),\ \Pi_\Sigma(\mathrm{obs}(\sigma_d,\ t))^{-1}\,.\,(\Pi_\Sigma(\sigma_b)\,.\,\sigma_c\,.\,a))$. Moreover, $\Pi_\Sigma(\sigma_b).\sigma_c.a = \Pi_\Sigma((\sigma.(t',a))_{|\Sigma_\mathrm{c}}) = \Pi_\Sigma(\sigma_d).\sigma_e$.
  Thus, $\Pi_\Sigma(\mathrm{obs}(\sigma_d))^{-1}.(\Pi_\Sigma(\sigma_b).\sigma_c.a) = \Pi_\Sigma(\mathrm{nobs}(\sigma_d,t)).\sigma_e$. Thus, $\mathrm{nobs}(\sigma_d,t) -_\mathrm{t} t \in \mathrm{G}(\mathrm{Reach}(\sigma_t,t), \Pi_\Sigma(\mathrm{nobs}(\sigma_d,t)).\sigma_e)$. This means that $\mathrm{P}(\sigma.(t',a),t)$ holds.

Thus, if $\sigma.(t',a) \in \mathrm{Pre}(\varphi,t) \wedge \sigma \notin \mathrm{Pre}(\varphi,t'), \mathrm{P}(\sigma,t) \implies \mathrm{P}(\sigma.(t',a),t)$.

– *Case 3:* $\sigma.(t',a) \in \mathrm{Pre}(\varphi,t)$ and $\sigma \in \mathrm{Pre}(\varphi,t')$. Then, let us consider $w_b = \mathrm{nobs}(\sigma_b,t') -_\mathrm{t} t'$. By induction hypothesis, since $\sigma \in \mathrm{Pre}(\varphi,t')$, $\mathrm{E}_\varphi(\sigma) \models \varphi \wedge w_b \in \mathrm{G}(\mathrm{Reach}(\sigma_s,t'), \Pi_\Sigma(\mathrm{nobs}(\sigma_b,t')).\sigma_c)$ holds.

- If $a \in \Sigma_\mathrm{u}$, then since $w_b \in \mathrm{G}(\mathrm{Reach}(\sigma_s,t'), \Pi_\Sigma(\mathrm{nobs}(\sigma_b,t')).\sigma_c)$,

$\mathrm{Safe}(\mathrm{Reach}(\sigma_s, t'), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c)$ holds. Following the definition of Safe, this means that $\mathrm{Safe}_{\mathrm{int}}(\mathrm{Reach}(\sigma_s, t'), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c, \emptyset)$ holds. Considering the definition of $\mathrm{Safe}_{\mathrm{int}}$, since $a \in \Sigma_{\mathrm{u}}$, there exists $w \in \mathrm{tw}(\Sigma)$ such that:

$$\begin{array}{l}\Pi_\Sigma(w) \preccurlyeq \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c \wedge \\ \forall t'' \in \mathbb{R}_{\geq 0}, \\ \mathrm{Safe}_{\mathrm{int}}(\mathrm{Reach}(\sigma_s, t') \text{ after } ((0,a).w, t''), \\ \quad \Pi_\Sigma(\mathrm{obs}(w, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c), \\ \quad \{(\mathrm{Reach}(\sigma_s, t'), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c)\}).\end{array}$$

Thus, w is such that:

$$\begin{array}{l}\mathrm{Safe}_{\mathrm{int}}(\mathrm{Reach}(\sigma_s, t'), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c, \emptyset) \wedge \forall t'' \in \mathbb{R}_{\geq 0}, \\ \mathrm{Safe}_{\mathrm{int}}(\mathrm{Reach}(\sigma_s, t') \text{ after } ((0,a).w, t''), \Pi_\Sigma(\mathrm{obs}(w, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c), \\ \quad \{(\mathrm{Reach}(\sigma_s, t'), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c\}))\end{array}$$

holds. Thus, following lemma 3, for all $t'' \in \mathbb{R}_{\geq 0}$,

$$\mathrm{Safe}_{\mathrm{int}}(\mathrm{Reach}(\sigma_s, t') \text{ after } ((0,a).w, t''), \Pi_\Sigma(\mathrm{obs}(w, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c), \emptyset)$$

holds. Hence, w satisfies:

$$\begin{array}{l}\Pi_\Sigma(w) \preccurlyeq \Pi_\Sigma(\mathrm{nobs}(\sigma_b), t').\sigma_c \wedge \\ (\mathrm{Reach}(\sigma_s.(t', a)) \text{ after } w) \in F_G \wedge \\ \forall t'' \in \mathbb{R}_{\geq 0}, \\ \mathrm{Safe}_{\mathrm{int}}(\mathrm{Reach}(\sigma_s.(t', a)) \text{ after } (w, t''), \Pi_\Sigma(\mathrm{obs}(w, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c, \emptyset))\end{array}$$

Thus, $w \in \mathrm{G}(\mathrm{Reach}(\sigma_s.(t', a)), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c)$. Thus, $\mathrm{G}(\mathrm{Reach}(\sigma_s.(t', a), t'), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t'))) \neq \emptyset$. This means that $t_1 = t'$, thus $\min(\{t, t_1\}) = t_1 = t'$, and $\sigma_d -_{\mathrm{t}} t' \in \mathrm{G}(\mathrm{Reach}(\sigma_s.(t', a)), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c)$. This implies that $\mathrm{E}_\varphi(\sigma.(t', a)) = \sigma_s.(t', a).\sigma_d \in F_G$, meaning that $\mathrm{E}_\varphi(\sigma.(t', a)) \models \varphi$. Moreover, $w_d = \mathrm{nobs}(\sigma_d, t) -_{\mathrm{t}} t$ satisfies:

$$\begin{array}{l}\Pi_\Sigma(w_d) \preccurlyeq \Pi_\Sigma(\mathrm{nobs}(\sigma_d, t)).\sigma_e \wedge \\ \mathrm{Reach}(\sigma_s.(t', a).\,\mathrm{obs}(\sigma_d, t)) \text{ after } (w_d) \in F_G \wedge \\ \forall t'' \in \mathbb{R}_{\geq 0}, \\ \mathrm{Safe}(\mathrm{Reach}(\sigma_s.(t', a).\,\mathrm{obs}(\sigma_d, t), t) \text{ after } (w_d, t''), \\ \quad \Pi_\Sigma(\mathrm{obs}(w_d, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_d, t)).\sigma_e)).\end{array}$$

Thus, considering that $\sigma_t = \sigma_s.(t', a).\,\mathrm{obs}(\sigma_d, t)$, it follows that $w_d \in \mathrm{G}(\mathrm{Reach}(\sigma_t, t), \Pi_\Sigma(\mathrm{nobs}(\sigma_d, t)).\sigma_e)$. This means that $\mathrm{P}(\sigma.(t', a), t)$ holds.

- Otherwise, $a \in \Sigma_{\mathrm{c}}$. Since $w_b \in \mathrm{G}(\mathrm{Reach}(\sigma_s, t'), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c)$, and $\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c \preccurlyeq \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c.a$, it follows that $w_b \in \mathrm{G}(\mathrm{Reach}(\sigma_s, t'), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c.a)$. This means that $t_2 = t'$, thus $\sigma_d -_{\mathrm{t}} t' \in \mathrm{G}(\mathrm{Reach}(\sigma_s, t'), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c.a)$. It follows that $\mathrm{nobs}(\sigma_d, t) -_{\mathrm{t}} t \in \mathrm{G}(\mathrm{Reach}(\sigma_s.\,\mathrm{obs}(\sigma_d, t), t), \Pi_\Sigma(\mathrm{nobs}(\sigma_d, t)).\sigma_e)$. Since $\sigma_t = \sigma_s.\,\mathrm{obs}(\sigma_d, t)$, $\mathrm{nobs}(\sigma_d, t) -_{\mathrm{t}} t \in \mathrm{G}(\mathrm{Reach}(\sigma_t, t), \Pi_\Sigma(\mathrm{nobs}(\sigma_d, t)).\sigma_e)$. Thus, $\mathrm{P}(\sigma.(t', a), t)$ holds.

Thus, in all cases, for all $t \geq t', \mathrm{P}(\sigma) \implies \mathrm{P}(\sigma.(t', a), t)$. This means that $\mathrm{P}(\sigma) \implies \forall t \geq t', \mathrm{P}(\sigma.(t', a), t)$. Thus, $\mathrm{P}(\sigma) \implies \mathrm{P}(\sigma.(t', a))$.

By induction, for all $\sigma \in \mathrm{tw}(\Sigma)$, $\mathrm{P}(\sigma)$ holds. In particular, for all $(\sigma, t) \in \mathrm{Pre}(\varphi), \mathrm{E}_\varphi(\sigma) \models \varphi$. This means that E_φ is sound in $\mathrm{Pre}(\varphi)$.

□

Proposition 8. E_φ is compliant, as per Definition 12.

Proof. For $\sigma \in \mathrm{tw}(\Sigma)$, let $\mathrm{P}(\sigma)$ be the predicate: "$\forall t \geq \mathrm{time}(\sigma), (\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t) \implies \sigma_{s|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}} \wedge \Pi_\Sigma(\sigma_{s|\Sigma_\mathrm{c}}.\mathrm{nobs}(\sigma_b, t)).\sigma_c = \Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}}) \wedge \sigma_{s|\Sigma_\mathrm{c}} \preccurlyeq_d \sigma_{|\Sigma_\mathrm{c}}$". Let us prove by induction that for all $\sigma \in \mathrm{tw}(\Sigma), \mathrm{P}(\sigma)$ holds.

—*Induction basis:* for $\sigma = \epsilon$. $\mathrm{store}_\varphi(\epsilon) = (\epsilon, \epsilon, \epsilon)$, and $\epsilon_{|\Sigma_\mathrm{c}} = \epsilon_{|\Sigma_\mathrm{u}} = \Pi_\Sigma(\epsilon) = \epsilon$. Thus, $\mathrm{P}(\epsilon)$ trivially holds.

—*Induction step:* suppose now that for some $\sigma \in \mathrm{tw}(\Sigma), \mathrm{P}(\sigma)$ holds. Let us consider (t', a) such that $\sigma.(t', a) \in \mathrm{tw}(\Sigma), t \geq \mathrm{time}(\sigma)$, $(\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t')$, and $(\sigma_t, \sigma_d, \sigma_e) = \mathrm{store}_\varphi(\sigma.(t', a), t)$. Then, by induction hypothesis, $\sigma_{s|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}}$, $\Pi_\Sigma(\sigma_{s|\Sigma_\mathrm{c}}.\sigma_b).\sigma_c = \Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}})$, and $\sigma_{s|\Sigma_\mathrm{c}} \preccurlyeq_d \sigma_{|\Sigma_\mathrm{c}}$.

–*Case 1:* $a \in \Sigma_\mathrm{u}$. By construction, σ_d satisfies $\Pi_\Sigma(\sigma_d) \preccurlyeq \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c$ and $\sigma_d \neq \epsilon \implies \mathrm{date}(\sigma_d(1)) \geq t'$.

•*Projection on* Σ_u.
Since $a \in \Sigma_\mathrm{u}$, $\sigma_{t|\Sigma_\mathrm{u}} = (\sigma_s.(t', a).\,\mathrm{obs}(\sigma_d, t))_{|\Sigma_\mathrm{u}}.\,\sigma_d \in \mathrm{tw}(\Sigma_\mathrm{c})$, thus $\sigma_{t|\Sigma_\mathrm{u}} = \sigma_{s|\Sigma_\mathrm{u}}.(t', a) = \sigma_{|\Sigma_\mathrm{u}}.(t', a) = (\sigma.(t', a))_{|\Sigma_\mathrm{u}}$.

•*Projection on* Σ_c.
$\Pi_\Sigma(\sigma_{t|\Sigma_\mathrm{c}}.\,\mathrm{nobs}(\sigma_d,\ t)).\,\sigma_e = \Pi_\Sigma((\sigma_s.(t', a).\,\mathrm{obs}(\sigma_d,\ t))_{|\Sigma_\mathrm{c}}.\,\mathrm{nobs}(\sigma_d,\ t)).\,\sigma_e = \Pi_\Sigma(\sigma_{s|\Sigma_\mathrm{c}}.\,\sigma_d).\,\sigma_e = \Pi_\Sigma(\sigma_{s|\Sigma_\mathrm{c}}).\,\Pi_\Sigma(\sigma_d).\,\sigma_e$. By construction, $\Pi_\Sigma(\sigma_d).\,\sigma_e = \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c$. Thus, $\Pi_\Sigma(\sigma_{t|\Sigma_\mathrm{c}}.\sigma_d).\sigma_e = \Pi_\Sigma(\sigma_{s|\Sigma_\mathrm{c}}).\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c = \Pi_\Sigma(\sigma_{s|\Sigma_\mathrm{c}}.\,\mathrm{nobs}(\sigma_b, t')).\sigma_c = \Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}}) = \Pi_\Sigma((\sigma.(t', a))_{|\Sigma_\mathrm{c}})$. Moreover, $\sigma_t \in \mathrm{tw}(\Sigma)$, and since $\sigma_t = \sigma_s.(t', a).\,\mathrm{obs}(\sigma_d, t)$, it follows that for all $i \in [1; |\,\mathrm{obs}(\sigma_d, t)|]$, $\mathrm{date}(\sigma_d(i)) \geq t'$. Since $\sigma_{s|\Sigma_\mathrm{c}} \preccurlyeq_d \sigma_{|\Sigma_\mathrm{c}}$, for all $i \in [1; |\sigma_{s|\Sigma_\mathrm{c}}|]$, $\mathrm{date}(\sigma_{s|\Sigma_\mathrm{c}}(i)) \geq \mathrm{date}(\sigma_{|\Sigma_\mathrm{c}}(i))$. Thus, for all $i \in [1; |\sigma_{t|\Sigma_\mathrm{c}}|]$, $\mathrm{date}(\sigma_{t|\Sigma_\mathrm{c}}(i)) \geq \mathrm{date}(\sigma_{|\Sigma_\mathrm{c}}(i))$. Since $\Pi_\Sigma(\sigma_{t|\Sigma_\mathrm{c}}.\sigma_d).\sigma_e = \Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}}), \Pi_\Sigma(\sigma_{t|\Sigma_\mathrm{c}}) \preccurlyeq \Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}})$. Thus $\sigma_{t|\Sigma_\mathrm{c}} \preccurlyeq_d \sigma_{|\Sigma_\mathrm{c}} = (\sigma.(t', a))_{|\Sigma_\mathrm{c}}$.

This means that if $a \in \Sigma_\mathrm{u}$, $\mathrm{P}(\sigma.(t', a))$ holds.

–*Case 2:* $a \in \Sigma_\mathrm{c}$. By construction, σ_d satisfies $\Pi_\Sigma(\sigma_d) \preccurlyeq \Pi_\Sigma(\sigma_b).\sigma_c.a$, and $\sigma_d \neq \epsilon \implies \mathrm{date}(\sigma_d(1)) \geq t'$.

•*Projection on* Σ_u.
$\sigma_{t|\Sigma_\mathrm{u}} = (\sigma_s.\,\mathrm{obs}(\sigma_d, t))_{|\Sigma_\mathrm{u}}$. Since $\sigma_d \in \mathrm{tw}(\Sigma_\mathrm{c})$, $\sigma_{t|\Sigma_\mathrm{u}} = \sigma_{s|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}} = (\sigma.(t', a))_{|\Sigma_\mathrm{u}}$.

•*Projection on* Σ_c.
$\Pi_\Sigma(\sigma_{t|\Sigma_\mathrm{c}}.\,\mathrm{nobs}(\sigma_d,\ t)).\,\sigma_e = \Pi_\Sigma((\sigma_s.\,\mathrm{obs}(\sigma_d,\ t))_{|\Sigma_\mathrm{c}}.\,\mathrm{nobs}(\sigma_d,\ t)).\,\sigma_e = \Pi_\Sigma(\sigma_{s|\Sigma_\mathrm{c}}.\,\sigma_d).\,\sigma_e = \Pi_\Sigma(\sigma_{s|\Sigma_\mathrm{c}}).\,\Pi_\Sigma(\sigma_d).\,\sigma_e$. By construction, $\Pi_\Sigma(\sigma_d).\sigma_e = \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c.a$. Thus, $\Pi_\Sigma(\sigma_{t|\Sigma_\mathrm{c}}.\sigma_d).\sigma_e = \Pi_\Sigma(\sigma_{s|\Sigma_\mathrm{c}}).\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c.a = \Pi_\Sigma(\sigma_{s|\Sigma_\mathrm{c}}.\,\mathrm{nobs}(\sigma_b, t')).\sigma_c.a = \Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}}).a = \Pi_\Sigma((\sigma.(t', a))_{|\Sigma_\mathrm{c}})$. Moreover, considering t_2 as defined in Definition 15, $t_2 \geq t'$, and $t \geq t'$, thus $\min(\{t, t_2\}) \geq t'$, which means that since there exists $w_d \in \mathrm{tw}(\Sigma)$ such that $\sigma_d = w_d +_\mathrm{t} \min(\{t, t_2\})$, if $\sigma_d \neq \epsilon$, then $\mathrm{date}(\sigma_d(1)) \geq t'$. Thus, for all $i \in [1; |\sigma_d|]$, $\mathrm{date}(\sigma_d(i)) \geq t' = \mathrm{time}(\sigma.(t', a))$. This still holds if $\sigma_d = \epsilon$, because then $[1; |\sigma_d|] = \emptyset$. Since $\sigma_{s|\Sigma_\mathrm{c}} \preccurlyeq_d \sigma_{|\Sigma_\mathrm{c}}$, for all $i \in [1; |\sigma_{s|\Sigma_\mathrm{c}}|]$, $\mathrm{date}(\sigma_{s|\Sigma_\mathrm{c}}(i)) \geq \mathrm{date}(\sigma_{|\Sigma_\mathrm{c}}(i))$. Thus, for all $i \in [1; |\sigma_{t|\Sigma_\mathrm{c}}|]$, $\mathrm{date}(\sigma_{t|\Sigma_\mathrm{c}}(i)) \geq \mathrm{date}((\sigma\,.\,(t', a))_{|\Sigma_\mathrm{c}}(i))$. Since $\Pi_\Sigma(\sigma_{t|\Sigma_\mathrm{c}}.\,\mathrm{nobs}(\sigma_d,\ t)).\,\sigma_e = \Pi_\Sigma((\sigma.(t', a))_{|\Sigma_\mathrm{c}}), \Pi_\Sigma(\sigma_{t|\Sigma_\mathrm{c}}) \preccurlyeq \Pi_\Sigma((\sigma.(t', a))_{|\Sigma_\mathrm{c}})$. Thus $\sigma_{t|\Sigma_\mathrm{c}} \preccurlyeq_d (\sigma.(t', a))_{|\Sigma_\mathrm{c}}$.

Thus if $a \in \Sigma_\mathrm{c}$, $\mathrm{P}(\sigma.(t, a))$ holds.

Thus $\mathrm{P}(\sigma) \implies \mathrm{P}(\sigma.(t, a))$.

By induction, for all $\sigma \in \mathrm{tw}(\Sigma)$, for all $t \geq \mathrm{time}(\sigma)$, $(\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t) \implies \sigma_{s|\Sigma_\mathrm{u}} = \sigma_{|\Sigma_\mathrm{u}} \wedge \Pi_\Sigma(\sigma_{s|\Sigma_\mathrm{c}}.\mathrm{nobs}(\sigma_b, t)).\sigma_c = \Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}}) \wedge \sigma_{s|\Sigma_\mathrm{c}} \preccurlyeq_d \sigma_{s|\Sigma_\mathrm{c}}$.

Thus E_φ is compliant. □

Proposition 9. E_φ is optimal in $\mathrm{Pre}(\varphi)$ as per Definition 13.

Proof. Let us consider $E' : \mathrm{tw}(\Sigma) \times \mathbb{R}_{\geq 0} \to \mathrm{tw}(\Sigma)$, that is compliant with respect to Σ_c and Σ_u. Let us also consider $\sigma \in \mathrm{tw}(\Sigma)$, (t', a) such that $\sigma.(t', a) \in \mathrm{tw}(\Sigma)$. Suppose now that $(\sigma, t') \in \mathrm{Pre}(\varphi)$, $E'(\sigma, t') = \mathrm{E}_\varphi(\sigma, t')$, and that $E'(\sigma.(t', a)) \prec_d \mathrm{E}_\varphi(\sigma.(t', a))$. Let us consider $(\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t')$, and $(\sigma_t, \sigma_d, \sigma_e) = \mathrm{store}_\varphi(\sigma.(t', a), t)$, where t is such that $\sigma_t = \mathrm{E}_\varphi(\sigma.(t', a))$.

Then, considering proof of soundness, since $(\sigma,\ t') \in \mathrm{Pre}(\varphi)$, $\mathrm{nobs}(\sigma_b,\ t') -_\mathrm{t} t' \in \mathrm{G}(\mathrm{Reach}(\sigma_s, t'), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c)$.

—If $a \in \Sigma_\mathrm{u}$, this means that $\sigma_d -_\mathrm{t} t' \in \mathrm{G}(\mathrm{Reach}(\sigma_s.(t', a)), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c)$. Thus, $\sigma_d -_\mathrm{t} t' = \mathrm{min}_\mathrm{lex}(\mathrm{max}_{\preccurlyeq}(\mathrm{G}(\mathrm{Reach}(\sigma_s.(t', a)), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c)))$. E' is compliant with respect to Σ_c and Σ_u, thus, since $\mathrm{E}_\varphi(\sigma, t') = E'(\sigma, t')$, there exists $\sigma_{d2} \in \mathrm{tw}(\Sigma)$ such that $E'(\sigma.(t', a)) = \sigma_s.(t', a).\sigma_{d2}$. Since $E'(\sigma.(t', a)) \prec_d \mathrm{E}_\varphi(\sigma.(t', a))$, then $\sigma_{d2} \prec_d \sigma_d$, thus $w_{d2} = \sigma_{d2} -_\mathrm{t} t' \prec_d \sigma_d -_\mathrm{t} t'$, meaning that $w_{d2} \notin \mathrm{G}(\mathrm{Reach}(\sigma_s.(t', a)), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c)$. Since E' is compliant, $\Pi_\Sigma(w_{d2}) \preccurlyeq \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c = \Pi_\Sigma(\sigma_{s|\Sigma_\mathrm{c}})^{-1}.\Pi_\Sigma(\sigma_{|\Sigma_\mathrm{c}})$. Thus, either $\mathrm{Reach}(\sigma_s\,.\,(t',\ a))$ after $w_{d2} \notin F_G$, or there exists $t'' \in \mathbb{R}_{\geq 0}$ such that $\mathrm{Safe}(\mathrm{Reach}(\sigma_s.(t'a))$ after $(w_{d2}, t''), \Pi_\Sigma(\mathrm{obs}(w_{d2}, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c))$ does not hold.

–*Case 1:* $\mathrm{Reach}(\sigma_s.(t', a))$ after $w_{d2} \notin F_G$. Then, $E'(\sigma.(t', a)) = \sigma_s.(t', a).\sigma_{d2} \not\models \varphi$. Thus, $\epsilon \in \mathrm{tw}(\Sigma_\mathrm{u})$ is such that $E'(\sigma.(t', a).\epsilon) \not\models \varphi$.

–*Case 2:* there exists $t'' \in \mathbb{R}_{\geq 0}$ such that $\mathrm{Safe}(\mathrm{Reach}(\sigma_s\,.\,(t',\ a))$ after $(w_{d2},\ t'')$, $\Pi_\Sigma(\mathrm{obs}(w_{d2}, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c))$ does not hold. Considering the definition of Safe:

•if $\Pi_\Sigma(\mathrm{obs}(w_{d2},\ t''))^{-1}\,.\,(\Pi_\Sigma(\mathrm{nobs}(\sigma_b,\ t'))\,.\,\sigma_c) = \epsilon$, then $\mathrm{Reach}(\sigma_s\,.\,(t',\ a))$ after (w_{d2}, t'')after$\mathrm{tw}(\Sigma_\mathrm{u}) \not\subseteq F_G$. This means that there exists $\sigma_u \in \mathrm{tw}(\Sigma_\mathrm{u})$ such that $\mathrm{Reach}(\sigma_s\,.\,(t',\ a)\,.\,\sigma_{d2},\ t'')$after $\sigma_u \notin F_G$. Since $\Pi_\Sigma(\mathrm{obs}(w_{d2},\ t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c) = \epsilon$, this means that $\mathrm{obs}(w_{d2}, t'') = w_{d2}$ (because it has the maximal length w_{d2} can have). Thus, $\mathrm{Reach}(\sigma_s\,.\,(t',\ a)\,.\,w_{d2},\ t'')$ after $\sigma_u = \mathrm{Reach}(\sigma_s.(t', a).w_{d2}.(\sigma_u{+_\mathrm{t}}t'')) \notin F_G$. Since $\sigma_u \in \mathrm{tw}(\Sigma_\mathrm{u})$, E' is compliant, and $\mathrm{date}((\sigma_u{+_\mathrm{t}}\ t'')(1)) \geq \mathrm{time}(\sigma_s.(t', a).w_{d2})$, it follows that $E'(\sigma.(t', a).(\sigma_u{+_\mathrm{t}}t'')) = \sigma_s.(t', a).w_{d2}.(\sigma_u{+_\mathrm{t}}\ t'') \not\models \varphi$.

•Otherwise, there exists $u \in \Sigma_\mathrm{u}$ such that for all $w \in \mathrm{tw}(\Sigma)$, if $\Pi_\Sigma(w) \preccurlyeq (\Pi_\Sigma(\mathrm{obs}(w_{d2}, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c))$, then there exists $t_3 \in \mathbb{R}_{\geq 0}$ such that $\mathrm{Safe}_\mathrm{int}(\mathrm{Reach}(\sigma_s.(t',\ a))$after $(w_{d2},\ t'')$ after $((0,\ u)\ .\ w,\ t_3)$, $\Pi_\Sigma(\mathrm{obs}(w_{d2}, t''))^{-1}.(\Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c))$ does not hold. Let us consider such an action $u \in \Sigma_\mathrm{u}$. Then, there exists $\sigma_{d3} \in \mathrm{tw}(\Sigma)$ such that $E'(\sigma.(t', a).(t'', u)) = \sigma_s.(t', a).\,\mathrm{obs}(\sigma_{d2}, t'').(t'', u).\sigma_{d3}$, and there exists $t_3 \in \mathbb{R}_{\geq 0}$ such that $\mathrm{Safe}(\mathrm{Reach}(\sigma_s\,.\,(t',\ a)\,.\,\mathrm{obs}(\sigma_{d2},\ t'')\,.\,(t'',\ u)\,.\,\sigma_{d3},\ t_3),\ \Pi_\Sigma(\mathrm{obs}(\sigma_{d2},\ t'')\,.\,\mathrm{obs}(\sigma_{d3},\ t_3))^{-1}\,.\,(\Pi_\Sigma(\mathrm{nobs}(\sigma_b,\ t'))\,.\,\sigma_c))$ does not hold. If $\mathrm{obs}(\sigma_{d2}, t'') = \epsilon$, then we can obtain a new uncontrollable event (t_4, u'), that we concatenate to the input and such that the output of E' is still not safe. Iterating this process, if no controllable event is emitted by E' when adding these uncontrollable events to the input, then the condition "$(q, \sigma) \in Q'$" from Definition 14 is satisfied at some point, but since the output is not safe, it means that the condition "$\{q' \in Q \mid (q', \sigma) \in Q'\} \subseteq F_G$}" is not satisfied. Thus, there is some $q' \in Q'$ such that $q \notin F_G$. Since $q' \in Q'$, it means that there exists $\sigma_u \in \mathrm{tw}(\Sigma_\mathrm{u})$, and $t_n \in \mathbb{R}_{\geq 0}$ such that $\mathrm{Reach}(E'(\sigma.(t', a).\sigma_u, t_n)) \notin F_G$. Since no controllable event is emitted by E' with input $\sigma.(t', a).\sigma_u$, this means that $E'(\sigma.(t', a).\sigma_u) \not\models \varphi$ (because $q = (l, v) \in F_G$ iff $l \in F$, it does not depend on time). Otherwise, E' emits controllable events, and then there are

less of them that it can emit in the next step, and since the output is still not safe, we end up in either the second parameter of $\mathrm{Safe}_{\mathrm{int}}$ being ϵ, or the first ones belonging to the third. As shown previously, in both cases, there exists an uncontrollable word $\sigma_u \in \mathrm{tw}(\Sigma_\mathrm{u})$ such that $E'(\sigma.(t',a).\sigma_u) \not\models \varphi$.

Thus, if $a \in \Sigma_\mathrm{u}$, there exists $\sigma_u \in \mathrm{tw}(\Sigma_\mathrm{u})$ such that $E'(\sigma.(t',a).\sigma_u) \not\models \varphi$.

—If $a \in \Sigma_\mathrm{c}$, then since $(\sigma,t') \in \mathrm{Pre}(\varphi)$, $\sigma_d -_\mathrm{t} t' \in \mathrm{G}(\mathrm{Reach}(\sigma_s,t'),\Pi_\Sigma(\mathrm{nobs}(\sigma_b,t')).\sigma_c.a)$. Then, using the same reasoning as when $a \in \Sigma_\mathrm{u}$, the output of E' must not be safe, and thus there must exist $\sigma_u \in \mathrm{tw}(\Sigma_\mathrm{u})$ such that $E'(\sigma.(t',a).\sigma_u) \not\models \varphi$.

This means that whenever $E'(\sigma) = \mathrm{E}_\varphi(\sigma) \wedge E'(\sigma.(t',a)) \prec_d \mathrm{E}_\varphi(\sigma.(t',a))$, then there exists $\sigma_u \in \Sigma_\mathrm{u}$ such that $E'(\sigma.(t',a).\sigma_u) \not\models \varphi$. Thus, E_φ is optimal. □

Proposition 10. The output of $\mathcal{E}$ for input σ is $\mathrm{E}_\varphi(\sigma)$.

Proof. In this proof, we use some notation from Section 4.2:

—$C^\mathcal{E} = \mathrm{tw}(\Sigma) \times \Sigma_\mathrm{c}^* \times Q \times \mathbb{R}_{\geq 0} \times \{\top,\bot\}$ is the set of configurations,
—$c_0^\mathcal{E} = \langle \epsilon,\epsilon,q_0,0,\bot\rangle \in C^\mathcal{E}$ is the initial configuration,
—$\Gamma^\mathcal{E} = ((\mathbb{R}_{\geq 0} \times \Sigma) \cup \{\epsilon\}) \times Op \times ((\mathbb{R}_{\geq 0} \times \Sigma) \cup \{\epsilon\})$ is the alphabet, composed of an optional input, an operation and an optional output,
—The set of operations, to be applied in the given order, is:
$\{\mathrm{compute}, \mathrm{dump}, \text{pass-uncont}, \text{store-cont}, \mathrm{delay}\}$.

Let us also introduce some specific notation. For a sequence of rules $w \in \Gamma^{\mathcal{E}*}$, we note $\mathrm{input}(w) = \Pi_1(w(1)).\Pi_1(w(2))\ldots\Pi_1(w(|w|))$ the concatenation of all inputs from w. In the same way, we define $\mathrm{output}(w) = \Pi_3(w(1)).\Pi_3(w(2))\ldots\Pi_3(w(|w|))$ the concatenation of all outputs from w. Since all configurations are not reachable from $c_0^\mathcal{E}$, for a word $w \in \Gamma^{\mathcal{E}*}$, we will say that $\mathrm{Reach}(w) = c$ if $c_0^\mathcal{E} \xhookrightarrow{w}_\mathcal{E} c$, or $\mathrm{Reach}(w) = \bot$ if such a c does not exist. Let us also define function Rules which, given a timed word and a date, returns the longest sequence of rules that can be applied with the given word as input at the given date:

$$\mathrm{Rules} : \left\{ \begin{array}{ccccc} \mathrm{tw}(\Sigma)\times & \mathbb{R}_{\geq 0} & \to & \Gamma^\mathcal{E} \\ (\sigma, & t) & \mapsto & \max_\preccurlyeq(\{w \in \Gamma^\mathcal{E} \mid \mathrm{input}(w) = \sigma \wedge \mathrm{Reach}(w) \neq \bot \wedge \Pi_4(c) = t\}) \end{array} \right.$$

Since time is not discrete, the rule delay can be applied an infinite number of times by slicing time. Thus, we consider that the rule delay is always applied a minimum number of times, i.e., when two rules delay are consecutive, they are merged into one rule delay, whose parameter is the sum of the parameters of the two rules. The runs obtained are equivalent, but it allows to consider the maximum (for prefix order) of the set used in the definition of Rules.

We then extend output to timed words with a date: for $\sigma \in \mathrm{tw}(\Sigma)$, and a date t, $\mathrm{output}(\sigma,t) = \mathrm{output}(\mathrm{Rules}(\sigma,t))$.

For $\sigma \in \mathrm{tw}(\Sigma)$ and $t \in \mathbb{R}_{\geq 0}$, let $\mathrm{P}(\sigma,t)$ be the predicate: "$\mathrm{E}_\varphi(\sigma,t) = \mathrm{output}(\sigma,t) \wedge (((\sigma_s,\sigma_b,\sigma_c) = \mathrm{store}_\varphi(\mathrm{obs}(\sigma,t),t) \wedge \langle \sigma_b^\mathcal{E},\sigma_c^\mathcal{E},q^\mathcal{E},t,b\rangle = \mathrm{Reach}(\mathrm{Rules}(\sigma,t))) \implies \sigma_b^\mathcal{E} = \mathrm{nobs}(\sigma_b,t) \wedge \sigma_c^\mathcal{E} = \sigma_c \wedge q^\mathcal{E} = \mathrm{Reach}(\sigma_s,t) \wedge (b = \top \implies \mathrm{G}(q^\mathcal{E},\sigma_c^\mathcal{E}) \neq \emptyset))$". Let $\mathrm{P}(\sigma)$ be the predicate "$\forall t \in \mathbb{R}_{\geq 0}, \mathrm{P}(\sigma,t)$ holds". Let us then prove that for all $\sigma \in \mathrm{tw}(\Sigma), \mathrm{P}(\sigma)$ holds.

—*Induction basis*: For $\sigma = \epsilon$, let us consider $t \in \mathbb{R}_{\geq 0}$. Then, $\mathrm{store}_\varphi(\epsilon,t) = (\epsilon,\epsilon,\epsilon)$, and $\mathrm{Reach}(\epsilon,t) = \langle l_0, v_0 + t\rangle$. On the other hand, the only rules that can be applied are delay, and possibly $\mathrm{compute}$, since there is not any input, nor any element to dump. Thus, $\mathrm{Rules}(\epsilon,t) = \epsilon/\,\mathrm{delay}(t)/\epsilon$, or there exists $t' \geq t$ such that $\mathrm{Rules}(\epsilon,t) = \epsilon/\,\mathrm{delay}(t')/\epsilon\,.\,\epsilon/\,\mathrm{compute}()/\epsilon\,.\,\epsilon/\,\mathrm{delay}(t-t')/\epsilon$. Let us consider $c = \mathrm{Reach}(\mathrm{Rules}(\epsilon,t))$. Then, $c = \langle \epsilon,\epsilon,\langle l_0, v_0 + t\rangle, t, b\rangle$. If rule $\mathrm{compute}$ appears in $\mathrm{Rules}(\epsilon,t)$, then $b = \top$, meaning that $\mathrm{G}(q_0 \text{ after } (\epsilon,t'),\epsilon) \neq \emptyset$, and thus that $\mathrm{G}(q_0 \text{ after } (\epsilon,t),\epsilon) \neq \emptyset$ since

$t \geq t'$. Otherwise $b = \bot$. All the other values remain unchanged between the two cases. In both cases, $\mathrm{output}(\mathrm{Rules}(\epsilon, t)) = \epsilon = \mathrm{E}_\varphi(\epsilon, t)$. Thus, $\mathrm{P}(\epsilon)$ holds.

—*Induction step*: Let us suppose now that for some $\sigma \in \mathrm{tw}(\Sigma)$, $\mathrm{P}(\sigma)$ holds. Let us consider $(t', a) \in \mathbb{R}_{\geq 0} \times \Sigma$ such that $\sigma.(t', a) \in \mathrm{tw}(\Sigma)$. Let us then prove that $\mathrm{P}(\sigma.(t', a))$ holds. Let us consider $t \in \mathbb{R}_{\geq 0}$, $c = \langle \sigma_b^{\mathcal{E}}, \sigma_c^{\mathcal{E}}, q^{\mathcal{E}}, t, b\rangle = \mathrm{Reach}(\mathrm{Rules}(\sigma, t'))$, $(\sigma_s, \sigma_b, \sigma_c) = \mathrm{store}_\varphi(\sigma, t')$, and $(\sigma_t, \sigma_d, \sigma_e) = \mathrm{store}_\varphi(\mathrm{obs}(\sigma.(t', a), t), t)$. If $t < t'$, then $\mathrm{obs}(\sigma.(t', a), t) = \mathrm{obs}(\sigma, t)$, and since $\mathrm{P}(\sigma)$ holds, $\mathrm{P}(\sigma.(t', a), t)$ trivially holds. Thus, in the following, we will consider that $t \geq t'$, so that $\mathrm{store}_\varphi(\mathrm{obs}(\sigma.(t', a), t), t) = \mathrm{store}_\varphi(\sigma.(t', a), t)$:

–If $a \in \Sigma_{\mathrm{u}}$, rule pass-uncont can be applied. Let us consider $c' = c$ after $((t', a)/\,\mathrm{pass\text{-}uncont}((t', a))/(t', a))$. Then, $c' = \langle \epsilon, \Pi_\Sigma(\sigma_b^{\mathcal{E}}).\sigma_c^{\mathcal{E}}, q', t', \bot\rangle$, with $q' = q^{\mathcal{E}}\,\mathrm{after}(0, a)$. Then, if $t \geq t_1^{\mathcal{E}}$, where $t_1^{\mathcal{E}} = \min(\{t'' \mid t'' \geq t' \wedge \mathrm{G}(q'\mathrm{after}(\epsilon, t'' - t'), \Pi_\Sigma(\sigma_b^{\mathcal{E}}).\sigma_c^{\mathcal{E}}) \neq \emptyset\})$, then rule $\mathrm{delay}(t_1^{\mathcal{E}} - t')$ can be applied, followed by rule compute. Since $q^{\mathcal{E}} = \mathrm{Reach}(\sigma_s, t')$, $\sigma_b^{\mathcal{E}} = \mathrm{nobs}(\sigma_b, t')$, and $\sigma_c^{\mathcal{E}} = \sigma_c$ (by induction hypothesis), then $\mathrm{G}(q'\mathrm{after}(\epsilon, t'' - t'), \Pi_\Sigma(\sigma_b^{\mathcal{E}}).\sigma_c^{\mathcal{E}}) = \mathrm{G}(\mathrm{Reach}(\sigma_s.(t', a), t''), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c)$, thus $t_1^{\mathcal{E}} = t_1$, where t_1 is defined in Definition 15. Thus, $c'\,\mathrm{after}\,((\epsilon/\,\mathrm{delay}(t_1^{\mathcal{E}} - t')/\epsilon).(\epsilon/\,\mathrm{compute}\,/\epsilon)) = \langle \sigma_d^{\mathcal{E}}, \sigma_e^{\mathcal{E}}, q'\,\mathrm{after}\,(\epsilon, t_1 - t'), t_1, \top\rangle$, with $\sigma_d^{\mathcal{E}} = \kappa_\varphi(q'\,\mathrm{after}\,(\epsilon, t_1 - t'), \Pi_\Sigma(\sigma_b^{\mathcal{E}}).\sigma_c^{\mathcal{E}}) +_{\mathrm{t}} t_1 = \kappa_\varphi(\mathrm{Reach}(\sigma_s.(t', a), t_1), \Pi_\Sigma(\sigma_b).\sigma_c) +_{\mathrm{t}} t_1 = \sigma_d$, and thus $\sigma_e^{\mathcal{E}} = \sigma_e$. Then, rules delay and dump can be applied until date t is reached. In the end, $\mathrm{Reach}(\mathrm{Rules}(\sigma.(t', a), t)) = c'$ after w, where w is composed of an alternation of rules delay and dump, thus $\mathrm{Reach}(\mathrm{Rules}(\sigma.(t', a), t)) = \langle \mathrm{nobs}(\sigma_d^{\mathcal{E}}, t), \sigma_e^{\mathcal{E}}, q'\text{ after }(\mathrm{obs}(\sigma_d^{\mathcal{E}}, t) -_{\mathrm{t}} t', t - t'), t, \top\rangle = \langle \mathrm{nobs}(\sigma_d, t), \sigma_e, \mathrm{Reach}(\sigma_t, t), t, \top\rangle$. Then, $\mathrm{output}(\mathrm{Rules}(\sigma.(t', a), t)) = \mathrm{output}(\mathrm{Rules}(\sigma, t')).(t', a).\,\mathrm{obs}(\sigma_d^{\mathcal{E}}, t) = \sigma_s.(t', a).\,\mathrm{obs}(\sigma_d, t) = \sigma_t$. Thus, if $t \geq t_1$, $\mathrm{P}(\sigma.(t', a), t)$ holds. Otherwise, $t < t_1$, and then rule dump cannot be applied, since $\Pi_5(c') = \bot$, and rule compute also cannot be applied. Thus, the only rule that can be applied is delay, so that $\mathrm{Reach}(\mathrm{Rules}(\sigma.(t', a), t)) = \langle \epsilon, \Pi_\Sigma(\sigma_b^{\mathcal{E}}).\sigma_c^{\mathcal{E}}, q'\text{ after }(\epsilon, t - t'), t', \bot\rangle$. Since $t < t_1$, this means that $\sigma_d = \epsilon$, and $\sigma_e = \Pi_\Sigma(\sigma_b).\sigma_c$. Thus, $\mathrm{output}(\mathrm{Rules}(\sigma.(t', a), t)) = \mathrm{output}(\mathrm{Rules}(\sigma, t')).(t', a) = \sigma_s.(t', a) = \sigma_t$, and $\sigma_d^{\mathcal{E}} = \sigma_d$, and $\sigma_e^{\mathcal{E}} = \sigma_e$. This means that $\mathrm{P}(\sigma.(t', a), t)$ holds when $t < t_1$. Thus, if $a \in \Sigma_{\mathrm{u}}$, then $\mathrm{P}(\sigma.(t', a), t)$ holds for all $t \geq t'$.

–Otherwise, $a \in \Sigma_{\mathrm{c}}$. Then, rule store-cont can be applied. Let us consider $c' = c\mathrm{after}((t', a)/\,\mathrm{store\text{-}cont}(a)/\epsilon)$. Then, $c' = \langle \epsilon, \Pi_\Sigma(\sigma_b^{\mathcal{E}}).\sigma_c^{\mathcal{E}}.a, q^{\mathcal{E}}, t', \bot\rangle$. Let us consider $t_2^{\mathcal{E}} = \min(\{t'' \in \mathbb{R}_{\geq 0} \mid t'' \geq t' \wedge \mathrm{G}(q^{\mathcal{E}}\text{ after }(\epsilon, t'' - t'), \Pi_\Sigma(\sigma_b^{\mathcal{E}}).\sigma_c^{\mathcal{E}}.a) \neq \emptyset\})$. Since $\mathrm{G}(q^{\mathcal{E}}\text{ after }(\epsilon, t'' - t'), \Pi_\Sigma(\sigma_b^{\mathcal{E}}).\sigma_c^{\mathcal{E}}.a) = \mathrm{G}(\mathrm{Reach}(\sigma_s, t''), \Pi_\Sigma(\mathrm{nobs}(\sigma_b, t')).\sigma_c.a)$, it follows that $t_2^{\mathcal{E}} = t_2$ as defined in Definition 15. If $t \geq t_2^{\mathcal{E}} = t_2$, then rule $\mathrm{delay}(t_2 - t')$ can be applied, followed by rule compute. Then, c' after $((\epsilon/\,\mathrm{delay}(t_2 - t')/\epsilon).(\epsilon/\,\mathrm{compute}()/\epsilon)) = \langle \sigma_d^{\mathcal{E}}, \sigma_e^{\mathcal{E}}, q\text{ after }(\epsilon, t_2 - t'), t_2, \top\rangle$, where $\sigma_d^{\mathcal{E}} = \kappa_\varphi(q\mathrm{after}(\epsilon, t_2 - t'), \Pi_\Sigma(\sigma_b^{\mathcal{E}}).\sigma_c^{\mathcal{E}}.a) +_{\mathrm{t}} t_2 = \kappa_\varphi(\mathrm{Reach}(\sigma_s, t_2), \Pi_\Sigma(\sigma_b).\sigma_c.a) +_{\mathrm{t}} t_2 = \sigma_d$. Then, $\sigma_e^{\mathcal{E}} = \sigma_e$. Then, an alternation of rules delay and dump can be applied until date t is reached. This leads to $\mathrm{Reach}(\mathrm{Rules}(\sigma\,.\,(t',\ a),\ t)) = \langle \mathrm{nobs}(\sigma_d^{\mathcal{E}},\ t),\ \sigma_e^{\mathcal{E}},\ q\text{ after }(\mathrm{obs}(\sigma_d^{\mathcal{E}},\ t),\ t),\ t,\ \top\rangle = \langle \mathrm{nobs}(\sigma_d,\ t),\ \sigma_e,\ \mathrm{Reach}(\sigma_t,\ t),\ t,\ \top\rangle$. Moreover, $\mathrm{output}(\mathrm{Rules}(\sigma\,.\,(t',\ a),\ t)) = \mathrm{output}(\sigma, t').\,\mathrm{obs}(\sigma_d, t) = \sigma_s.\,\mathrm{obs}(\sigma_d, t) = \mathrm{E}_\varphi(\sigma.(t', a), t)$. Thus, if $t \geq t_2$, $\mathrm{P}(\sigma.(t', a), t)$ holds. Otherwise, $t < t_2$, meaning that $\sigma_d^{\mathcal{E}} = \epsilon = \sigma_d$, and $\sigma_e^{\mathcal{E}} = \Pi_\Sigma(\sigma_b^{\mathcal{E}}).\sigma_c^{\mathcal{E}}.a = \Pi_\Sigma(\mathrm{nobs}(\sigma_b,\ t'))\,.\,\sigma_c\,.\,a = \sigma_e$, and $\mathrm{output}(\sigma\,.\,(t',\ a),\ t) = \mathrm{output}(\sigma,\ t') = \sigma_s = \mathrm{E}_\varphi(\sigma.(t', a), t)$. Thus, $\mathrm{P}(\sigma.(t', a), t)$ holds.

Thus, $\mathrm{P}(\sigma) \implies \mathrm{P}(\sigma.(t, a))$.

Thus, by induction, for all $\sigma \in \mathrm{tw}(\Sigma)$, $\mathrm{P}(\sigma)$ holds. In particular, for all $\sigma \in \mathrm{tw}(\Sigma)$, and for all $t \in \mathbb{R}_{\geq 0}$, $\mathrm{output}(\sigma, t) = \mathrm{E}_\varphi(\sigma, t)$, meaning that the output of the enforcement monitor $\mathcal{E}$ with input σ at time t is exactly the output of function E_φ with the same input and the same date. □