Resilience of Timed Systems
Sundararaman Akshay, Blaise Genest, Loïc Hélouët, Shankara Krishna, Sparsa Roychowdhury

To cite this version:
Sundararaman Akshay, Blaise Genest, Loïc Hélouët, Shankara Krishna, Sparsa Roychowdhury. Resilience of Timed Systems. 2020. hal-03129402v2

HAL Id: hal-03129402
https://inria.hal.science/hal-03129402v2
Preprint submitted on 26 Aug 2021
Resilience of Timed Systems

S. Akshay
IIT Bombay, Mumbai, India

Blaise Genest
Univ. Rennes, IRISA, CNRS & INRIA, Rennes, France

Loïc Hélouët
Univ. Rennes, IRISA, CNRS & INRIA, Rennes, France

S. Krishna
IIT Bombay, Mumbai, India

Sparsa Roychowdhury
IIT Bombay, Mumbai, India

Abstract
This paper addresses reliability of timed systems in the setting of resilience, that considers the behaviors of a system when unspecified timing errors such as missed deadlines occur. Given a fault model that allows transitions to fire later than allowed by their guard, a system is universally resilient (or self-resilient) if after a fault, it always returns to a timed behavior of the non-faulty system. It is existentially resilient if after a fault, there exists a way to return to a timed behavior of the non-faulty system, that is, if there exists a controller which can guide the system back to a normal behavior. We show that universal resilience of timed automata is undecidable, while existential resilience is decidable, in EXPSPACE. To obtain better complexity bounds and decidability of universal resilience, we consider untimed resilience, as well as subclasses of timed automata.

1 Introduction
Timed automata [2] are a natural model for cyber-physical systems with real-time constraints that have led to an enormous body of theoretical and practical work. Formally, timed automata are finite-state automata equipped with real valued variables called clocks, that measure time and can be reset. Transitions are guarded by logical assertions on the values of these clocks, which allows for the modeling of real-time constraints, such as the time elapsed between the occurrence of two events. A natural question is whether a real-time system can handle unexpected delays. This is a crucial need when modeling systems that must follow a priori schedules such as trains, metros, buses, etc. Timed automata are not a priori tailored to handle unspecified behaviors: guards are mandatory time constraints, i.e., transition firings must occur within the prescribed delays. Hence, transitions cannot occur late, except if late transitions are explicitly specified in the model. This paper considers the question of resilience for timed automata, i.e., study whether a system returns to its normal specified timed behavior after an unexpected but unavoidable delay.

Several works have addressed timing errors as a question of robustness [10, 8, 7], to guarantee that a property of a system is preserved for some small imprecision of up to $\epsilon$ time units. Timed automata have an ideal representation of time: if a guard of a transition contains a constraint of the form $x = 12$, it means that this transition occurs exactly when the value of clock $x$ is 12. Such an arbitrary precision is impossible in an implementation [10]. One way of addressing this is through guard enlargement, i.e., by checking that there exists a small value $\epsilon > 0$ such that after replacing guards of the form $x \in [a, b]$ by $x \in [a - \epsilon, b + \epsilon]$. 

© S. Akshay, B. Genest, L. Hélouët, S. Krishna, S. Roychowdhury; licensed under Creative Commons License CC-BY 4.0
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
**Resilience of Timed Systems**

<table>
<thead>
<tr>
<th></th>
<th>Universal Resilience</th>
<th>Existential Resilience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timed</td>
<td>Undecidable for TA (Prop. 18)</td>
<td>EXPSPACE (Thm. 14)</td>
</tr>
<tr>
<td></td>
<td>EXPSPACE-C for IRTA (Thm. 20)</td>
<td>PSPACE-Hard (Thm. 15, Thm. 32)</td>
</tr>
<tr>
<td>Untimed</td>
<td>EXPSPACE-C (Thm. 21)</td>
<td>PSPACE-C (Thm. 16, Rmk. 17)</td>
</tr>
</tbody>
</table>

Table 1: Summary of results for resilience.

The considered property is still valid, as shown in [7] for $\omega$-regular properties. In [15], robust automata are defined that accept timed words and their neighbors i.e., words whose timing differences remain at a small distance, while in [16, 12, 19, 1], the authors consider robustness via modeling clock drifts. Our goal is different: rather than being robust w.r.t. to slight imprecisions, we wish to check the capacity to recover from a possibly large time deviation. Thus, for a bounded number of steps, the system can deviate arbitrarily, after which, it must return to its specified timed behavior.

A first contribution of this paper is a formalization of resilience in timed automata. We capture delayed events with faulty transitions. These occur at dates deviating from the original specification and may affect clock values for an arbitrarily long time, letting the system diverge from its expected behavior. A system is resilient if it recovers in a finite number of steps after the fault. More precisely, we define two variants. A timed automaton is $K$-$\forall$-resilient if for every faulty timed run, the behavior of the system $K$ steps after the fault cannot be distinguished from a non-faulty behavior. In other words, the system always repairs itself in at most $K$ steps after a fault, whenever a fault happens. This means that, after a fault happens, all the subsequent behaviors (or extensions) of the system are restored to normalcy within $K$ steps. A timed automaton is $K$-$\exists$-resilient if for every timed run ending with a fault, there exists an extension in which, the behavior of the system $K$ steps after the fault cannot be distinguished from a non-faulty behavior. There can still be some extensions which are beyond repair, or take more than $K$ steps after fault to be repaired, but there is a guarantee of at least one repaired extension within $K$ steps after the fault.

In the first case, the timed automaton is fully self-resilient, while in the second case, there exist controllers choosing dates and transitions so that the system gets back to a normal behavior. We also differentiate between timed and untimed settings: in timed resilience recovered behaviors must be indistinguishable w.r.t. actions and dates, while in untimed resilience recovered behaviors only need to match actions.

Our results are summarized in Table 1: we show that the question of universal resilience and inclusion of timed languages are inter-reducible. Thus timed universal resilience is undecidable in general, and decidable for classes for which inclusion of timed languages is decidable and which are stable under our reduction. This includes the class of Integer Reset Timed Automata (IRTA) [18] for which we obtain EXPSPACE containment. Further, untimed universal resilience is EXPSPACE-Complete in general.

Our main result concerns existential resilience, which requires new non-trivial core contributions because of the quantifier alternation ($\forall\exists$). The classical region construction is not precise enough: we introduce strong regions and develop novel techniques based on these, which ensure that all runs following a strong region have (i) matching integral time elapses, and (ii) the fractional time can be retimed to visit the same set of locations and (usual) regions. Using this technique, we show that existential timed resilience is decidable, in EXPSPACE. We also show that untimed existential resilience is PSPACE-Complete.

**Related Work:** Resilience has been considered with different meanings: In [13], faults are modeled as conflicts, the system and controller as deterministic timed automata, and avoiding faults reduces to checking reachability. This is easier than universal resilience which reduces to timed language inclusion, and existential resilience which requires a new notion of
regions. In [14] a system, modeled as an untimed I/O automaton, is considered “sane” if its runs contain at most $k$ errors, and allow a sufficient number $s$ of error-free steps between two violations of an LTL property. It is shown how to synthesize a sane system, and compute (Pareto-optimal) values for $s$ and $k$. In [17], the objective is to synthesize a transducer $E$, possibly with memory, that reads a timed word $\sigma$ produced by a timed automaton $A$, and outputs a timed word $E(\sigma)$ obtained by deleting, delaying or forging new timed events, such that $E(\sigma)$ satisfies some timed property. A related problem, shield synthesis [5], asks given a network of deterministic I/O timed automata $N$ that communicate with their environment, to synthesize two additional components, a pre-shield, that reads outputs from the environment and produces inputs for $N$, and a post-shield, that reads outputs from $N$ and produces outputs to the environment to satisfy timed safety properties when faults (timing, location errors,...) occur. Synthesis is achieved using timed games. Unlike these, our goal is not to avoid violation of a property, but rather to verify that the system recovers within boundedly many steps, from a possibly large time deviation w.r.t. its behavior. Finally, faults in timed automata have also been studied in a diagnosis setting, e.g. in [6], where faults are detected within a certain delay from partial observation of runs.

2 Preliminaries

Let $\Sigma$ be a finite non-empty alphabet and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$ a set of finite or infinite words over $\Sigma$. $\mathbb{R}, \mathbb{R}_{\geq 0}, \mathbb{Q}, \mathbb{N}$ respectively denote the set of real numbers, non-negative reals, rationals, and natural numbers. We write $(\Sigma \times \mathbb{R}_{\geq 0})^\infty = (\Sigma \times \mathbb{R}_{\geq 0})^* \cup (\Sigma \times \mathbb{R}_{\geq 0})^\omega$ for finite or infinite timed words over $\Sigma$. A finite (infinite) timed word has the form $w = (a_1, d_1)\ldots(a_n, d_n)$ (resp. $w = (a_1, d_1)\ldots$) where for every $i$, $d_i \leq d_{i+1}$. For $i \leq j$, we denote by $w_{[i,j]}$, the sequence $(a_i, d_i)\ldots(a_j, d_j)$. Untiming of a timed word $w \in (\Sigma \times \mathbb{R}_{\geq 0})^\infty$ denoted Unt$(w)$, is its projection on the first component, and is a word in $\Sigma^\infty$. A clock is a real-valued variable $x$ and an atomic clock constraint is an inequality of the form $a \gg x \gg b$, with $\gg, \gg \in \{\leq, <\}$, $a, b \in \mathbb{N}$. An atomic diagonal constraint is of the form $a \gg x - y \gg b$, where $x$ and $y$ are different clocks. Guards are conjunctions of atomic constraints on a set $X$ of clocks.

Definition 1. A timed automaton[2] is a tuple $\mathcal{A} = (L, I, X, \Sigma, T, F)$ with finite set of locations $L$, initial locations $I \subseteq L$, finitely many clocks $X$, finite action set $\Sigma$, final locations $F \subseteq L$, and transition relation $T \subseteq L \times \mathcal{G} \times \Sigma \times 2^X \times L$ where $\mathcal{G}$ are guards on $X$.

A valuation of a set of clocks $X$ is a map $\nu : X \to \mathbb{R}_{\geq 0}$ that associates a non-negative real value to each clock in $X$. For every clock $x$, $\nu(x)$ has an integral part $[\nu(x)]$ and a fractional part $\text{frac}(\nu(x)) = \nu(x) - [\nu(x)]$. We will say that a valuation $\nu$ on a set of clocks $X$ satisfies a guard $g$, denoted $\nu \models g$ if and only if replacing every $x \in X$ by $\nu(x)$ in $g$ yields a tautology. We will denote by $[g]$ the set of valuations that satisfy $g$. Given $\delta \in \mathbb{R}_{\geq 0}$, we denote by $\nu + \delta$ the valuation that associates value $\nu(x) + \delta$ to every clock $x \in X$. A configuration is a pair $C = (l, \nu)$ of a location of the automaton and valuation of its clocks. The semantics of a timed automaton is defined in terms of discrete and timed moves from a configuration to the next one. A timed move of duration $\delta$ lets $\delta \in \mathbb{R}_{\geq 0}$ time units elapse from a configuration $C = (l, \nu)$ which leads to configuration $C' = (l, \nu + \delta)$. A discrete move from configuration $C = (l, \nu)$ consists of taking one of the transitions leaving $l$, i.e., a transition of the form $t = (l, g, a, R, l')$ where $g$ is a guard, $a \in \Sigma$ a particular action name, $R$ is the set of clocks reset by the transition, and $l'$ the next location reached. A discrete move with transition $t$ is allowed only if $\nu \models g$. Taking transition $t$ leads the automaton to configuration $C' = (l', \nu')$ where $\nu'(x) = \nu(x)$ if $x \notin R$, and $\nu'(x) = 0$ otherwise.
A region \( r \) of \( A \) is the equivalence class induced by \( \sim \). For a valuation \( \nu \), we denote by \([\nu]\) the region of \( \nu \), i.e., its equivalence class. We will also write \( \nu \in r \) (\( \nu \) is a valuation in region \( r \)) when \( r = [\nu] \). For a given automaton \( A \), there exists only a finite number of regions, bounded by \( 2^K \), where \( K \) is the size of the constraints set in \( A \). It is well known that for a clock constraint \( \psi \) that, if \( \nu \sim \nu' \), then \( \nu' \models \psi \) if and only if \( \nu' \models \psi \). A region \( r' \) is a time successor of another region \( r \) if for every \( \nu \in r \), there exists \( \delta \in \mathbb{R}_{>0} \) such that \( \nu + \delta \in r' \). We denote by \( \text{Reg}(X) \) the set of all possible regions of the set of clocks \( X \). A region \( r \) satisfies a guard \( g \) if and only if there exists a valuation \( \nu \in r \) such that \( \nu \models g \). The region automaton of a timed automaton \( A = (L, I, X, \Sigma, T, F) \) is the untimed automaton \( \mathcal{R}(A) = (S_R, I_R, \Sigma, T_R, F_R) \) that recognizes the untimed language \( \text{Unt}(\mathcal{L}(A)) \). States of \( \mathcal{R}(A) \) are of the form \( (l, r) \), where \( l \) is a location of \( A \) and \( r \) a region, i.e., \( S_R \subseteq L \times \text{Reg}(X), I_R \subseteq I \times \text{Reg}(X), \) and \( F_R \subseteq F \times \text{Reg}(X) \). The transition relation \( T_R \) is such that \(((l, r), a, (l', r')) \in T_R \) if there exists a transition \( t = (l, g, a, R, l') \in T \) such that there exists a time successor region \( r'' \) of \( r \) such that \( r'' \) satisfies the guard \( g \), and \( r' \) is obtained from \( r'' \) by resetting values of clocks in \( R \). The size of the region automaton is the number of states in \( \mathcal{R}(A) \) ans is denoted \( |\mathcal{R}(A)| \). For a region \( r \) defined on a set of clocks \( Y \), we define a projection operator \( \Pi_X(r) \) to represent the region \( r \) projected on the set of clocks \( X \subseteq Y \). Let \( \rho = (l_0, \nu_0) (l_1, d_1) (l_1, \nu_1) \cdots \) be a run of \( A \), where every \( t_i \) is of the form \( t_i = (l_i, g_i, a_i, R_i, l_i') \). The abstract run \( \sigma_\rho = (l_0, r_0) a_1 (l_1, r_1) \cdots \) of \( \rho \) is a path in the region automata \( \mathcal{R}(A) \) such that, \( \forall i \in \mathbb{N}, r_i = [\nu_i] \). We represent runs using variables \( \rho, \pi \) and the corresponding abstract runs with \( \sigma_\rho, \sigma_\pi \) respectively. The region automaton can be used to prove non-emptiness, as \( \mathcal{L}(A) \neq \emptyset \) if \( \mathcal{R}(A) \) accepts some word.

### 3 Resilience Problems

We define the semantics of timed automata when perturbations can delay the occurrence of an action. Consider a transition \( t = (l, g, a, R, l') \), with \( g := x \leq 10 \), where action \( a \) can occur as long as \( x \) has not exceeded 10. Timed automata have a idealized representation of time, and do not consider perturbations that occur in real systems. Consider, for instance that \( 'a' \) is a physical event planned to occur at a maximal time stamp 10: a water tank...
reaches its maximal level, a train arrives in a station etc. These events can be delayed, and nevertheless occur. One can even consider that uncontrollable delays are part of the normal behavior of the system, and that \( \mathcal{L}(A) \) is the ideal behavior of the system, when all delays are met. In the rest of the paper, we propose a fault model that assigns a maximal error to each fireable action. This error model is used to encode the fact that an action might occur at time points slightly greater than what is allowed in the original model semantics.

**Definition 3** (Fault model). A fault model \( P \) is a map \( P : \Sigma \rightarrow \mathbb{Q}_{\geq 0} \) that associates to every action in \( a \in \Sigma \) a possible maximal delay \( P(a) \in \mathbb{Q}_{\geq 0} \).

For simplicity, we consider only executions in which a single timing error occurs. The perturbed semantics defined below easily adapts to a setting with multiple timing errors. With a fault model, we can define a new timed automaton, for which every run \( \rho = (l_0,v_0) \rightarrow (l_1,v_1) \rightarrow (l_2,v_2) \cdots \) contains at most one transition \( t_i = (l,g,a,r,l') \) occurring later than allowed by guard \( g \), and agrees with a run of \( A \) until this faulty transition is taken.

**Definition 4** (Enlargement of a guard). Let \( \phi \) be an inequality of the form \( a \bowtie_1 x \bowtie_2 b \), where \( \bowtie_1, \bowtie_2 \in \{\leq, <\} \). The enlargement of \( \phi \) by a time error \( \delta \) is the inequality \( \phi_{\delta} \) of the form \( a \bowtie_1 x \leq b + \delta \).

Let \( g \) be a guard of the form
\[
g = \bigwedge_{i \in 1..m} \phi_i = a_i \bowtie_1 x_i \bowtie_2 b_i \land \bigwedge_{j \in 1..q} \phi_j = a_j \bowtie_1 x_j - y_j \bowtie_2 u_j \ b_j.
\]

The enlargement of \( g \) by \( \delta \) is the guard \( g_{\delta} = \bigwedge_{i \in 1..m} \phi_{i,\delta} \land \bigwedge_{j \in 1..q} \phi_j \).

For every transition \( t = (l,g,a,R,l') \) with enlarged guard
\[
g_{\delta} = \bigwedge_{i \in 1..m} \phi_{i,\delta} = a_i \bowtie_1 x_i \leq b_i + P(a) \land \bigwedge_{j \in 1..q} \phi_j = a_j \bowtie_1 x_j - y_j \bowtie_2 u_j \ b_j,
\]
we can create a new transition \( t_{f',\rho} = (l,g',a,R,l') \) called a faulty transition such that,
\[
g_{f',\rho} = \bigwedge_{i \in 1..m} \phi_i = b_i \bowtie_1 x_i \leq b_i + \rho(a) \land \bigwedge_{j \in 1..q} \phi_j = a_j \bowtie_1 x_j - y_j \bowtie_2 u_j \ b_j \text{ with } \bowtie_1 \in \{<, \leq\} \setminus \bowtie_1,
\]

Diagonal constraints remain unchanged under enlargement, as the difference between clocks \( x \) and \( y \) is preserved on time elapse. From now, we fix a fault model \( P \) and write \( t_f \) and \( g_f \) instead of \( t_{f',\rho} \) and \( g_{f',\rho} \). Clearly, \( g \) and \( g_f \) are disjoint, and \( g \land g_f \) is equivalent to \( g_{\delta} \).

**Definition 5** (Enlargement of automata). Let \( A = (L,I,X,\Sigma,T,F) \) be a timed automaton.

The enlargement of \( A \) by a fault model \( P \) is the automaton \( A_P = (L_P, I, X, \Sigma, T_P, F_P) \), where
\[
L_P = L \cup \{l \mid l \in L\} \quad \text{and} \quad F_P = F \cup \{l \mid l \in F\}.
\]
A location \( l \) indicates that an unexpected delay has occurred.
\[
T_P = T \cup \hat{T} \quad \text{such that} \quad \hat{T} = \{(l,g,l,a,R,l') \mid (l,g,a,R,l') \in T\} \cup \{(l,g,l,a,R,l') \mid (l,g,a,R,l') \in T\}
\]

\( l \in \hat{T} \) i.e., \( \hat{T} \) is the set of transitions occurring after a fault.

A run of \( A_P \) is faulty if it contains a transition of \( \hat{T} \). It is just faulty if its last transition belongs to \( \hat{T} \) and all other transitions belong to \( T \). Note that while faulty runs can be finite or infinite, just faulty runs are always finite prefix of a faulty run, and end in a location \( \ell \).

**Definition 6** (Back To Normal (BTN)). Let \( K \geq 1 \), \( A \) be a timed automaton with fault model \( P \). Let \( \rho = (l_0,v_0) \rightarrow (l_1,v_1) \rightarrow (l_2,v_2) \cdots \) be a (finite or infinite) faulty accepting run of \( A_P \), with associated timed word \( (a_1,d_1)(a_2,d_2) \cdots \) and let \( i \in \mathbb{N} \) be the position of the faulty transition in \( \rho \). Then \( \rho \) is back to normal (BTN) after \( K \) steps if there exists an accepting run \( \rho' = (l_0',v_0') \rightarrow (l_1',v_1') \rightarrow (l_2',v_2') \cdots \) of \( A \) with associated timed word \( (a_1',d_1')(a_2',d_2') \cdots \) and an index \( \ell \in \mathbb{N} \) such that \( (a_1',d_1')(a_2',d_2') \cdots = (a_{i+K},d_{i+K})(a_{i+K+1},d_{i+K+1}) \cdots \rho \).
The delay can be recovered since the running time since the last stop (covering 2 stations) without any delay, and the red locations represent behaviors when the train chooses to skip the next station to recover from a delay. This mechanism is invoked once the train leaves the faulty transition from $l_1$ (red dotted arrow in Figure 1) is fired. A transition $\text{skip}$ happens after skipping stop at the next station.

**Definition 7 (Resilience).** A timed automaton $\mathcal{A}$ is
- (untimed) $K$-$\forall$-resilient if every finite faulty accepting run is (untimed) BTN in $K$ steps.
- (untimed) $K$-$\exists$-resilient if every just faulty run $\rho_f$ can be extended into a maximal accepting run $\rho_f$ which is (untimed) BTN in $K$ steps.

Intuitively, a faulty run of $\mathcal{A}$ is BTN if the system has definitively recovered from a fault, i.e., it has recovered and will follow the behavior of the original system after its recovery.

The definition of existential resilience considers maximal (infinite, or finite but ending at a fixed goal (a reachability objective, some $\omega$-regular property...)). A classical example of this is a metro network, where trains are often delayed, but nevertheless recover from these delays to reach their destination on time. This motivates the following definition of resilience.

**Example 8.** We model train services to a specific destination such as an airport. On an average, the distance between two consecutive stations is covered in $\leq 4$ time units. At each stop in a station, the dwell time is in between 1 and 2 time units. To recover from a delay, the train is allowed to skip an intermediate station (as long as the next stop is not the destination). Skipping a station is a choice, and can only be activated if there is a delay. We model this system with the timed automaton of Figure 1. There are 5 locations: $l_1$, and $l_2$ represent the normal behavior of the train and $l_3, l_4, l_5$ represent the skipping mechanism. These locations can only be accessed if the faulty transition (represented as a red dotted arrow in Figure 1) is fired. A transition $t_{ij}$ goes from $l_i$ to $l_j$, and $t_{21}$ denotes the faulty transition from $l_2$ to $l_1$. The green locations represent the behavior of the train without any delay, and the red locations represent behaviors when the train chooses to skip the next station to recover from a delay. This mechanism is invoked once the train leaves the station where it arrived late (location $l_3$). When it departs, $x$ is reset as usual; the next arrival to a station (from location $l_4$) happens after skipping stop at the next station. The delay can be recovered since the running time since the last stop (covering 2 stations) is between 6 and 8 units of time. Asking if this system is resilient amounts to asking if this mechanism can be used to recover from delays (defined by the fault model $\mathcal{P}(\text{arr}) = 2$).
see Figure 3, Appendix A). Consider the faulty run \( \rho_f = (\ell_1,0|0)(\ell_2,0|2)(\ell_2,0|1) \) reading (\( \text{dep},2\))\((\text{arr},8)(\text{late},8)(\text{dep},10)(\text{skip},10)(\text{arr},18)(\text{dep},19) \). Run \( \rho_f \) is BTN in 4 steps. It matches the non-faulty run \( \rho = (\ell_1,0|0)(\ell_2,0|2)(\ell_1,4|0)(\ell_1,4|0)(\ell_2,0|2)(\ell_2,0|1) \) reading (\( \text{dep},2\))\((\text{arr},6)(\text{dep},8)(\text{arr},12)(\text{dep},14)(\text{arr},18)(\text{dep},19) \). This automaton is \( K\text{-}\varepsilon\text{-resilient} \) for \( K = 4 \) and fault model \( \mathcal{P} \), as skipping a station after a delay of \( \leq 2 \) time units allows to recover the time lost. It is not \( K\text{-}\varepsilon\text{-resilient} \) for any \( K \), as skipping is not mandatory, and a train can be late for an arbitrary number of steps. In Appendix A we give another example that is \( 1\text{-}\varepsilon\text{-resilient} \).

\[ K\text{-}\varepsilon\text{-resilience} \] always implies \( K\text{-}\exists\text{-resilience} \). In case of \( K\text{-}\varepsilon\text{-resilience} \), every faulty run \( \rho_w \) has to be BTN in \( \leq K \) steps after the occurrence of a fault. This implies \( K\text{-}\exists\text{-resilience} \) since, any just faulty run \( \rho_w \) that is the prefix of an accepting run \( \rho \) of \( \mathcal{A}_\mathcal{P} \) is BTN in less than \( K \) steps. The converse does not hold: \( \mathcal{A}_\mathcal{P} \) can have a pair of runs \( \rho_1, \rho_2 \), sharing a common just faulty run \( \rho_f \) as prefix such that \( \rho_1 \) is BTN in \( K \) steps, witnessing existential resilience, while \( \rho_2 \) is not. Finally, an accepting run \( \rho = \rho_f \rho_s \) in \( \mathcal{A}_\mathcal{P} \) s.t., \( \rho_f \) is just faulty and \(|\rho_s| < K \), is BTN in \( K \) steps since \( \varepsilon \) is a suffix of a run accepted by \( \mathcal{A} \).

4 Existential Resilience

In this section, we consider existential resilience both in the timed and untimed settings.

Existential Timed Resilience. As the first step, we define a product automaton \( \mathcal{B} \otimes_K \mathcal{A} \) that recognizes BTN runs. Intuitively, the product synchronizes runs of \( \mathcal{B} \) and \( \mathcal{A} \) as soon as \( \mathcal{B} \) has performed \( K \) steps after a fault, and guarantees that actions performed by \( \mathcal{A} \) and \( \mathcal{B} \) are performed at the same date in the respective runs of \( \mathcal{A} \) and \( \mathcal{B} \). Before this synchronization, \( \mathcal{A} \) and \( \mathcal{B} \) take transitions or stay in the same location, but let the same amount of time elapse, guaranteeing that synchronization occurs after runs of \( \mathcal{A} \) and \( \mathcal{B} \) of identical durations. The only way to ensure this with a timed automaton is to track the global timing from the initial state of both automata \( \mathcal{A} \) and \( \mathcal{B} \) till \( K \) steps after the fault, even though we do not need the timing for individual actions till \( K \) steps after the fault.

**Definition 9 (Product).** Let \( \mathcal{A} = (L_A, I_A, X_A, \Sigma, T_A, F_A) \) and \( \mathcal{B} = (L_B, I_B, X_B, \Sigma, T_B, F_B) \) be two timed automata, where \( \mathcal{B} \) contains faulty transitions. Let \( K \in \mathbb{N} \) be an integer. Then, the product \( \mathcal{B} \otimes_K \mathcal{A} \) is a tuple \( (L, I, X \cup X_B, (\Sigma \cup \{*\})^2, T, F) \) where \( L \subseteq \{L_B \times L_A \times [-1, K]\} \), \( F = L_B \times F_A \times [-1, K] \), and initial set of states \( I = I_B \times I_A \times \{-1\} \). Intuitively, \(-1 \) means no fault has occurred yet. Then we assign \( K \) and decrement to 0 to denote that \( K \) steps after fault have passed. The set of transitions \( T \) is as follows: We have \( (I_B, I_A, n, g, < x, y >, R, (l''_B, l''_A, n')) \in T \) if and only if either:

\[ n = 0 \] (no fault has occurred, or less than \( K \) steps of \( \mathcal{B} \) have occurred), the action is \( < x, y >= < a, * > \), we have transition \( t_B = (l_B, g, a, R, l''_B) \in T_B, l_A = l''_A \) (the location of \( \mathcal{A} \) is unchanged) and either: \( n = -1 \), the transition \( t_B \) is faulty and \( n' = K \), or \( n = -1 \), the transition \( t_B \) is non faulty and \( n' = -1 \), or \( n > 0 \) and \( n' = n - 1 \).

\[ n = n' \neq 0 \] (no fault has occurred, or less than \( K \) steps of \( \mathcal{B} \) have occurred), the action is \( < x, y >= < *, a > \), we have the transition \( t_A = (l_A, g, a, R, l''_A) \in T_A, l_B = l''_B \) (the location of \( \mathcal{B} \) is unchanged).

\[ n = n' = 0 \] (at least \( K \) steps after a fault have occurred), the action is \( < x, y >= < a, a > \) and there exists two transitions \( t_B = (l_B, g, a, R, l''_B) \in T_B \) and \( t_A = (l_A, g, a, R, l''_A) \in T_A \) with \( g = g_A \land g_B \) and \( R = R_B \cup R_A \) (\( t_A \) and \( t_B \) occur synchronously).
Resilience of Timed Systems

Run\(s\) of \(B \otimes_K A\) are sequences of the form \(\rho^\otimes = (l_0, t_0^A, n_0) (t_1, t_1^A, d_1) \ldots (t_k, t_k^A, d_k) (l_k, t_k^A, n_k)\) where each \((t_i, t_i^A) \in (T_B \cup \{t_i\}) \times (T_A \cup \{t_i^A\})\) defines uniquely the transition of \(B \otimes_K A\), where \(t_i\) corresponds to the transitions with action \(*\). Transitions are of types \((t_i, t_i^A)\) or \((t_x, t_x^A)\) up to a fault and \(K\) steps of \(T_B\), and \((t_i, t_i^A) \in T_B \times T_A\) from there on.

For any timed run \(\rho^\otimes\) of \(A_P \otimes_K A\), the projection of \(\rho^\otimes\) on its first component is a timed run \(\rho\) of \(A_P\), that is projecting \(\rho^\otimes\) on transitions of \(A_P\) and remembering only location and clocks of \(A_P\) in states. In the same way, the projection of \(\rho^\otimes\) on its second component is a timed run \(\rho'\) of \(A\). Given timed runs \(\rho\) of \(A_P\) and \(\rho'\) of \(A\), we denote by \(\rho \otimes \rho'\) the timed run (if it exists) of \(A_P \otimes_K A\) such that the projection on the first component is \(\rho\) and the projection on the second component is \(\rho'\). For \(\rho \otimes \rho'\) to exist, we need \(\rho, \rho'\) to have the same duration, and for \(\rho_s\) the suffix of \(\rho\) starting \(K\) steps after a fault (if there is a fault and \(K\) steps, \(\rho_s = \epsilon\) the empty run otherwise), \(\rho_s\) needs to be suffix of \(\rho'\) as well.

A run \(\rho^\otimes\) of \(A_P \otimes_K A\) is accepting if its projection on the second component \((A)\) is accepting (i.e., ends in an accepting state if it is finite and goes through an infinite number of accepting state if it is infinite). We can now relate the product \(A_P \otimes_K A\) to BTN runs.

\begin{prop}
Let \(\rho_f\) be a faulty accepting run of \(A_P\). The following are equivalent:
\begin{enumerate}
\item \(\rho_f\) is BTN in \(K\)-steps
\item there is an accepting run \(\rho^\otimes\) of \(A_P \otimes_K A\) s.t., the projection on its first component is \(\rho_f\)
\end{enumerate}
\end{prop}

Let \(\rho\) be a finite run of \(A_P\). We denote by \(T_{\rho}^\otimes\) the set of states of \(A_P \otimes_K A\) such that there exists a run \(\rho^\otimes\) of \(A_P \otimes_K A\) ending in this state, whose projection on the first component is \(\rho\). We then define \(S_{\rho}^\otimes\) as the set of states of \(R(A_P \otimes_K A)\) corresponding to \(T_{\rho}^\otimes\), i.e., \(S_{\rho}^\otimes = \{(s, [\nu]) \in R(A_P \otimes_K A) \mid (s, \nu) \in T_{\rho}^\otimes\}\). If we can compute the set \(S = \{S_{\rho}^\otimes \mid \rho\) is a finite run of \(A_P\}\), we would be able to solve timed universal resilience. Proposition 18 shows that universal resilience is undecidable. Hence, computing \(S\) is impossible. Roughly speaking, it is because this set depends on the exact timing in a run \(\rho\), and in general one cannot use the region construction.

We can however show that in some restricted cases, we can use a modified region construction to build \(S_{\rho}^\otimes\), which will enable decidability of timed existential resilience. First, we restrict to just faulty runs, i.e., consider runs of \(A_P\) and \(A\) of equal durations, but that did not yet synchronize on actions in the product \(A_P \otimes_K A\). For a timed run \(\rho\), by its duration, we mean the time-stamp or date of occurrence of its last event. Second, we consider abstract runs \(\tilde{\sigma}\) through a so-called strong region automaton, as defined below. Intuitively, \(\tilde{\sigma}\) keeps more information than in the usual region automaton to ensure that for two timed runs \(\rho_1 = (t_1, d_1)(t_2, d_2) \ldots\) and \(\rho_2 = (t_1, e_1)(t_2, e_2) \ldots\) associated with the same run of the strong region automaton, we have \([e_i] = [d_i]\) for all \(i\). Formally, we build the strong region automaton \(R_{\text{strong}}(B)\) of a timed automaton \(B\) as follows. We add a virtual clock \(x\) to \(B\) which is reset at each integral time point, add constraint \(x < 1\) to each transition guard, and add a virtual self loop transition with guard \(x = 1\) resetting \(x\) on each state. We then make the usual region construction on this extended timed automaton to obtain \(R_{\text{strong}}(B)\). The strong region construction thus has the same complexity as the standard region construction. Let \(\mathcal{L}(R_{\text{strong}}(B))\) be the language of this strong region automaton, where these self loops on the virtual clock are projected away. Indeed, these additional transitions, added to capture ticks at integral times, do not change the overall behavior of \(B\), i.e., we have \(\text{Unl}(\mathcal{L}(B)) \subseteq \mathcal{L}(R_{\text{strong}}(B)) \subseteq \mathcal{L}(R(B)) = \text{Unl}(\mathcal{L}(B))\) so \(\text{Unl}(\mathcal{L}(B)) = \mathcal{L}(R_{\text{strong}}(B))\).

For a finite abstract run \(\tilde{\sigma}\) of the strong region automaton \(R_{\text{strong}}(A_P)\), we define the set \(S_{\tilde{\sigma}}^\otimes\) of states of \(R_{\text{strong}}(A_P \otimes_K A)\) (the virtual clock is projected away, and our region is wrt original clocks) such that there exists a run \(\tilde{\sigma}^\otimes\) through \(R_{\text{strong}}(A_P \otimes_K A)\) ending in this

\(\tilde{\sigma}\).
state and whose projection on the first component is \( \bar{\sigma} \). Let \( \bar{\sigma}_\rho \) be the run of \( R_{\text{strong}}(A_P) \) associated with a run \( \rho \) of \( A_P \). It is easy to see that \( S_{\bar{\sigma}_\rho}^{\otimes K} = \bigcup_{\delta \mid \bar{\sigma}_\rho = \delta} S_{\rho}^{\otimes K} \). For a just faulty timed run \( \rho \) of \( A_P \), we have a stronger relation between \( S_{\rho}^{\otimes K} \) and \( S_{\bar{\sigma}_\rho}^{\otimes K} \):

**Proposition 11.** Let \( \rho \) be a just faulty run of \( A_P \). Then \( S_{\rho}^{\otimes K} = S_{\bar{\sigma}_\rho}^{\otimes K} \).

**Proof.** First, notice that given a just faulty timed run \( \rho \) of \( A_P \) and a timed run \( \rho' \) of \( A \) of same duration, the timed run \( \rho \otimes \rho' \) (the run of \( A_P \otimes_K A \) such that \( \rho \) is the projection on the first component and \( \rho' \) on the second component) exists.

To show that \( S_{\rho}^{\otimes K} = S_{\bar{\sigma}_\rho}^{\otimes K} \), we show that for any pair of just faulty runs \( \rho_1, \rho_2 \) of \( A_P \) with \( \bar{\sigma}_{\rho_1} = \bar{\sigma}_{\rho_2} \), we have \( S_{\rho_1}^{\otimes K} = S_{\rho_2}^{\otimes K} \), which yields the result as \( S_{\bar{\sigma}_\rho}^{\otimes K} = \bigcup_{\rho' \mid \bar{\sigma}_{\rho'} = \bar{\sigma}_\rho} S_{\rho'}^{\otimes K} \). Consider \( \rho_1, \rho_2 \), two just faulty timed runs of \( A_P \) with \( \bar{\sigma}_{\rho_1} = \bar{\sigma}_{\rho_2} \) and let \( (l_{A_P}, l_A, K, r) \in S_{\rho_1}^{\otimes K} \). Then, this implies that there exists the times \( \nu_1 = r \) and a timed run \( \rho_1' \) of \( A \) with the same duration as \( \rho_1 \), such that \( \rho_1 \otimes \rho_1' \) ends in state \( (l_{A_P}, l_A, K, \nu_1) \). The following lemma completes the proof:

**Lemma 12.** There exists \( \nu_2 \models r \) and a timed run \( \rho_2' \) of \( A \) with the same duration as \( \rho_2 \), such that \( \rho_2 \otimes \rho_2' \) ends in state \( (l_{A_P}, l_A, K, \nu_2) \).

The idea of the proof (detailed in appendix B) is to show that we can construct \( \rho_2' \) which will have the same transitions as \( \rho_1' \), with same integral parts in timings (thanks to the information from the strong region automaton), but possibly different timings in the fractional parts, called a retiming of \( \rho_1' \). Notice that \( \rho_2 \) is a retiming of \( \rho_1 \), as \( \bar{\sigma}_{\rho_1} = \bar{\sigma}_{\rho_2} \). We translate the requirement on \( \rho_2' \) into a set of constraints (which is actually a partial ordering) on the fractional parts of the dates of its transitions, and show that we can indeed set the dates accordingly. This translation follows the following idea: the value of a clock \( x \) just before firing transition \( t \) is obtained by considering the date \( d \) of \( t \) minus the date \( d' \) of the latest transition \( t' \) at which \( x \) has been last reset before \( t \). In particular, the inference \( x - y \) between clocks \( x, y \) just before firing transition \( t \) is \( (d - d') - (d - d'') \) or \( d'' - d' \). That is, the value of a clock or its difference can be obtained by considering the difference between two dates of transitions. A constraint given by \( x - y \in (n, n + 1) \) is equivalent with the constraint given by \( d'' - d' \in (n, n + 1) \), and similar constraints on the fractional parts can be given.

Lemma 12 gives our result immediately. Indeed, the lemma implies that \( \nu_2' \models r \) and \( (l_{A_P}, l_A, K, r) \in S_{\rho_2}^{\otimes K} \) from which we infer that \( S_{\rho_1}^{\otimes K} \subseteq S_{\rho_2}^{\otimes K} \). By a symmetric argument we get the other containment also, and hence we conclude that \( S_{\rho_1}^{\otimes K} = S_{\rho_2}^{\otimes K} \).

**Algorithm to solve Existential Timed Resilience.** We can now consider existential timed resilience, and prove that it is decidable thanks to Propositions 10 and 11. The main idea is to reduce the existential resilience question to a question on the sets of regions reachable after just faulty runs. Indeed, focusing on just faulty runs means that we do not have any actions to match, only the duration of the run till the fault, whereas if we had tried to reason on faulty runs in general, actions have to be synchronized \( K \) steps after the fault and then we cannot compute the set of \( S_{\rho}^{\otimes K} \). We can show that reasoning on \( S_{\rho}^{\otimes K} \) for just faulty runs is sufficient. Let \( \rho_f \) be a just faulty timed run of \( A_P \). We say that \( s \in S_{\rho_f}^{\otimes K} \) is safe if there exists a (finite or infinite) maximal accepting run of \( A_P \otimes_K A \) from \( s \), and that \( S_{\rho_f}^{\otimes K} \) is safe if there exists \( s \in S_{\rho_f}^{\otimes K} \) which is safe.

**Lemma 13.** There exists a maximal accepting extension of a just faulty run \( \rho_f \) that is \( \text{BTN in } K \)-steps iff \( S_{\rho_f}^{\otimes K} \) is safe. Further, deciding if \( S_{\rho_f}^{\otimes K} \) is safe can be done in \( \text{PSPACE} \).

**Proof.** Let \( \rho_f \) a just faulty run. By Proposition 10, there exists an extension \( \rho \) of \( \rho_f \) that is \( \text{BTN in } K \) steps if and only if there exists an accepting run \( \rho^{\otimes K} \) of \( A_P \otimes_K A \) such that \( \rho_f \)}}
is a prefix of the projection of $\rho^{\otimes K}_f$ on its first component, if and only if there exists a just faulty run $\rho^{\otimes K}_f$ of $A_P \otimes_K A$ such that its projection on the first component is $\rho_f$, and such that an accepting state of $A_P \otimes_K A$ can be reached after $\rho^{\otimes K}_f$, if and only if $S^{\otimes K}_{\rho_f}$ is safe.

Safety of $S^{\otimes K}_{\rho_f}$ can be verified using a construction similar to the one in Theorem 16: it is hence a reachability question in a region automaton, solvable with a PSPACE complexity. ▶

This lemma means that it suffices to consider the set of $S^{\otimes K}_{\rho_f}$ over all $\rho_f$ just faulty, which we can compute using region automaton thanks to Prop. 11, which gives:

▶ Theorem 14. K-∃-resilience of timed automata is in EXPSPACE.

Proof. Lemma 13 implies that $A$ is not K-timed existential resilient if and only if there exists a just faulty run $\rho_f$ such that $S^{\otimes K}_{\rho_f}$ is not safe. This latter condition can be checked. Let us denote by $R_{\text{strong}}(A_P) = (S_{R(A_P)}, I_{R(A_P)}, \Sigma, T_{R(A_P)}, F_{R(A_P)})$ the strong region automaton associated with $A_P$. We also denote $R^{\otimes K} = (S_{R^{\otimes K}}, I_{R^{\otimes K}}, \Sigma, T_{R^{\otimes K}}, F_{R^{\otimes K}})$ the strong region automaton $R_{\text{strong}}(A_P \otimes_K A)$. Let $\rho_f$ be a just faulty run, and let $\sigma = \bar{\sigma}_f$ denote the run of $R_{\text{strong}}(A_P)$ associated with $\rho_f$. Thanks to Proposition 11, we have $S^{\otimes K}_{\rho_f} = S^{\otimes K}_\sigma$, as $S^{\otimes K}_{\rho_f}$ does not depend on the exact dates in $\rho_f$, but only on their regions, i.e., on $\sigma$.

So it suffices to find a reachable witness $S^{\otimes K}_\sigma$ of $R^{\otimes K}$ which is not safe, to conclude that $A$ is not existentially resilient. For that, we build an (untimed) automaton $B$. Intuitively, this automaton follow $\sigma$ up to a fault of the region automaton $R_{\text{strong}}(A_P)$, and maintains the set $S^{\otimes K}_\sigma$ of regions of $R^{\otimes K}$. This automaton stops in an accepting state immediately after occurrence of a fault. Formally, the product subset automaton $B$ is a tuple $(S_B, I, \Sigma, T, F)$ with set of states $S_B = S_{R_{\text{strong}}(A_P)} \times 2^{S^{\otimes K}_\sigma \times \{0, 1\}}$, set of initial states $I = I_{R_{\text{strong}}(A_P)} \times \{I_{R^{\otimes K}}\} \times \{0\}$, and set of final states $F = S_{R_{\text{strong}}(A_P)} \times 2^{S^{\otimes K}_\sigma \times \{1\}}$. The set of transitions $T \subseteq S_B \times \Sigma \times S_B$ is defined as follows,

\[ (l, r, S, 0, a, (l', r', s', b)) \in T \text{ if and only if } t_R = ((l, r), a, (l', r')) \in T_{R_{\text{strong}}(A_P)} \text{ and } b = 1 \text{ if and only if } t_R \text{ is faulty and } b = 0 \text{ otherwise.} \]

\[ S' = \text{the set of states } s' \text{ of } R_{\text{strong}}(A_P \otimes_K A) \text{ whose first component is } (l', r') \text{ and such that there exists } s \in S(l, a, s') \in T_{R^{\otimes K}}. \]

Intuitively, 0 in the states means no fault has occurred yet, and 1 means that a fault has just occurred, and thus no transition exists from this state. We have that for every prefix $\sigma$ of a just faulty abstract run of $R_{\text{strong}}(A_P)$, ending on a state $(l, r)$ of $R_{\text{strong}}(A_P)$ then, there exists a unique accepting path $\sigma^0 \in B$ such that $\sigma$ is the projection of $\sigma^0$ on its first component. Let $(l, r, S, 1)$ be the state reached by $\sigma^0$. Then $S^{\otimes K}_\sigma = S$. Thus, non-existential resilience can be decided by checking reachability of a state $(l, r, S, 1)$ such that $S$ is not safe in automaton $B$. As $B$ is of doubly exponential size, reachability can be checked in EXPSPACE. Again, since EXPSPACE is closed under complementation we obtain that checking existential resilience is EXPSPACE. ▶

While we do not have a matching lower bound, we complete this subsection with following (easy) hardness result (we leave the details in Appendix B.1 due to lack of space).

▶ Theorem 15. The K-∃-resilience problem for timed automata is PSPACE-Hard.

Existential Untimed Resilience. We next address untimed existential resilience, which we show can be solved by enumerating states $(l, r)$ of $R(A)$ reachable after a fault, and for each of them proving existence of a BTN run starting from $(l, r)$. This enumeration and the following check uses polynomial space, yielding PSPACE-Completeness of K-∃-resilience.

▶ Theorem 16. Untimed K-∃-resilience is PSPACE-Complete.
Thus, we re-label every transition occurring before a fault and till \( K \) steps after the fault. Then, we re-label every transition occurring before a fault and till \( K \) steps after the fault. Then, we can see that the acceptance word of \( A_{1} \) after 15 time units if no fault occurs, and as \( A_{2} \) after 15 time units if a fault occurs. This is done by merging state \( s_{1} \) in the gadget with the initial state of \( A_{1} \) and state \( s_{2} \) with the initial state of \( A_{2} \). Then, we can see that \( \mathcal{L}(A_{1}) \subseteq \mathcal{L}(A_{2}) \) if and only if \( \mathcal{U} \) is 2-\( \forall \)-resilient. A complete proof of this theorem can be found in Appendix C.

Next we show that the reduction is also possible in the reverse direction.

**Proposition 19.** \( \forall \)-resilience can be reduced to language inclusion for timed automata with \( \varepsilon \)-transitions.

**Proof Sketch.** Given a timed automaton \( A \), we build a timed automaton \( A^{S} \) that recognizes all suffixes of timed words in \( \mathcal{L}(A) \). Given a fault model \( P \), we build an automaton \( B^{P} \) from \( A_{P} \) which remembers if a fault has occurred, and how many transitions were taken since a fault. Then, we re-label every transition occurring before a fault and till \( K \) steps after the fault. Then, we can see that the acceptance word of \( A_{1} \) after 15 time units if no fault occurs, and as \( A_{2} \) after 15 time units if a fault occurs. This is done by merging state \( s_{1} \) in the gadget with the initial state of \( A_{1} \) and state \( s_{2} \) with the initial state of \( A_{2} \). Then, we can see that \( \mathcal{L}(A_{1}) \subseteq \mathcal{L}(A_{2}) \) if and only if \( \mathcal{U} \) is 2-\( \forall \)-resilient. A complete proof of this theorem can be found in Appendix C.

**Remark 17.** The hardness reduction in the proof of Theorem 16 holds even for deterministic timed automata. It is known that PSPACE-Completeness of emptiness still holds for deterministic TAs. Hence, considering deterministic timed automata will not improve the complexity of \( \mathcal{K} \)-\( \exists \)-resilience. Considering IRTAs does not change complexity either, as the gadget used in Theorem 16 can be adapted to become an IRTA (as shown in Appendix D).

### 5 Universal Resilience

In this section, we consider the problem of universal resilience and show that it is very close to the language inclusion question in timed automata, albeit with a few subtle differences. One needs to consider timed automata with \( \varepsilon \)-transitions [11], which are strictly more expressive than timed automata. First, we show a reduction from the language inclusion problem.

**Proposition 18.** Language inclusion for timed automata can be reduced to \( \mathcal{K} \)-\( \forall \)-resilience. Thus, \( \mathcal{K} \)-\( \forall \)-resilience is undecidable in general for timed automata.

**Proof Sketch.** Given \( A_{1} = (L_{1}, \{l_{0}\}, X_{1}, \Sigma_{1}, T_{1}, F_{1}) \) and \( A_{2} = (L_{2}, \{l_{0}\}, X_{2}, \Sigma_{2}, T_{2}, F_{2}) \), we start by defining a gadget \( G_{\text{and}} \) as shown in Fig 2 (right). Next, we define an automaton \( \mathcal{U} \) that behaves as \( A_{1} \) if no fault occurs, and as \( A_{2} \) after 15 time units if a fault occurs. This is done by merging state \( s_{1} \) in the gadget with the initial state of \( A_{1} \) and state \( s_{2} \) with the initial state of \( A_{2} \). Then, we can see that \( \mathcal{L}(A_{1}) \subseteq \mathcal{L}(A_{2}) \) if and only if \( \mathcal{U} \) is 2-\( \forall \)-resilient. A complete proof of this theorem can be found in Appendix C.
fault by $\varepsilon$, keeping the same locations, guards and resets, and leave transitions occurring
more than $K$ steps after a fault unchanged to obtain an automaton $B^{P,\varepsilon}$. Accepting locations
of $B^{P,\varepsilon}$ are those of $A$ occurring after a fault in $B^{P}$. Then, every faulty run accepted by
$B^{P,\varepsilon}$ is associated with a word $\rho = (t_1, d_1) \cdots (t_f, d_f)(t_{f+1}, d_{f+1}) \cdots (t_{f+K}, d_{f+K}) \cdots (t_n, d_n)$
where $t_1, \ldots, t_{f+K}$ are $\varepsilon$ transitions. A run $\rho$ is BTN iff $(a_{f+K+1}, d_{f+K+1}) \cdots (a_n, d_n)$ is a
suffix of a timed word of $A$, i.e., is recognized by $A^S$. We can check that every word in
$B^{P,\varepsilon}$ (reading only $\varepsilon$ before a fault) is recognized by the suffix automaton $A^S$, by solving a
language inclusion problem for timed automata with $\varepsilon$ transitions. □

We note that $\varepsilon$-transitions are critical for the reduction of Proposition 19. To get
decidability of $K$-$\forall$-resilience, it is thus necessary (but not sufficient) to be in a class with
decidable timed language inclusion, such as Event-Recording timed automata [3], Integer
Reset timed automata (IRTA) [18], or Strongly Non-Zeno timed automata [4]. However,
to obtain decidability of $K$-$\forall$-resilience using Proposition 19, one needs also to ensure
that inclusion is still decidable for automata in the presence of $\varepsilon$ transitions. When a
subclass $C$ of timed automata is closed by enlargement (due to the fault model), and if timed
language inclusion is decidable, even with $\varepsilon$ transitions, then Proposition 19 implies that
$K$-$\forall$-resilience is decidable for $C$. We show that this holds for the case of IRTA and leave
other subclasses for future work. For IRTA [18], we know that $\mathcal{L}(A) \subseteq \mathcal{L}(B)$ is decidable
in EXPSPACE when $B$ is an IRTA [18] (even with $\varepsilon$ transitions), from which we obtain an
upper bound for $K$-$\forall$-resilience of IRTA. The enlargement of guards due to the fault can add
transitions that reset clocks at non-integral times, but it turns out that the suffix automaton
$A^S$ of Proposition 19 is still an IRTA. A matching lower bound is obtained by encoding
inclusion for IRTA with $K$-$\forall$-resilience using a trick to replace the gadget in Proposition 18
by an equivalent IRTA. Thus, we have Theorem 20 (proof in Appendix D).

$\blacktriangleright$ Theorem 20. $K$-$\forall$-resilience is EXPSPACE-Complete for IRTA.

Finally, we conclude this section by remarking that universal untimed resilience is decidable
for timed automata in general, using the reductions of Propositions 18 and 19:

$\blacktriangleright$ Theorem 21. Untimed $K$-$\forall$-resilience is EXPSPACE-Complete.

Proof Sketch. Untimed language inclusion of timed automata is EXPSPACE-Complete [9],
so the reduction of Proposition 18 immediately gives the EXPSPACE lower bound. For
the upper bound, we use the region construction and an $\varepsilon$-closure to build an automaton
$A^S_U$ that recognizes untimed suffixes of words of $A$, and an automaton $B^{P}_U$ that recognizes
suffixes of words played $K$ steps after a fault. Both are of exponential size. Then untimed
$K$-$\forall$-resilience amounts to checking $\mathcal{L}(B^{P}_U) \subseteq \mathcal{L}(A^S_U)$, yielding EXPSPACE upper bound. □

6 Conclusion

Resilience allows to check robustness of a timed system to unspecified delays. A universally
resilient timed system recovers from any delay in some fixed number of steps. Existential
resilience guarantees the existence of a controller that can bring back the system to a normal
behavior within a fixed number of steps after an unexpected delay. Interestingly, we show
that existential resilience enjoys better complexities/decidability than universal resilience.
Universal resilience is decidable only for well behaved classes of timed automata such as IRTA,
or in the untimed setting. A future work is to investigate resilience for other determinizable
classes of timed automata, and a natural extension of resilience called continuous resilience,
where a system recovers within some fixed duration rather than within some number of steps.
References


Consider a just faulty run where fault occurs either from the initial location \(\ell_1\) and \(\ell_2\), a transition \(t_{12}\) from \(\ell_1\) to \(\ell_2\) and a transition \(t_{21}\) from \(\ell_2\) to \(\ell_1\). The enlarged automaton \(\mathcal{A}_P\) has two extra locations \(\ell_1', \ell_2'\), extra transitions between \(\ell_1\) and \(\ell_2\), and from \(\ell_1\) to \(\ell_2\) and from \(\ell_2\) to \(\ell_1\) respectively. We represent a configuration of the automata with a pair \((\ell, \nu(x), \nu(y))\) where \(\ell\) belongs to the set of the locations and \(\nu(x)\) (resp. \(\nu(y)\)) represents the valuation of clock \(x\) (resp. clock \(y\)). Let \(\rho_f = (\ell_1, 0|0)\xrightarrow{(t_{12}, 6)} (\ell_2, 6|0)\xrightarrow{(t_{21}, 13)} (\ell_1, 0|7)\xrightarrow{(t_{12}, 4)} (\ell_2, 4|0)\) be a faulty run reading the faulty word \((a, 6)(b, 13)(a, 19) \in \mathcal{L}(\mathcal{A}_P)\). This run is 1-BTN since the run \(\sigma = (\ell, 0|0)\xrightarrow{(t_{12}, 6)} (\ell_2, 6|0)\xrightarrow{(t_{21}, 12)} (\ell_1, 0|6)\xrightarrow{(t_{12}, 19)} (\ell_2, 7|0)\) is an accepting run of \(\mathcal{A}\), reading timed word \(\omega_\sigma = (a, 6)(b, 12)(a, 19) \in \mathcal{L}(\mathcal{A})\). Similarly, the run \(\rho' = (\ell, 0|0)\xrightarrow{(t_{12}, 14)} (\ell_2, 14|0)\xrightarrow{(t_{21}, 20)} (\ell_1, 0|6)\xrightarrow{(t_{12}, 31)} (\ell_2, 11|0)\) of \(\mathcal{A}_P\) reading word \((a, 14)(b, 20)(a, 31)\) is 1-BTN because of run \(\sigma' = (\ell_1, 0|0)\xrightarrow{(t_{12}, 10)} (\ell_2, 10|0)\xrightarrow{(t_{21}, 15)} (\ell_1, 0|5)\xrightarrow{(t_{12}, 19)} (\ell_2, 4|0)\xrightarrow{(t_{21}, 20)} (\ell_1, 0|1)\xrightarrow{(t_{12}, 31)} (\ell_2, 11|0)\) reading the word \(\omega_{\sigma'} = (a, 10)(b, 15)(a, 19)(b, 20)(a, 31)\). One can notice that \(\rho'\) and \(\sigma'\) are of different lengths. In fact, we can say something stronger, namely it is 1-\(\varphi\)-resilient (and hence 1-\(\exists\)-resilient) as explained below.

The example consists of a single \((a.b)^*\) loop, where action \(a\) occurs between 3 and 12 time units after entering location \(\ell_1\), and action \(b\) occurs less than 7 time units after entering \(\ell_2\). A fault occurs either from \(\ell_1\), in which case action \(a\) occurs 12 + \(d\) time units after entering \(\ell_1\), with \(d \in [0, 2]\), or from \(\ell_2\), i.e., when \(b\) occurs exactly 7 time units after entering \(\ell_2\). Once a fault has occurred, the iteration of \(a\) and \(b\) continues on \(\ell_1\) and \(\ell_2\) with non-faulty constraints.

Consider a just faulty run \(\rho_f\) where fault occurs on event \(a\). The timed word generated in \(\rho_f\) is of the form \(w_f = (a, d_1)(b, d_2) \ldots (a, d_k)(b, d_{k+1})(a, d_{k+2})\), where \(d_{k+2} = d_{k+1} + 12 + x\) with \(x \in [0, 2]\). The word \(w = (a, d_1)(b, d_2) \ldots (a, d_k)(b, d_{k+1})(a, d_{k+1} + 5)(b, d_{k+1} + 5 +
\[ x \cdot (a, d_{k+1} + 5 + x + 7) \] is also recognized by the normal automaton, and ends at date \( d_{k+1} + 12 + x \). Hence, for every just faulty word \( w_f \) which delays action \( a \), there exists a word \( w \) such for every timed word \( v \), if \( w_f \cdot v \) is accepted by the faulty automaton, \( w \cdot v \) is accepted by the normal automaton. Now, consider a fault occurring when playing action \( b \). The just faulty word ending with a fault is of the form \( w_f = (a, d_1)(b, d_2) \ldots (a, d_k)(b, d_k + 7) \). All occurrences of \( a \) occur at a date between \( d_j + 3 \) and \( d_j + 12 \) for some date \( d_j \) at which location \( \ell_1 \) is reached, (except the first time stamp \( d_1 \in (5, 12) \)) and all occurrences of \( b \) at a date strictly smaller than \( d_i + 7 \), where \( d_i \) is the date of last occurrence of \( a \). Also, for any value \( \epsilon \leq 7 \) the word \( w_\epsilon = (a, d_1)(b, d_2) \ldots (a, d_k)(b, d_k + 7 - \epsilon) \) is non-faulty. Let \( v_1 = 12 - d_1 \), recall that \( d_1 \in (5, 12) \). If we choose \( \epsilon < v_1 \) then the run \( w_\epsilon^+ = (a, d_1 + \epsilon)(b, d_2 + \epsilon) \ldots (a, d_k + \epsilon)(b, d_k + 7) \) is also non-faulty because \( 5 < d_1 + \epsilon < d_1 + v_1 = 12 \). Clearly, we can extend \( w_\epsilon^+ \) to match transitions fired from \( w_\epsilon \) hence, the automaton of the example is 1-\( \nu \)-resilient.

### B Proofs for section 4

#### Lemma 12
There exists \( \nu_2 \models r \) and a timed run \( \rho_2' \) of \( A \) with the same duration as \( \rho_2 \), such that \( \rho_2 \otimes \rho_2' \) ends in state \( (l_{A_\nu}, l_A, K, \nu_2) \).

**Proof.** Let \( t_1, \ldots, t_n \) be the sequence of transitions of \( \rho_1, \rho_2 \) taken respectively, at dates \( d_1, \ldots, d_n \) and \( e_1, \ldots, e_n \). Similarly, we will denote by \( t_1', \ldots, t_k' \) the sequence of transitions of \( \rho_1', \) taken at dates \( d_1', \ldots, d_k' \). Run \( \rho_2' \) will pass by the same transitions \( t_1', \ldots, t_k' \), but with possibly different dates \( e_1', \ldots, e_k' \) such that:

- the duration of \( \rho_2' \) is the same as the duration of \( \rho_2 \),
- \( \widehat{\sigma}_{\rho_2'} \) follows the same sequence of states of \( R_{\text{strong}}(A) \) as \( \widehat{\sigma}_{\rho_1'} \) (in particular, \( \rho_2' \) is a valid run as it fulfills the guards of its transitions, which are the same as those of \( \rho_1' \)),
- \( \widehat{\sigma}_{\rho_2 \otimes \rho_2'} \) reaches the same state of \( R_{\text{strong}}(A_\nu \otimes K A) \) as \( \widehat{\sigma}_{\rho_1 \otimes \rho_1'} \).

We translate these into three successive requirements on the dates \( (e_i')_{i \leq k} \) of \( \rho_2' \):

R1. The first requirement is \( e_k' = e_n \),

R2. The second requirement sets the integral part \( \lfloor e_i' \rfloor = \lfloor d_i' \rfloor \) for all \( i \leq k \). Remark that we already have \( \lfloor e_i' \rfloor = \lfloor e_n \rfloor = \lfloor d_i' \rfloor = \lfloor d_k' \rfloor \) by the first requirement and the hypothesis.

R3. The third requirement tackles the fractional part \( \text{frac}(e_i') \) for all \( i \leq k \). It is given as a set of satisfiable constraints, defined hereafter as a partial ordering on \( \text{frac}(e_i') \) for all \( i \leq k \).

Notice that the value of a clock \( x \) just before firing transition \( t_i \) is obtained by considering the date \( d_i \) of \( t_i \) minus the date \( d_i' \) of the latest transition \( t_j \), \( j < i \) at which \( x \) has been last reset before \( i \). In particular, the difference \( x - y \) between clocks \( x, y \) just before firing transition \( t_i \) is \( (d_i - d_i') - (d_j - d_j') = d_i'' - d_j'' \). That is, the value of a clock or its difference can be obtained by considering the difference between two dates of transitions. A constraint \( c \) given by \( x - y \in (n, n+1) \) is equivalent with the constraint \( d(c) \) given by \( d'' - d''' \in (n, n+1) \).

We then characterize the conditions required for the run \( \rho_2 \otimes \rho_2' \) to reach the same region \( r \) of \( R_{\text{strong}}(A_\nu \otimes K A) \) which was reached by \( \rho_1 \otimes \rho_1' \). These conditions are described as on region \( r \) in the following equivalent ways:

1. A set of constraints \( C \) on the disjoint union \( X'' = X_{A_\nu} \cup X_A \) of clocks of \( A_\nu \) and \( A \), of the form \( x - y \in (n, n+1) \) or \( x - y = n \) or \( x - y > \text{Max} \) (possibly considering a null clock \( y \)) for \( n \in \mathbb{Z} \),
2. The associated set of constraints $C' = \{ d(c) \mid c \in C \}$ on $D = \{ d_x \mid x \in X_{A_{L{\omega}}}, \} \cup \{ d'_x \mid x' \in X_A \}$, with $d_x$ the date of the latest transition $t_{ij}^{\omega}$ that resets the clock $x \in X_{A_{L{\omega}}}$, and $d'_x$ the date of the latest transition $t_{ij}^{\omega}$ that resets clock $x' \in X_A$.

3. An ordering $\leq'$ over $FP = \{ \text{frac}(\tau) \mid \tau \in D \}$ defined as follows: for each constraint $\tau - \tau' \in (n,n+1)$ of $C'$, if $|\tau| = |\tau'| + n$ then $\text{frac}(\tau) <' \text{frac}(\tau')$, and if $|\tau| = |\tau'| + n + 1$ then $\text{frac}(\tau') <' \text{frac}(\tau)$.

   For each constraint $\tau - \tau' = n$ of $C'$, then $\text{frac}(\tau') =' \text{frac}(\tau)$.

   For each constraint $\tau - \tau' > c_{\text{max}}$ of $C'$ such that $|\tau| = |\tau'| + c_{\text{max}}$, we have $\text{frac}(\tau') >' \text{frac}(\tau)$ (if $|\tau| \geq |\tau'| + c_{\text{max}} + 1$, then we don't need to do anything), where $c_{\text{max}} = \max\{ (e_x \mid x \in X) \}$).

Further, path $\rho_2'$ needs to visit the regions $r_1, \ldots, r_k$ visited by $\rho_1'$. For each $i$, visiting region $r_i$ is characterized by a set of constraints $C_i$, which we translate as above as an ordering $\leq_i$ on $FP' = \{ \text{frac}(d'_i) \mid i \leq k \}$.

Thus, finally, we can collect all the requirements for having $\rho'$ with required properties by defining $\leq''$ over $FP' \cup FP$ (notice that it is not a disjoint union) as the transitive closure of the union of all $\leq_i$ and of $\leq$. As the union of constraints on $C_i'$ and on $C'$ is satisfied by the dates $(d_i)_i \leq n$ and $(d'_i)_i \leq k$ of $\rho_1$ and $\rho_1'$, the union of constraints is satisfiable. Equivalently, $\leq''$ is a partial ordering, respecting the total natural ordering $\leq$ on $FP \cup FP'$. We will denote $\tau =\ns \tau'$ whenever $\tau \leq'' \tau'$ and $\tau' \leq'' \tau$, and $\tau <'' \tau'$ if $\tau \leq'' \tau'$ but we don't have $\tau ='' \tau'$. Because $\leq''$ is a partial ordering, there is no $\tau, \tau'$ with $\tau <'' \tau' <'' \tau$.

Note that there is only one way of fulfilling the first two requirements R1. and R2; namely by matching $e'_k$ and $e_n$, and by witnessing dates with the same integral parts in $e'_k, e_n$ as well as $d''_k, d_n$. While this takes care of the last values, to obtain the remaining values, we can apply any greedy algorithm fixing successively $\text{frac}(e'_{k-1}) \ldots \text{frac}(e'_1)$ and respecting $\leq''$ to yield the desired result. We provide a concrete such algorithm for completeness:

We will start from the fixed value of $\text{frac}(e'_{k-1})$ and work backwards. Let us assume inductively that $\text{frac}(e'_{k-1}) \ldots \text{frac}(e'_{i+1})$ have been fixed. We now describe how to obtain $\text{frac}(e'_i)$. If $\text{frac}(d'_i) ='' \text{frac}(d'_j)$, $j > i$ then we set $\text{frac}(e'_i) = \text{frac}(e'_j)$. If $\text{frac}(d'_i) ='' \text{frac}(d'_j)$, then we set $\text{frac}(e'_i) = \text{frac}(e_j)$. Otherwise, consider the sets $L_i = \{ \text{frac}(e_j) \mid i \leq j, \text{frac}(d'_j) <'' \text{frac}(d'_i) \}$ and $U_i = \{ \text{frac}(e_j) \mid j \leq i, \text{frac}(d'_j) >'' \text{frac}(d'_i) \}$. We then set $\text{frac}(e'_i)$ to any value in $(l_i, u_i)$. It remains to show that we indeed possible.

By contradiction, consider that there exists $i$ such that $l_i \geq u_i$, and consider the maximal (first) such $i$. First, assume that both $l_i$ and $u_i$ are of the form $\text{frac}(\sigma_{j_1}), \text{frac}(\sigma_{j_2})$ respectively, i.e. corresponds to clock values in the last regions of $\rho_2$. The contradiction hypothesis is $l_i = \text{frac}(\sigma_{j_1}) \geq u_i = \text{frac}(\sigma_{j_2})$. By definition of $L_i$ and $U_i$, we also have $\text{frac}(d_{j_1}) <' \text{frac}(d_{j_2})$. This is a contradiction with $l_i = \text{frac}(\sigma_{j_1}) \geq u_i = \text{frac}(\sigma_{j_2})$. As $u_i = \text{frac}(\sigma_{j_2})$ with $k > j$ is similar since it has been fixed before $\text{frac}(\sigma_{j_1})$. We have $\text{frac}(d_{j_1}) <'' \text{frac}(d_{j_2})$ by definition of $L_i, U_i$. In particular, $\text{frac}(d_{j_1}) <'' \text{frac}(d_{j_2})$: That is, $k \in U_j$, and by construction, and as $j > i$, we have $l_i = \text{frac}(e'_{j_i}) <' \text{frac}(e_k) = u_i$, a contradiction.
We take any automaton $A$. Then, if $A$ has no accepting word, there is no timed word after a fault which is a suffix of a word in $L(A)$, and conversely, if $L(A) \neq \emptyset$, then any word recognized from $s_1$ is also recognized from $q_e$. So the language emptiness problem reduces to a 2-$\exists$-resilience question.

B.1 Hardness for $K$-$\exists$-resilience

Theorem 15 The $K$-$\exists$-resilience problem for timed automata is PSPACE-Hard.

Proof. We proceed by reduction from the language emptiness problem, which is known to be PSPACE-Complete for timed automata. We can reuse the gadget $G_{\text{und}}$ of Figure 6. We take any automaton $A$ and collapse its initial state to state $s_1$ in the gadget. We recall that $s_1$ is accessible at date 15 only after a fault. We add a self loop with transition, $t_c = (s_2, \sigma, \text{true}, \emptyset, s_2)$ for every $\sigma \in \Sigma$. This means that after reaching $s_2$, which is accessible only at date 15 if no fault has occurred, the automaton accepts any letter with any timing. Then, if $A$ has no accepting word, there is no timed word after a fault which is a suffix of a word in $L(A)$, and conversely, if $L(A) \neq \emptyset$, then any word recognized from $s_1$ is also recognized from $q_e$. So the language emptiness problem reduces to a 2-$\exists$-resilience question.

B.2 Untimed $K$-$\exists$-resilience

Theorem 16 Untimed $K$-$\exists$-resilience is PSPACE-Complete.

Proof. Membership : For every run of $A$, there is a path in $R(A)$. So, $A$ is untimed $K$-$\exists$-resilient if and only if, for all states $q$ reached by a just faulty run, there exists a maximal accepting path $\sigma$ from $q$ such that, $K$ steps after, the sequence of actions on its suffix $\sigma_a$ agrees with that of an accepting path $\sigma$ in $R(A)$. We now prove that this property can be verified in PSPACE.

Let $q = (l, r)$ be a state of $R(A_F)$ reached after a just faulty run. $K$ steps after reaching $q = (l, r)$ of $R(A_F)$, one can check in PSPACE, if there exists a path $\sigma_s$ whose sequence of actions is the same as the suffix of an accepting path $\sigma$ of $R(A)$. That is, either both these end in a pair of accepting states from which no transitions are defined (both paths are maximal), or visit a pair of states twice such that the cyclic part of the path contains both an accepting state of $R(A_F)$ and an accepting state of $R(A)$. To find these paths $\sigma, \sigma_s$, one just needs to guess them, i.e., build them synchronously by adding a pair of transitions to the already built path only if they have the same label. One needs to remember the current pair of states reached, and possibly guess a pair of states $(s_A, s_{A_F})$ on which a cycle starts, and two bits $b_A$ (resp. $b_{A_F}$) to remember if an accepting state of $A$ (resp. $A_F$) has been seen since $(s_A, s_{A_F})$. A maximal finite path or a lasso can be found on a path of length smaller than $|R(A_F)| \times |R(A)|$, and the size of the currently explored path can be memorized with
log₂(|R(𝐴ₚ)| × |R(𝐴)|) bits. This can be done in PSPACE. The complement of this, i.e., checking that no maximal path originating from q with the same labeling as a suffix of a word recognized by R(𝐴) K steps after a fault exists, is in PSPACE too.

Now, to show that 𝐴 is not untimed K-3-resilient, we simply have to find one untimed non-K-3-resilient witness state q reachable immediately after a fault. To find it, non-deterministically guess such a witness state q along with a path of length not more than the size of |R(𝐴ₚ)| and apply the PSPACE procedure above to decide whether it is a untimed non-K-3-resilience witness. Guess of q is non-deterministic, which gives an overall NPSPACE complexity, but again, using Savitch’s theorem, we can say that untimed K-3-resilience is in PSPACE.

**Hardness:** We can now show that untimed K-3-resilience is PSPACE-Hard. Consider a timed automaton 𝐴 with alphabet Σ and the construction of an automata that uses a gadget shown in Figure 5 (right). Let us call this automaton B_{Σ;ₐ;A}. This automaton reads a word (a, 1), (b, 1), (c, 11) and then accepts all timed words 2 steps after a fault, via Σ loop on a particular accepting state q_c. If B_{Σ;ₐ;A} takes the faulty transition (marked in dotted red) then it resets all clocks of 𝐴 and behaves as 𝐴. The accepting states are q_u ∪ F. Then, 𝐴 has an accepting word if and only if B_{Σ;ₐ;A} is untimed 2-3-resilient. Since the emptiness problem for timed automata is PSPACE-Complete, the result follows.

**C Proofs for section 5**

▶ **Proposition 18** Language inclusion for timed automata can be reduced to K-𝑉-resilience.

Thus, K-𝑉-resilience is undecidable in general for timed automata.

**Proof.** Let 𝐴₁ = (L₁, {l₀₁}, X₁, Σ₁, T₁, F₁) and 𝐴₂ = (L₂, {l₀₂}, X₂, Σ₂, T₂, F₂) be two timed automata with only one initial state (w.l.o.g.). We build a timed automaton 𝐵 such that 𝐿(𝐴₁) ⊆ 𝐿(𝐴₂) if and only if 𝐵 is 2-𝑉-resilient.

We first define a gadget 𝐺_{und} that allows to reach a state s₁ at an arbitrary date d₁ = 15 when a fault happens, and a state s₂ at date d₂ = d₁ = 15 when no fault occur. This gadget is shown in Fig 6. G_{und} has 6 locations s₀, s₁, s₁, s₁, s₂ ∉ L₁ ∪ L₂, three new clocks x, y, z ∉ X₁ ∪ X₂, three new actions a, b, c ∉ Σ₁ ∪ Σ₂, and 5 transitions t₀, t₁, t₂, t₃, t₄ ∉ T₁ ∪ T₂ defined as: t₀ = (s₀, a, go, {y, s₁}) with g₀ := x ≤ 10, t₁ = (s₁, b, g₁, ∅, s₁, s₁) with g₁ := x > 11 ∧ y < 1, t₂ = (s₁, b, g₂, ∅, s₁) with g₂ := x ≤ 10, t₃ = (s₁, c, g₃, X₁, s₁) with g₃ := z = 15, and t₄ = (s₂, c, g₄, X₂, s₂) with g₄ := z = 15. Clearly, in this gadget, transition t₁ can never fire, as a configuration with x > 11 and y < 1 is not accessible.

We build a timed automaton 𝐵 that contains all transitions of 𝐴₁ and 𝐴₂, but preceded by 𝐺_{und} by collapsing the initial location of 𝐴₁ i.e., l₀₁ with s₁ and the initial location of 𝐴₂ i.e., l₀₂ with s₂. We also use a fault model 𝑃 : a → [0, 2], that can delay transitions t₀ with action a by up to 2 time units. The language 𝐿(𝐵) is the set of words:

𝐿(𝐵) = \{(a, d₁)(b, d₂)(c, 15)(σ₁, d₃) \ldots (σₙ, dₙ₊₂) | (d₁ ≤ 10) ∧ (d₂ ≤ 10) ∧ (d₃ ≤ 10) ∧ \exists w = (σ₁, d₃') \ldots (σₙ, dₙ₊₂') ∈ 𝐿(𝐴₂), \forall i ∈ 3..n + 2, dₙ' = d₁ - 15\}

The enlargement of 𝐵 is denoted by 𝐵_𝑃. The words in 𝐿(𝐵_𝑃) is the set of words in 𝐿(𝐵) (when there is no fault) plus the set of words in:

𝐿^F(𝐵_𝑃) = \{(a, d₁)(b, d₂)(c, 15)(σ₁, d₃) \ldots (σₙ, dₙ₊₂) | (10 < d₁ ≤ 12) ∧ (d₂ > 11 ∧ (d₂ - d₁ < 1) ∧ \exists w = (σ₁, d₃') \ldots (σₙ, dₙ₊₂') ∈ 𝐿(𝐴₁), \forall i ∈ 3..n + 2, dₙ' = d₁ - 15\}

Now, 𝐵 is K-𝑉-resilient for K = 2 if and only if every word in 𝐿^F(𝐵_𝑃) is BTN after 2 steps (K = 2), i.e., for every word w = (a, d₁)(b, d₂)(c, 15)(σ₁, d₃) \ldots (σₙ, dₙ₊₂) in 𝐿^F(𝐵_𝑃),
if there exists a word $w = (a, d'_1)(b, d'_2)(c, 15)(d_1, d_3) \ldots (d_n, d_{n+2})$ in $L(B)$. This means that every word of $A_1$ is a word of $A_2$. So $L(A_1) \subseteq L(A_2)$ if and only if $B$ is 2-$\forall$-resilient.

As language inclusion for timed automata is undecidable [2], an immediate consequence is that $K$-$\forall$-resilience of timed automata is undecidable.

\[ \square \]

**Proposition 19** $K$-$\forall$-resilience can be reduced to language inclusion for timed automata with $\varepsilon$-transitions.

**Proof.** Given a timed automaton $A = (L, I, X, \Sigma, T, F)$, we can build a timed automaton $A^S$ that recognizes all suffixes of timed words recognized by $A$. Formally, $A^S$ contains the original locations and transitions of $A$, a copy of all location, a copy of all transitions where letters are replaced by $\varepsilon$, and a transition from copies to original locations labeled by their original letters.

We have $A^S = (L^S, F^S, X, \Sigma \cup \{\varepsilon\}, T^S, F)$, where $L^S = L \cup \{l' | l \in L\}$, $F^S = \{l' \in L_S, l \in I\}$ $T^S = T \cup \{(l'_1, g, \varepsilon, R, l'_2) | \exists (l_1, g, \sigma, R, l_2) \in T\} \cup \{(l'_1, g, \sigma, R, l_2) | \exists (l_1, g, \sigma, R, l_2) \in T\}$.

Obviously, for every timed word $(a_1, d_1)(a_2, d_2) \ldots (a_n, d_n)$ recognized by $A$, and every index $k \in 1..n$, the words $(\varepsilon, d_1)(\varepsilon, d_k)(a_{k+1}, d_{k+1}) \ldots (a_n, d_n)$ is recognized by $A^S$.

Given a timed automaton $A$ and a fault model $P$, we build an automaton $B^P$ which remembers if a fault has occurred, and how many transitions have been taken since a fault.

\[ \square \]

**Definition 23 (Counting automaton).** Let $A_P = (L, I, X, \Sigma, T, F)$ and be a timed automaton with faulty transitions. Let $K \in \mathbb{N}$ be an integer. Then, the faulty automaton $B^P$ is a tuple $B^P = (L^P, \Gamma^P, X, \Sigma, T^P, F^P)$ where $L^P \subseteq \{L \times \{0\}\}$, $F^P = F \times [-1, K]$, and initial set of states $I^P = I \times \{-1\}$. Intuitively, $-1$ means no fault has occurred yet. Then we assign $K$ and decrement to $0$ to denote that $K$ steps after fault have passed. The set of transitions $T^P$ is as follows: We have $((l, n), g, a, R, (l', n')) \in T^P$ if and only if either:

- $n \neq 0$ (no fault has occurred, or less than $K$ steps of $B$ have occurred), we have transition $t = (l, g, a, R, l) \in T$, and either: $n = -1$, the transition $t$ is faulty and $n' = K$, or $n = -1$, the transition $t$ is non faulty and $n' = -1$, or $n > 0$ and $n' = n - 1$.
- $n = n' = 0$ (at least $K$ steps after a fault have occurred), and there exists a transition $t = (l, g, a, R, l') \in T$.

Then, we can build an automaton $B^P_\varepsilon$ by re-labeling every transition occurring before a fault and until $K$ steps after the fault by $\varepsilon$, keeping the same locations, guards and resets, and leave transitions occurring more than $K$ steps after a fault unchanged. The relabeled transitions are transitions starting from a location $(l, n)$ with $n \neq 0$. Accepting locations of $B^P_\varepsilon$ are of the form $(l, 0)$ where $l$ is an accepting locations of $A$ occurring after a fault in $B^P$. Then, every faulty run accepted by $B^P_\varepsilon$ is associated with a word of the form $\rho = (t_1, d_1) \ldots (t_d, d_f)(t_{f+1}, d_{f+1}) \ldots (t_{f+K}, d_{f+K}) \ldots (t_n, d_n)$ where $t_1, \ldots t_{f+K}$ are $\varepsilon$ transitions. A run $\rho$ is BTN if and only if $(a_{f+K+1}, d_{f+K+1}) \ldots (a_n, d_n)$ is a suffix of a timed word of $A$, i.e., is recognized by $A^S$.

Now one can check that every word in $B^P_\varepsilon$ (reading only $\varepsilon$ before that fault) is recognized by the suffix automaton $A^S$, i.e. solve a language inclusion problem for timed automata with $\varepsilon$ transitions.

\[ \square \]

**Theorem 21** Untimed $K$-$\forall$-resilience is EXPSPACE-Complete.
D Resilience of Integer Reset Timed Automata

Let us recall some elements used to prove decidability of language inclusion in IRTA. For a given IRTA $A$, we can define a map $f : \rho \rightarrow w_{unt}$ that maps every run $\rho$ of $A$ to an untimed word $w_{unt} \in (\{\checkmark, \delta\} \cup \Sigma)^*$. For a real number $x$ with $k = \lfloor x \rfloor$, we define a map $dt(x)$ from $\mathbb{R}$ to $\{\checkmark, \delta\}^*$ as follows: $dt(x) = (\checkmark, \delta)^k$ if $x$ is integral, and $dt(x) = (\delta, \checkmark)^k, \delta$ otherwise. Then, for two reals $x < y$, the map $dte(x, y)$ is the suffix that is added to $dt(x)$ to obtain $dt(y)$. Last, the map $f$ associates to a word $w = (a_1, d_1) \ldots (a_n, d_n)$ the word...
\[ f(w) = w_1a_1w_2a_2 \ldots w_n a_n \] where each \( w_i \) is the word \( w_i = dte(d_{i-1}, d_i) \). The map \( f \) maps global time elapse to a word of \( \varphi \) and \( \delta \) but keeps actions unchanged. We define another map \( f_\downarrow : w \rightarrow \{ \varphi, \delta \}^* \) that maps every word \( w \) of \( A \) to a word in \( \{ \varphi, \delta \}^* \) dropping the actions from \( f(w) \). Consider for example, a word \( w = (a, 1, 6)(b, 2, 7)(c, 3, 4) \) then, \( f(w) = \delta \varphi \delta a \varphi \delta b \varphi \delta c \), and \( f_\downarrow(w) = \delta \varphi \delta \varphi \delta \varphi \delta \). It is shown in [18] for two timed words \( \rho_1, \rho_2 \) with \( f(\rho_1) = f(\rho_2) \) then \( \rho_1 \in L(A) \) if and only if \( \rho_2 \in L(A) \). It is also shown that we can construct a Marked Timed Automata (MA) from \( A \) with one extra clock and polynomial increase in the number of locations such that \( \text{Unt}(L(MA)) = f(L(A)) \). The MA of \( A \) duplicates transitions of \( A \) to differentiate firing at integral/non integral dates, plus transitions that make time elapsing visible using the additional clock which is reset at each global integral time stamp.

**Definition 24 (Marked Timed Automaton (MA)).** Given a timed automaton \( A = (L, L_0, X, \Sigma, T, F) \) the Marked Timed Automata of \( A \) is a tuple \( MA = (L', L'_0, X \cup \{ n \}, \Sigma \cup \{ \varphi, \delta \}, T', F') \) such that

1. \( n \not\in X \)
2. \( L' = L_0^0 \cup L^+ \) where for \( \alpha \in \{0, +\}, L^\alpha = \{ l^\alpha \mid l \in L \} \)
3. \( L'_0 = \{ l^0 \mid l \in L_0 \} \)
4. \( F' = \{ l^0, l^+ \mid l \in F \} \) and
5. \( T' \) is defined as follows,

\[
T' = \{ (l^0, a, g \wedge n = 0?, R, l^0) \mid (l, a, g, R, l') \in E \}
\cup \{ (l^+, a, g \wedge n < 1?, R, l^+) \mid (l, a, g, R, l') \in E \}
\cup \bigcup_{l \in L} \{ (l^0, \delta, 0 < n < 1?, \emptyset, l^+) \}
\cup \bigcup_{l \in L} \{ (l^+, \varphi, n = 1?, \{ n \}, l^0) \}
\]

Then we have the following results.

**Theorem 25 ([18] Thm.5).** Let \( A \) be a timed automaton and \( MA \) be its marked automaton. Then \( \text{Unt}(L(MA)) = f(L(A)) \).

**Remark 26.** The marked timed automaton of an IRTA is also an IRTA.

The proofs of resilience for IRTA will also rely on the following properties.

**Theorem 27 (Thm.3, [18]).** If \( A \) is an IRTA and \( f(w) = f(w') \), then \( w \in L(A) \) if and only if \( w' \in L(A) \).

**Lemma 28.** The timed suffix language of an IRTA \( A \) can be recognized by an \( \varepsilon \)-IRTA \( A^S \).

**Proof.** Let \( A = (L, X, \Sigma, T, G, F) \) be a timed automaton. We create an automaton \( A^S = (L^S, X, \Sigma \cup \{ \varepsilon \}, T^S, G, F) \) as follows. We set \( L^S = L \cup L_\varepsilon \), where \( L_\varepsilon = \{ l_\varepsilon \mid l \in L \} \) i.e., \( L^S \) contains a copy of locations in \( A \) and another “silent” copy. The initial location of \( A^S \) is \( l_{0, \varepsilon} \). We set \( T^S = T \cup T_\varepsilon \cup T'_\varepsilon \), where \( T_\varepsilon = \{ (l_\varepsilon, \varepsilon, \text{true}, \emptyset, l) \mid l \in L \} \) and \( T'_\varepsilon = \{ (l_\varepsilon, \varepsilon, g, R, l') \mid \exists (l, a, g, R, l') \in T \} \). Clearly, for every timed word \( w = (a_1, d_1) \ldots (a_i, d_i) (a_{i+1}, d_{i+1}) \ldots (a_n, d_n) \) of \( L(A) \) and index \( i \), the word \( w' = (\varepsilon, d_1) \ldots (\varepsilon, d_i) (a_{i+1}, d_{i+1}) \ldots (a_n, d_n) = (a_{i+1}, d_{i+1}) \ldots (a_n, d_n) \) is a recognized by \( A^S \), and it is easy to verify that \( A^S \) is an \( \varepsilon \)-IRTA.

**Lemma 29.** For two IRTA \( A \) and \( B \) and their corresponding marked automata \( A_M \) and \( B_M \), \( L(A) \subseteq L(B) \) if and only if \( \text{untime}(L(A_M)) \subseteq \text{untime}(L(B_M)) \).

**Proof.** \( \Rightarrow \) Assume, \( L(A) \subseteq L(B) \) and assume there exists a word \( w \in \text{untime}(L(A_M)) \), but \( w \notin \text{untime}(L(B_M)) \). Now, there exists a timed word \( \rho \in L(A) \) such that, \( f(\rho) = w \). Clearly,
Resilience of Timed Systems

Figure 8 The automaton $B$ (left) and the faulty automaton $B_P$ (right)

$\rho \in \mathcal{L}(B)$, then clearly $f(\rho) = w \in \text{untimed}(\mathcal{L}(B_M))$ a contradiction. So, $\text{untimed}(\mathcal{L}(A_m)) \subseteq \text{untimed}(\mathcal{L}(B_m))$.

$(\Leftarrow)$ Assume, $\text{untimed}(\mathcal{L}(A_M)) \subseteq \text{untimed}(\mathcal{L}(B_M))$, and $\mathcal{L}(A) \not\subseteq \mathcal{L}(B)$. Then, there exists a timed word $\rho \in \mathcal{L}(A)$ such that $\rho \notin \mathcal{L}(B)$. Assume $f(\rho) = w$, then clearly, $w \in \text{untimed}(\mathcal{L}(A_M))$ and $w \in \text{untimed}(\mathcal{L}(B_M))$. So, there exists a timed word $\rho' \in \mathcal{L}(A)$ such that, $f(\rho') = w = f(\rho)$. According to Theorem 27 we can conclude that, $\rho \in \mathcal{L}(B)$ a contradiction.

Remark 30. Lemma 29 shows that the timed and untimed language inclusion problems for IRTA are in fact the same problem. So, we can solve the timed language inclusion problem by solving an untimed language inclusion problem of IRTA and vice-versa, the untimed language inclusion for IRTA is also EXPSPACE-Complete.

Theorem 31. Timed $K$-$\forall$-resilience of IRTA is EXPSPACE-Hard.

Proof. The proof is obtained by a reduction from the language inclusion problem of IRTA, known to be EXPSPACE-Complete [4]. The idea of the proof follows the same lines as the untimed $K$-$\forall$-resilience of timed automata. Assume we are given IRTA $A_1, A_2$. $a, b, c$ are symbols not in the alphabets of $A_1, A_2$. Consider $B$ in Figure 8 (left). It is easy to see that $L(B) = (a, 1)(b, 1)(c, 11)(L(A_1), 11)$, where $L(A_1) + k = \{(a_1, d_1 + k)(a_2, d_2 + k)\ldots(a_n, d_n + k) | (a_1, d_1)\ldots(a_n, d_n) \in L(A_1)\}$. Associate a fault model $P(a) = 1$, where the fault of $a$ is 1.

We construct an IRTA $B_P$ as shown in Figure 8 (right). Notice that in general, IRTAs are not closed under the fault operation; the enlarged guard in $B$ would read $1 \leq x \leq 2$, and reset $y$. This transition violates the integer reset condition; however, since the transition on $1 < x < 2$ resetting $y$ clearly does not lead to acceptance in $B_P$, we prune away that transition resulting in $B_P$ as in Figure 8 (right). Indeed, this resulting faulty automaton is an IRTA.

The language accepted by $B_P$ is $L(B) \cup (a, 2)(b, 2)(c, 11)(L(A_2) + 11)$. Considering $K = 2$, $B_P$ is BTN in 2 steps after the fault if and only if $L(A_2) \subseteq L(A_1)$. The EXPSPACE hardness of the timed $K$-$\forall$-resilience of IRTA follows from the EXPSPACE completeness of the inclusion of IRTA.

Theorem 32. $K$-$\exists$-resilience for IRTA is PSPACE-Hard.

Proof. Consider an IRTA $A$ with alphabet $\Sigma$ and the construction of an automata that uses a gadget shown below in Figure 9 (left). Let us call this automaton $B_{\Sigma \subseteq A}$. It is easy to see that the $L(B_{\Sigma \subseteq A}) = (a, 1)(b, 1)(c, 11)((\Sigma \times \mathbb{R})^* + 11)$, where $L(A_1) + k = \{(a_1, d_1 + k)(a_2, d_2 + k)\ldots(a_n, d_n + k) | (a_1, d_1)\ldots(a_n, d_n) \in L(A_1)\}$. The $\Sigma$ loop on a particular accepting state $q_e$ is responsible for acceptance of all timed word. Now, associate a
fault model $\mathcal{P}(a) \rightarrow 1$ with $B$, where the fault of $a$ is 1. Let us call this enlarged automaton $B(\Sigma^* \subseteq A)_P$. We can prune away the transition $1 < x < 2$ resetting $y$ which does not lead to acceptance, and resulting in an IRTA with the same language, represented in Figure 9 (right). The language accepted by $B(\Sigma^* \subseteq A)_P$ is $L(B(\Sigma^* \subseteq A)) \cup (a, 2)(b, 2)(c, 11)(L(A) + 11)$. The accepting states are $q_e \cup F$, where $F$ is the set of final states of $A$. Then $B(\Sigma^* \subseteq A)_P$ is $K$-$3$-resilient if and only if $L(A) \neq \emptyset$.

\[ \text{Figure 9} \text{ The IRTA } B(\Sigma^* \subseteq A) \text{ (left) and the faulty IRTA } B(\Sigma^* \subseteq A)_P \text{ (right)} \]

Remark 33. The untimed language inclusion problem is shown to be EXPSPACE-Complete in Remark 30. The emptiness checking of timed automata is done by checking the emptiness of its untimed region automaton. So, to show the hardness of untimed $K$-$\forall$-resilient or $K$-$3$-resilient problems for IRTA, it is sufficient to reduce the untimed language inclusion problem and untimed language emptiness problem of IRTA respectively. This reduction can be done by using the same gadget as shown in Theorem 31 and Theorem 32 respectively.