Resilience of Timed Systems

Sundararaman Akshay, Blaise Genest, Loïc Hélouët, Shankara Krishna, Sparsa Roychowdhury

Resilience of Timed Systems

S. Akshay[1], B. Genest[2,3], L. Hélouët[2,4], S. Krishna[1], and S. Roychowdhury[1]

[1] IIT Bombay, Bombay, India
[2] Univ. Rennes, Rennes, France
[3] CNRS
[4] INRIA
akshayss@gmail.com
blaise.genest@irisa.fr
loic.helouet@inria.fr krishnas@cse.iitb.ac.in
sparsa@cse.iitb.ac.in

Abstract. Imperfections in timed systems has often been addressed as a robustness problem asking whether a timed system preserves some properties despite imperfect time measurement. Robustness, however, considers systems as reliable if there exists even a infinitesimal amount of perturbation for which a system preserves its properties.
This paper addresses correctness of timed systems in a different setting called *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, we say that a system is *universally resilient* if and only if it always returns to a timed behavior of the non-faulty system. It is *existentially resilient* for a given fault model if and only if it can return to a timed behavior of the non-faulty system. We show that checking universal resilience of timed automata is undecidable, but existential resilience is in EXPSPACE. To obtain decidability and better complexity bounds, we consider untimed versions of the problem as well as known subclasses of timed automata.

Keywords: Timed automata; Fault tolerance; Integer-resets; Resilience

1 Introduction

Timed automata are a natural model to represent cyber-physical systems with real-time constraints. Originally introduced in the 90's [2], they have led to an enormous body of work in the past 30 years, both from theoretical and practical points of view. 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 constraints on the values of these clocks, which allows for the modeling of real-time issues, such as the time elapsed between the occurrence of two events. Clocks, guards, and resets give timed automata a huge expressive power, but result in several simple properties such as language inclusion being undecidable. A natural question when modeling real-time systems is whether the considered system can handle unexpected delays. This is particularly true when modeling systems equipped with a priori schedules such as

trains, metros, etc., buses. In this paper, we are interested in 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.

Timed automata are not a priori tailored to handle unspecified behaviors : guards are strict 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. A second drawback is that 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. In real implementations, one cannot trigger an event at an exact date: clocks may have some imprecision, and some time elapses between a measurement, the decision to perform an action, and the effective realization of that action [10]. To address this issue, and guarantee soundness of results verified on timed automata, several works have considered the notion of *robustness* [10, 8, 7]. A property of a system is considered robust if and only if there exists an infinitesimal value ϵ such that an imprecision of ϵ in time measurement still preserves the considered property. Robustness is often addressed by considering guard enlargement, i.e., replacing guards of the form $x \in [a, b]$ by $x \in [a-\epsilon, b+\epsilon]$, and then checking that some value for ϵ preserves the considered property. It was shown [7] that robustness of ω-regular languages is a decidable problem. [12] addresses robustness through the definition of *robust automata* that accept timed words of an automaton and their neighbors i.e., words with timing differences that remain at a close distance. In [13, 11, 16, 1], the authors consider robustness via modeling clock drifts. The diagnosis of faults using timed automata has been studied in [6].

However, in all these cases, robustness addresses delays by considering that events can occur early/late with an arbitrarily small variation e.g., a timing prescribed by a guard. As a result the goal is often to check that the perturbed behaviours are close to those of the specification. Typically, this is not the scenario in resilience, where the question is to recover from (a possibly large) deviation from specified behavior. In other words, resilience asks whether a system can behave as originally specified after an event is delayed by some possibly large value δ. We start by capturing delayed events by new faulty transitions. As these events occur at dates that were not originally planned, a fault may affect values of clocks for an arbitrarily long time, letting the system diverge from its specified behavior. Then, a system is resilient if it stops diverging a finite number of steps after a fault. More precisely, we define two variants. A timed automaton is $K - \forall - resilient$ if for every fault, K steps after this fault, the behavior of the system cannot be distinguished from a non-faulty behavior. A timed automaton is $K - \exists - resilient$ if for every faulty run, K steps after the fault, there exists a run in which the system behaves as the original automaton. We consider universal and existential resilience in the timed and untimed setting.

Our results are summarized in Table 1. We start by showing that universal timed resilience can be brought back to a comparison of timed languages, so unsurprisingly, it is undecidable in general. But for an interesting subclass of

	Timed $K - \forall -$ RES	Timed $K - \exists -$ RES
TA	Undecidable (Thm. 3)	EXPSPACE (Thm. 4) PSPACE-Hard (Thm. 5)
IRTA	EXPSPACE-C (Thm. 8)	PSPACE-C (Thm. 9)
	Untimed $K - \forall -$ RES	Untimed $K - \exists -$ RES
TA	EXPSPACE-C (Thm. 1)	PSPACE-C (Thm. 2)
IRTA	EXPSPACE-C (Thm. 7)	PSPACE-C (Thm. 2 and Rmk 4)

Table 1. Summary of results for resilience

timed automata, namely, Integer Reset timed automata (IRTA)[15], it turns out that universal resilience is EXPSPACE-Complete. On the other hand, we show that existential timed resilience is in fact decidable in EXPSPACE in the general case, and is PSPACE-Complete for IRTAs. We also consider untimed variants of these problems (i.e., the recovery only needs to be in terms of the actions seen and not the time-stamps). In this setting, we show that universal resilience is decidable and EXPSPACE-Complete in general, while existential untimed resilience is PSPACE-Complete, even for IRTAs.

2 Preliminaries

Let Σ be a finite alphabet. A timed word is an element of $(\Sigma \times \mathbb{R}_{\geq 0})^*$ of the form $w = (a_1, d_1) \dots (a_n, d_n)$ such that for every i, $d_i \leq d_{i+1}$. A language L is a subset of Σ^*, and a timed language is a set of timed words. A prefix of w is a word w_1 such that $w = w_1.w_2$. A suffix of w is a word w_2 such that $w = w_1.w_2$. We denote by $w_{[i,j]}$, for $i \leq j$ the sequence $(a_i, d_i) \dots (a_j, d_j)$. The untiming of w is the untimed word $Unt(w) = a_1.a_2 \dots a_n \in \Sigma^*$. The untiming of a timed language L is the set of all untimings of timed words in L. A clock is a real-valued variable x and an atomic clock constraint is an inequality of the form $a \bowtie_l x \bowtie_u b$, where $\bowtie_l, \bowtie_u \in \{\leq, <\}$, $a, b \in \mathbb{N}$. Guards are conjunctions of atomic clock constraints on a set X of clocks.

Definition 1 (timed automaton). *A timed automaton[2] is a tuple $\mathcal{A} = (L, L_0, X, \Sigma, T, F)$ where L is a finite set of locations, $L_0 \subseteq L$ is a finite set of initial locations, X is a finite set of clocks, Σ is a finite set of action names, $F \subseteq L$ is a set of final locations, and $T \subseteq L \times \Sigma \times \mathcal{G} \times 2^X \times L$ is a transition relation where $\mathcal{G}$ is the set of guards on X.*

A valuation of a set of clocks X is a map ν that associates each clock in X to a non-negative real value. For every clock x, $\nu(x)$ has an integral part $\lfloor \nu(x) \rfloor$ and a fractional part $fract(\nu(x)) = \nu(x) - \lfloor \nu(x) \rfloor$. We will say that a valuation ν on a set of clocks X satisfies a guard g, denoted $\nu \models g$ iff replacing x by $\nu(x)$ in g yields a tautology. We will denote by $[g]$ the set of valuations that satisfy g. Given $d \in \mathbb{R}$, we denote by $\nu + d$ the valuation that associates value $\nu(x) + d$

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* lets d time units elapse from a configuration $C = (l, \nu)$ which leads to configuration $C' = (l, \nu + d)$. 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 is $\nu \models g$. 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.

Definition 2 (runs, timed words). *A run of $\mathcal{A}$ is a sequence $\sigma \in (T \times \mathbb{R})^*$ such that $\sigma = (t_1, d_1).(t_2.d_2) \dots (t_n, d_n)$ and there exists a sequence of timed and discrete moves $\delta_0.t_1.\delta_1.t_2 \dots \delta_{n-1}.t_n$ allowed in $\mathcal{A}$ such that $d_i = \sum_{j<i} \delta_j$. One can associate a timed word w_σ to every run of $\mathcal{A}$ by letting $w_\sigma = (a_1, d_1).(a_2, d_2) \dots (a_n, d_n)$, where a_i is be the action in transition t_i. A timed word $w = (a_1, d_1).(a_2, d_2) \dots (a_n, d_n)$ is accepted by $\mathcal{A}$ if there exists a run that produces this timed word, and ends in a final location. The timed language of $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A})$ of all its timed words. The untimed language of $\mathcal{A}$ is the language $Unt(\mathcal{L}(\mathcal{A}))$.*

It was shown in [2] that the behaviors of a timed automaton can be captured by an abstraction called the region automaton. Formally, given a clock x, let c_x be the largest constant such that $\mid x \mid \leq c_x$.Let ν, ν' be two valuations of clocks in X. Then ν and ν' are equivalent, written $\nu \sim \nu'$ if and only if, (i) $\forall x \in X$, either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ or both $\nu(x) \geq c_x$ and $\nu'(x) \geq c_x$ (ii) For all $x, y \in X$ with $\nu(x) \leq c_x$, and $\nu(y) \leq c_y$, $fract(\nu(x)) \leq fract(v(y))$ if and only if $fract(\nu'(x)) \leq fract(\nu'(y))$ (iii) For all $x \in X$ with $\nu(x) \leq c_x$, $fract(\nu(x)) = 0$ if and only if $fract(\nu'(x)) = 0$ A region r of $\mathcal{A}$ is the equivalence class induced by $\sim$. For a given automaton $\mathcal{A}$, there exists only a finite number of regions, bounded by 2^K, where K is the size of the constraints set in $\mathcal{A}$. It is well known that for a constraint δ if $\nu \sim \nu'$, then $\nu \models \delta$ if and only if $\nu' \models \delta$. A region r' is a time successor of another region r if for every $\nu \in r$, there exists $d \in \mathbb{R}^{>0}$ such that $\nu + d \in r'$. We denote by $Reg(X)$ the set of all possible regions of the set of clocks X. For a timed automaton $\mathcal{A} = (L, L_0, X, \Sigma, T, F)$, its region automaton is the untimed automaton $\mathcal{R}(\mathcal{A}) = (L \times Reg(X), \Sigma, \Delta, F \times Reg(X))$ that recognizes the same untimed language as $\mathcal{A}$. States of $\mathcal{R}(\mathcal{A})$ are of the form (l, r), where l is a location of $\mathcal{A}$ and r a region. The transition relation Δ is such that $((l, r), a, (l', r')) \in \Delta$ if there exists a transition $t = (l, g, a, R, l')$ such that there exists a region r'' that is a time successor of r and satisfies guard g, and r' is obtained from r'' by resetting values of clocks in R. This is useful in particular, to prove non-emptiness of $\mathcal{L}(\mathcal{A})$, as $\mathcal{L}(\mathcal{A}) \neq \emptyset$ if and only if $\mathcal{R}(\mathcal{A})$ accepts some word.

3 Resilience Problems

In this section, we define the semantics of timed automata when perturbations can delay the occurrence of an action. Let us consider a transition $t = (l, g, a, R, l')$, where $g ::= x \leq 10$. This means that an action a can occur as long as x has not reached a value greater than 10. Now, timed automata have a idealized representation of time, that does not always consider perturbations that occur in real systems. Consider, for instance that a is a physical event planed 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 $\mathcal{L}(\mathcal{A})$ 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 models is used to encode the fact that an action might occur at dates slightly greater than the dates allowed in the original model semantics.

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

For simplicity, we consider only execution in which a single timing error occurs. However, 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, that accepts all executions of $\mathcal{A}$ where at most one transition $t = (l, a, g, r, l')$ may have occurred later than specified in guard g.

Definition 4 (enlargement (of a guard)). *Let ϕ be an inequality of the form $a \bowtie_l x \bowtie_u b$, where $\bowtie_l, \bowtie_u \in \{\leq, <\}$. The enlargement of ϕ by a time error δ is the inequality $\phi_{\triangleright\delta}$ of the form $a \bowtie_l x \leq b + \delta$. Let g be a guard of the form*

$$g = \bigwedge_{i\in 1..m} \phi_i = a_i \bowtie_{l_i} x_i \bowtie_{u_i} b_i \wedge \bigwedge_{j\in 1..q} \phi_j = a_j \bowtie_{l_j} x_j - y_j \bowtie_{u_j} b_j.$$

The enlargement of g by δ is the guard $g_{\triangleright\delta} = \bigwedge_{i\in 1..m} \phi_{i_{\triangleright\delta}} \wedge \bigwedge_{j\in 1..q} \phi_j$

Definition 5 (Faulty transitions). *For every transition $t = (l, g, a, R, l')$ with enlarged guard*

$$g_{\triangleright\mathcal{P}(a)} = \bigwedge_{i\in 1..m} \phi_i = a_i \bowtie_{l_i} x_i \leq b_i + \mathcal{P}(a) \wedge \bigwedge_{j\in 1..q} \phi_j = a_j \bowtie_{l_j} x_j - y_j \bowtie_{u_j} b_j, \text{ we can create a new transition } t_{f,\mathcal{P}} = (l, g_{f,\mathcal{P}}, a, R, \bullet l') \text{ called a faulty transition such that}$$

$$g_{f,\mathcal{P}} = \bigwedge_{i\in 1..m} \phi_i = b_i \bar{\bowtie}_{l_i} x_i \leq b_i + \mathcal{P}(a) \wedge \bigwedge_{j\in 1..q} \phi_j = a_j \bowtie_{l_j} x_j - y_j \bowtie_{u_j} b_j \text{ with } \bar{\bowtie}_{l_i} \in \{<, \leq\} \setminus \bowtie_{l_i}$$

When the fault model is clear from the context, we will simply write t_f and g_f instead of $t_{f,\mathcal{P}}$ and $g_{f,\mathcal{P}}$. Clearly, g and g_f are disjoint, and $g \vee g_f$ is equivalent to $g_{\triangleright\delta}$. For example, consider a transition of the form (l, a, g, R, l'), with a guard $g = 3 < x < 12 \wedge y > 5$. Let $\mathcal{P}(a) = 2$ then $g_{\triangleright\mathcal{P}(a)} = 3 < x \leq 14 \wedge y > 5$ and $g_f = 12 \leq x \leq 14 \wedge y > 5$.

Definition 6 (enlargement of automata). *Let $\mathcal{A} = (L, L_0, X, \Sigma, T, F)$ be a timed automaton, and $\mathcal{P}$ be a fault model. The enlargement of $\mathcal{A}$ by $\mathcal{P}$ is the automaton $\mathcal{A}_\mathcal{P} = (L_\mathcal{P}, L_0, X, \Sigma, T_\mathcal{P}, F_\mathcal{P})$, where*

- $L_\mathcal{P} = L \uplus \{\overset{\bullet}{l} \mid l \in L\}$. *A location $\overset{\bullet}{l}$ indicates that an unexpected delay has occurred.*
- $T_\mathcal{P} = T \cup \overset{\bullet}{T}$ *such that,* $\overset{\bullet}{T} = \{(l, g_f, a, R, \overset{\bullet}{l'}) \mid (l, g, a, R, l') \in T\} \cup \{(\overset{\bullet}{l}, g, a, R, \overset{\bullet}{l'}) \mid (l, g, a, R, l') \in T\}$
- $F_\mathcal{P} = F \cup \{\overset{\bullet}{l} \mid l \in F\}$

A run of $\mathcal{A}_\mathcal{P}$ is said to be faulty if it contains a transition of $\overset{\bullet}{T}$.

Definition 7. *Let $w \in \mathcal{L}(\mathcal{A}_\mathcal{P})$ be a timed word. Then, w is faulty at step i if and only if $w_{[1..i-1]}$ is a prefix of some word in $\mathcal{L}(\mathcal{A})$ and $w_{[1..i]}$ is not. We denote by $\mathcal{L}^{1F}(\mathcal{A}_\mathcal{P})$ the language of faulty timed words w that are faulty at step $|w|$ (i.e. the word ends with a fault). We denote by $\mathcal{L}^{F}(\mathcal{A}_\mathcal{P})$ the language of faulty timed words accepted by $\mathcal{A}_\mathcal{P}$.*

Remark 1. One can immediately notice that a timed word recognized by a faulty run is not necessarily a faulty word, as there might be several runs reading the same sequence of actions but through a different sequence of transitions in T^*.

Definition 8 (Back to Normal). *Let $w \in \mathcal{L}(\mathcal{A}_\mathcal{P})$ be a timed word that is faulty at step i. w is said to be back to normal (BTN) after k steps if and only if $w_{[i+k..n]}$ is a suffix of some word in $\mathcal{L}(\mathcal{A})$, i.e., if there exists $w' \in \mathcal{L}(\mathcal{A})$ such that $w_{[i+k..n]} = w'_{[i+k..n]}$*

The notion of back to normal can be recast in the untimed setting. A faulty word is back to normal in the untimed sense k steps after a fault iff it is faulty at some step i, and there exists $w' \in \mathcal{L}(\mathcal{A})$ such that $Unt(w_{[i+k..n]}) = Unt(w'_{[i+k..n]})$. We are now ready to define several notions of resilience for timed automata.

Definition 9 (resilience). *Let $K \in \mathbb{N}$, $\mathcal{A}$ be a timed automaton with fault model $\mathcal{P}$.*

- *We say that $\mathcal{A}$ is $K - \forall -$ resilient if for every faulty word w in $\mathcal{L}^F(\mathcal{A}_\mathcal{P})$, w is back to normal (BTN) in K steps. In other words, $\mathcal{A}$ is $K - \forall -$ resilient if and only if for every $w \in \mathcal{L}^{1F}(\mathcal{A}_\mathcal{P})$ and for every w' such that $w.w' \in \mathcal{L}(\mathcal{A}_\mathcal{P})$, $w.w'$ is BTN in K steps.*
- *We say that $\mathcal{A}$ is $K - \exists -$ resilient if and only if, for every faulty word w in $\mathcal{L}^{1F}(\mathcal{A}_\mathcal{P})$, there exists w' such that $w.w' \in \mathcal{L}(\mathcal{A}_\mathcal{P})$ and $w.w'$ is BTN in K steps.*

As for the notion of BTN words, we can define $K - \{\forall, \exists\} - resilience$ in the untimed sense, by requiring that the BTN notion used in Definition 9 refers to untimed comparison of words. For instance, $\mathcal{A}$ is (untime) $K - \forall - resilient$ if

and only if for every word w in $\mathcal{L}(\mathcal{A}_\mathcal{P})$ faulty at some step i, there exists a word w' in $\mathcal{L}(\mathcal{A})$ such that $Unt(w_{[i+k..n]}) = Unt(w'_{[i+k..n]})$.

Note that $K - \forall - resilience$ implies $K - \exists - resilience$. In case of $K - \forall - resilience$, any faulty word w can be seen to be BTN in $\leq K$ steps after a fault occurring at step i. This implies $K - \exists - resilience$ since any word $w \in \mathcal{L}^{1F}(\mathcal{A}_\mathcal{P})$ is the prefix of a word $w.v$ that is BTN in less than K steps. Obviously, the converse does not hold. Indeed, $\mathcal{L}(\mathcal{A}_\mathcal{P})$ can contain a pair of words $w_1 = w.v_1$,$w_2 = w.v_2$ where $w = a_1 \dots a_n \in \mathcal{L}^{1F}(\mathcal{A}_\mathcal{P})$, and such that w_1 is BTN in K steps, witnessing existential resilience, while w_2 is not. We can however show that resilience properties are monotonic.

Proposition 1 (monotonicity). *Let $K \in \mathbb{N}$. A timed automaton that is $K - \forall - resilient$ (resp. $K - \exists - resilient$) is $(K+1) - \forall - resilient$ (resp. $(K+1) - \exists - resilient$).*

The proof is straightforward : as $(a_{i+k}, d_{i+k}) \dots (a_{i+n}, d_{i+n})$ is a suffix of a word in $\mathcal{L}(\mathcal{A})$, then so is $(a_{i+k+1}, d_{i+k+1}) \dots (a_{i+n}, d_{i+n})$. The converse is not true: a system that is $K-resilient$ need not be $(K-1)-resilient$. Monotonicity allows to address $\forall$ and $\exists$ resilience questions without fixing a bound K. Indeed, for the decidable cases, resilience will be decided on finite structures from which a maximal value for K can be derived.

Example : We model reactions of a braking system in a self driving car. The system detects pedestrians and decides whether to start braking or not. The braking decision system is defined by a timed automaton $\mathcal{B}_s$ represented Figure 1. It contains four important states: From s_0 the system detects pedestrians, from s_1 it must make a decision, from s_2 it starts braking, and from s_3 it stops the car. Action 'p' represents arrival of a pedestrian, 'd' represents the detection, 'b' represents the brake deployment and '$stop$' represents the stopping of the car. The system can spend at most 1 time unit from s_1 to s_2, from s_2 to s_3, and from s_3 to s_4. It means that time elapsed between detection of a pedestrian and decision must be short. Similarly the time spent between the decision to brake and braking takes less than 1 time unit, and time elapsed between braking start and car stop is also smaller than 1 time unit. Now, assume that detection can occur late. We define a fault model that can delay detection $\mathcal{P}(d) = 1$. The fault model enlarges the constraint $x \leq 1$ to $x \leq 2$, which adds to $\mathcal{A}_s$ the faulty transition $(s_1, d, 1 < x \leq 2, \{x_1\}, s_2)$, shown in red in Fig. 1, that can only fire when the detection takes longer than 1 unit of time. We denote by $\mathcal{A}_s^f$ the enlarged automaton. $\mathcal{A}_s$ is $0 - \exists - resilient$ but, not $0 - \forall$-resilient. Indeed, the detection can be delayed by up to 1 time units, i.e., faulty words in $\mathcal{L}^{1F}(\mathcal{A}_s^f)$ will be of the form $(p,0).(d, 1+\delta)$. Now, any of these words can be extended to be accepted by $\mathcal{A}_s^f$, i.e. any word of the form $(p,0).(d,1+\delta).(b,2+\delta).(stop,3+\delta)$. However, not all of these words are BTN in 0 steps. For example, the run $(p,0).(d,2).(b,2).(stop,3)$ is BTN because $(p,0).(d,1).(b,2).(stop,3)$ is a timed word of $\mathcal{A}_s$. However, the faulty word $(p,0).(d,2).(b,3).(stop,4)$ is accepted by $\mathcal{A}_s^f$, but no run of $\mathcal{A}_s$ ends with $(b,3).(stop,4)$, so $\mathcal{A}_s$ is not 0-$\forall$-resilient.

Now, every word $(p,0).(d,1+\delta) \in \mathcal{L}^{1F}(\mathcal{A}_s^f)$ is a prefix of word $(p,0).(d,1+\delta).(b,2).(stop,3)$ which is BTN. So $\mathcal{A}_s$ is 0-$\exists$-resilient.

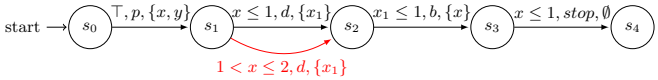


Fig. 1. Example model of brake subsystem $\mathcal{A}_s$.

We observe that given an integer K, and a region automaton $\mathcal{R}(\mathcal{A}_\mathcal{P})$ for $\mathcal{A}_\mathcal{P}$, we can easily compute an automaton $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ that counts if at least K transitions have occurred since the occurrence of a fault. States of $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ are of the form (l,r,K) or of the form $(\overset{\bullet}{l},r,c)$ with $0 \le c \le K$ when they are reached after a fault. Transitions are of the form:

- $((l,r,K),a,(l',r',K))$ if $((l,r),a,(l',r'))$ is a transition of $\mathcal{R}(\mathcal{A}_\mathcal{P})$,
- $\left((l,r,K),a,(\overset{\bullet}{l'},r',K)\right)$ if $\left((l,r,),a,(\overset{\bullet}{l'},r')\right)$ is a faulty transition of $\mathcal{R}(\mathcal{A}_\mathcal{P})$,
- If $\left((\overset{\bullet}{l},r),a,(\overset{\bullet}{l'},r')\right)$ is a transition of $\mathcal{R}(\mathcal{A}_\mathcal{P})$, then we add the transitions
  - $\left((\overset{\bullet}{l},r,c),a,(\overset{\bullet}{l'},r',c-1)\right)$ if $c>0$, and
  - $\left((\overset{\bullet}{l},r,0),a,(\overset{\bullet}{l'},r',0)\right)$, that are transitions that occur more than K steps after the initial fault.

One can already remark that for every state (l,r) such that there exists a transition $\left((\overset{\bullet}{l}{}',r',1),a,(\overset{\bullet}{l},r,0)\right)$, the words recognized from (l,r) are the suffixes that need to be compared to suffixes of words in $\mathcal{L}(\mathcal{A})$ to check resilience. We will use this property in Section 4.

4 Untimed Resilience

Theorem 1. *Untimed $K-\forall-$resilience is EXPSPACE-complete.*

Proof. *Membership:* We first show that K-$\forall$-resilience is in EXPSPACE. Let $\mathcal{A}$ be a timed automaton, and $\mathcal{P}$ be a fault model. The enlarged automaton is $\mathcal{A}_\mathcal{P}$ and its region automaton is $\mathcal{R}(\mathcal{A}_\mathcal{P})$. Note that states of the enlarged region automaton are of the form (l,r) if no fault has occurred yet, and $(\overset{\bullet}{l},r)$ if a fault has occurred.

We build an automaton $\mathcal{B}$ that recognizes all faulty words of $\mathcal{A}_\mathcal{P}$ that are not BTN after K steps. We synchronize $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ and $\mathcal{R}(\mathcal{A})$ the following way: States of $\mathcal{B}$ are of 3 types : (i) states of the type $(l,r,K,\emptyset)$, represent states

(l, r, K) of $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ reached without using a faulty transition, and (ii) states of the type $(\overset{\bullet}{l}, r, c, \emptyset)$, with $c > 0$, represent states $(\overset{\bullet}{l}, r, c)$ of $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ reached through a faulty run, but where the fault occurred less than K steps ago. (iii) Last, we have states of the form $(\overset{\bullet}{l}, r, 0, X)$ where $X \subseteq L \times Reg(X)$ is a subset of states of $\mathcal{R}(\mathcal{A})$ that are accessible by recognizing an untimed word that is played by $R^K(\mathcal{A}_\mathcal{P})$ from the K^{th} steps after a fault. This set X is a standard subset construction, and is built inductively with the transition relation shown below.

- There is a transition from $(l, r, K, \emptyset)$ to $(l', r', K, \emptyset)$ with label a if there is a transition from (l, r, K) to (l', r', K) with letter a in $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$. This transition mimics the behavior of $\mathcal{A}$ before a fault, we do not remember states of $\mathcal{R}(\mathcal{A})$, as there is no need to compare the current run of $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ to an untimed suffix of $\mathcal{L}(\mathcal{A})$.
- There is a transition from $(l, r, K, \emptyset)$ to $(\overset{\bullet}{l'}, r', K, \emptyset)$ with label a if there is a transition from (l, r, K) to $(\overset{\bullet}{l'}, r', K)$ with letter a in $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$. This transition represents occurrence of a fault in $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$. There is still no need to compare the current run of $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ to an untimed suffix of $\mathcal{L}(\mathcal{A})$, but from there, the counter value will decrease to count steps occurred after the fault.
- There is a transition from $(\overset{\bullet}{l}, r, c, \emptyset)$ to $(\overset{\bullet}{l'}, r', c-1, \emptyset)$ with label a if there is a transition from $(\overset{\bullet}{l}, r, c)$ to $(\overset{\bullet}{l'}, r', c-1)$ with letter a in $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$. This transition is used to remember that a fault has occurred, and to count the number of steps that have occurred since the fault.
- There is a transition from $(\overset{\bullet}{l}, r, 1, \emptyset)$ to $(\overset{\bullet}{l'}, r', 0, L \times Reg(X))$ with label a if there is a transition from $(\overset{\bullet}{l}, r, 1)$ to $(\overset{\bullet}{l'}, r', 0)$ with letter a in $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$. This transition represents what occurs K steps after a fault and from then, the faulty runs should coincide with some suffix of a word of $\mathcal{L}(\mathcal{A})$.
- There is a transition from $(\overset{\bullet}{l}, r, 0, X)$ to $(\overset{\bullet}{l'}, r', 0, Y)$ with label a and sets of states $X, Y \subseteq L \times Reg(X)$ if there exists a transition from $(\overset{\bullet}{l}, r, 0)$ to $(\overset{\bullet}{l'}, r', 0)$ with letter a in $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$, and for every $y = (l_y, r_y) \in Y$, there exists $x = (l_x, r_x) \in X$ such that $((l, r), a, (l', r'))$ is a transition of $\mathcal{R}(\mathcal{A})$.

Intuitively, components l, r, c of states copy transitions of $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ and component X synchronizes transitions of $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ with transitions of $\mathcal{R}(\mathcal{A})$ on common letters, performing a standard subset construction. The accepting states of $\mathcal{B}$ are states of the form $(\overset{\bullet}{l}, r, 0, X)$ where l is an accepting location (and hence so is $\overset{\bullet}{l}$) and X contains only pairs of the form (l_x, r_x) with $l_x \notin F$. This means that no suffix of a word of $\mathcal{A}$ is able to match with the untimed suffix of the faulty word currently recognized.

Lemma 1. *Let ρ be a run that leads to an accepting state of $\mathcal{B}$. Then, the word recognized by ρ is not BTN.*

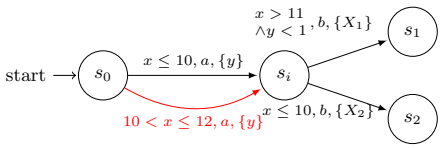


Fig. 2. A gadget timed automaton $\mathcal{G}$

A proof of this lemma can be found in Appendix B. If a word w is recognized by $\mathcal{B}$, then it is an untimed faulty word that is not BTN after K steps. Hence, $\mathcal{A}$ is $K - \forall - resilient$ if and only if $L(\mathcal{B}) = \emptyset$. The size of $\mathcal{R}(\mathcal{A})$ and $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ is in $O(2^{|X|})$. Hence, the size of $\mathcal{B}$ is in $O(2^{2^{|X|}})$. Now, proving that $L(\mathcal{B}) \neq \emptyset$ is a reachability question, which is known to be solvable in space that is logarithmic w.r.t. the size of the considered automaton. So, one can find a path to an accepting state of $\mathcal{B}$ in EXPSPACE.

Hardness : We now show the hardness by reducing the untimed language inclusion problem of timed automata to an untimed K-$\forall$-resilience problem. Consider the gadget automaton $\mathcal{G}$ in Figure 2 with four states s_0, s_i, s_1, s_2, and two sets of clocks $X = X_1 \uplus X_2$. Let us equip this automaton with a fault model $\mathcal{P} : a \to 2$ that can delay action a of up to 2 time units. In this gadget, s_1 can be reached only when a fault happens, and s_2 can be reached when no fault happens. Let us connect two timed automata $\mathcal{A}_1 = (L_1, \{l_0^1\}, X_1, \Sigma_1, T_1, F_1)$ and $\mathcal{A}_2 = (L_2, \{l_0^2\}, X_2, \Sigma_2, T_2, F_2)$ with respective sets of clocks X_1 and X_2 to this gadget by collapsing their initial states respectively l_0^1 with s_1 and l_0^2 with s_2, and let us call this automaton $\mathcal{G} \cup \mathcal{A}_1 \cup \mathcal{A}_2$.

Then when the fault happens, the automaton $\mathcal{G} \cup \mathcal{A}_1 \cup \mathcal{A}_2$ accepts words whose untimings are of the form $a.b.w_1$, where $w_1 \in Unt(\mathcal{L}(\mathcal{A}_1))$. Similarly, if no fault happens then it accepts words whose untimings are of the form $a.b.w_2$ where $w_2 \in Unt(\mathcal{L}(\mathcal{A}_2))$. If we require the maximal number of steps to occur after a fault to be $K = 1$, then every untimed sub-word of $\mathcal{B}_\mathcal{P}$ after b has to be either in $Unt(\mathcal{L}(\mathcal{A}_2))$ or in $Unt(\mathcal{L}(\mathcal{A}_1))$.

Then, solving untime 1-$\forall$-resilience for $\mathcal{G} \cup \mathcal{A}_1 \cup \mathcal{A}_2$ amounts to solving the untimed language inclusion problem $Unt(\mathcal{L}(\mathcal{A}_1)) \subseteq Unt(\mathcal{L}(\mathcal{A}_2))$, which is known to be EXPSPACE-Complete [9].

Theorem 2. *Untimed K-$\exists$-resilience is PSPACE-Complete.*

Proof. Containment : We show that we can find a witness of violation of K-$\exists$-resilience in PSPACE, if it exists. We check if there exists a path ρ of $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ that ends in a state $q = (l, r)$, which is K steps after a fault (we enumerate over all possibilities of q), such that from q, for every path that reads w and reaches an accept state in $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$, from every state of $R(\mathcal{A})$ (again we enumerate over

all states of the region automaton), reading w does not allow to reach an accept state of $\mathcal{R}(\mathcal{A})$.

One needs to visit each pair $q = (l, r)$ of location and region in $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ accessible K steps after a fault, and check if there exists a state (l_i, r_i) in $\mathcal{R}(\mathcal{A})$, accepting runs ρ from (l, r) and ρ_i from (l_i, r_i) such that ρ and ρ_i recognize the same untimed word. This is sufficient, as existential untimed resilience compares all untimed suffixes K steps after a fault with all untimed suffixes of $\mathcal{A}$.

Using an ordering on regions (such as the order proposed in Appendix A), one can enumerate states of $\mathcal{R}^K(\mathcal{A}_\mathcal{P})$ in PSPACE, and check that they are accessible from (l_0, r_0) K steps after a fault. One can also enumerate pairs (l_i, r_i) in PSPACE, and check that (l_i, r_i) is accessible from (l_0, r_0). Then, for each state (l, r) and state (l_i, r_i), one can check in PSPACE whether there exists a run ρ starting from (l, r) and a run ρ_i starting from (l_i, r_i) such that $Unt(w_\rho) = Unt(w_{\rho_i})$. This can be done through a joint exploration of pairs of states (l', r') and (l_j, r_j) that are accessible from $(l, r), (l_i, r_i)$, and such that l' and l_j are accepting locations. As the size of the region automata $\mathcal{R}(\mathcal{A})$ and $\mathcal{R}(\mathcal{A}_\mathcal{P})$ are bounded (and of exponential sizes), such an exploration may need an exponential number of steps, but needs only logarithmic space (wrt the exponential region automata) to check reachability of F from (l, r) and (l_i, r_i). This can hence be done in PSPACE. In case of success, then (l, r) is not a witness of non-existential resilience, and one can consider the next state of $\mathcal{R}(\mathcal{A}_\mathcal{P})$ according to ordering. In case of failure with (l_i, r_i) one can resume search of a common word with the state of $\mathcal{R}(\mathcal{A})$ that immediately follows (l_i, r_i). If the search for a common word accepted fails for (l, r) with all (l_i, r_i), then (l, r) is a witness of violation of K-existential resilience. Once all pairs (l, r) have been explored without finding a witness of non-existential resilience, one can claim that $\mathcal{A}$ is K-$\exists$-resilient. To run this algorithm, one needs to remember the current state (l, r) explored in $\mathcal{R}(\mathcal{A}_\mathcal{P})$, the state (l_i, r_i) from which we try to find a matching word, the current states (l', r') and (l_j, r_j) explored in the followed pair of paths, and the number of steps performed since the beginning of this path. This requires only polynomial space.

Hardness : We can now show that untimed K-$\exists$-resilience is PSPACE hard. Consider a timed automaton $\mathcal{A}$ with alphabet Σ and the construction of an automata that uses a gadget shown below in Figure 3. Let us call this automaton $\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}$. 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_e. If $\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}$ takes the faulty transition(marked as red) then it resets all clocks of $\mathcal{A}$ and behaves as $\mathcal{A}$. The accepting states are $q_e \cup F$. Then, if $\mathcal{A}$ has an accepting word, $\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}$ is untimed K-$\exists$-resilient.

Remark 2. Notice that the hardness reduction in the proof of Thm. 2 holds even for deterministic timed automata. Indeed, the construction of Fig. 3 is deterministic (the faulty and normal transitions have disjoint guards, the universal timed language over Σ is recognized by a determinisitic automaton, so the existential resilience problem can encode a non-emptiness problem for deterministic

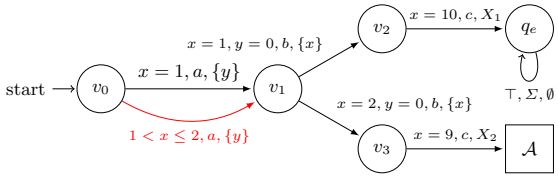


Fig. 3. Reducing emptiness to untimed $\exists -$ *resilience*

automata. However, it is known [2] that PSPACE hardness of the emptiness question still holds for deterministic TAs. Similarly, it is known that universality of the suffix languages is in PSPACE, even for deterministic automata [14]. Hence, considering determinitic timed automata will not improve the complexity of existential resilience.

5 A timed product of automata

In this section, we define a product of automata, that synchronizes its components on time before a fault, ignoring transitions labels, and on events once K steps have occurred after a fault. A first ingredient is to make time elapsing visible.

Definition 10 (Time-elapsing automata). *Let, $\mathcal{A} = (L, L_0, X, \Sigma, T, F)$ be a timed automaton. The time-elapsing version of $\mathcal{A}$ is the automaton $\mathcal{A}^+ = (L, L_0, X, \Sigma \cup \{*\}, T^+, F)$ where $*$ is a particular symbol not appearing in Σ, and $T' = T \cup \{(l, *, true, \emptyset, l) \mid l \in L\}$*

Intuitively, $\mathcal{A}^+$ behaves as $\mathcal{A}$ but has special transitions (the $*$ transitions) that are always feasible from a location, and reset no clocks. Obviously, if w is a word of $\mathcal{A}^+$, then its projection on letters that are not $*$ is a word of $\mathcal{L}(\mathcal{A})$

Definition 11 (Product). *Let $\mathcal{A}^+ = (L_A, L_{A_0}, X_A, \Sigma, T_A, F_A)$ and $\mathcal{B}^+ = (L_B, L_{B_0}, X_B, \Sigma, T_B, F_B)$ be two time elapsing automata, where $\mathcal{B}$ contains faulty transitions. Let $K \in \mathbb{N}$ be an integer. Then, the product $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$ is a tuple $(L_{A\times B}, L_{A_0\times B_0}, X_A \cup X_B, \Sigma, T_{A\times B}, F_{A\times B})$ where $L_{A\times B} \subseteq \{L_A \times L_B \times [-1, K]\} \cup \{\bot\}$, $F_{A\times B} = F_A \times F_B \times \{0\}$, and initial state of the product is $l^0_{A\times B} = (l^0_B, l^0_A, -1)$. 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. We build states and transitions inductively:*

1. *If $(l_B, l_A, -1)$ is a location of $L_{A\times B}$, $t_B = (l_B, b, g_B, R_B, l'_B)$ is a non-faulty transition of $\mathcal{B}^+$ and $t_A = (l_A, a, g_A, R_A, l'_A)$ is a transition of $\mathcal{A}^+$, then $((l_B, l_A, -1), b, g_B \wedge g_A, R_B \cup R_A, (l'_B, l'_A, -1))$ is a transition of $T_{A\times B}$. This synchronization still holds for $a = *$ or $b = *$.*
2. *If $(l_B, l_A, -1)$ is a location of $L_{A\times B}$, $t_B = (l_B, b, g_B, R_B, l'_B)$ is a faulty transition of $\mathcal{B}^+$ and $t_A = (l_A, a, g_A, R_A, l'_A)$ is a transition of $\mathcal{A}^+$, then $((l_B, l_A, -1), b, g_B \wedge g_A, R_B \cup R_A, (l'_B, l'_A, K))$ is a transition of $T_{A\times B}$.*

3. *If* (l_B, l_A, c) *is a location of* $L_{A\times B}$ *with* $c > 0$, $t_B = (l_B, b, g_B, R_B, l'_B)$ *is a transition of* $\mathcal{B}^+$ *with* $b \neq *$ *and* $t_A = (l_A, a, g_A, R_A, l'_A)$ *is a transition of* $\mathcal{A}^+$, *then* $((l_B, l_A, c), b, g_B \wedge g_A, R_B \cup R_A, (l'_B, l'_A, c-1))$ *is a transition of* $T_{A\times B}$.
4. *If* (l_B, l_A, c) *is a location of* $L_{A\times B}$ *with* $c > 0$, $t_B = (l_B, b, g_B, R_B, l'_B)$ *is a transition of* $\mathcal{B}^+$ *with* $b = *$ *and* $t_A = (l_A, a, g_A, R_A, l'_A)$ *is a transition of* $\mathcal{A}^+$ *with* $a \neq *$, *then* $((l_B, l_A, c), b, g_B \wedge g_A, R_B \cup R_A, (l'_B, l'_A, c))$ *is a transition of* $T_{A\times B}$.
5. *If* $(l_B, l_A, 0)$ *is a location of* $L_{A\times B}$, $t_B = (l_B, b, g_B, R_B, l'_B)$ *is a transition of* $\mathcal{B}^+$ *and* $t_A = (l_A, a, g_A, R_A, l'_A)$ *is a transition of* $\mathcal{A}^+$ *with* $b = a$, *then* $((l_B, l_A, 0), b, g_B \wedge g_A, R_B \cup R_A, (l'_B, l'_A, 0))$ *is a transition of* $T_{A\times B}$.
6. *If* $(l_B, l_A, 0)$ *is a location of* $L_{A\times B}$, $t_B = (l_B, b, g_B, R_B, l'_B)$ *is a transition of* $\mathcal{B}^+$ *and for every* $t_A = (l_A, a, g_A, R_A, l'_A)$ *of* $\mathcal{A}^+$, $b \neq a$, *then* $((l_B, l_A, 0), b, g_B, R_B, \bot)$ *is a transition of* $T_{A\times B}$.
7. *If* $(l_B, l_A, 0)$ *is a location of* $L_{A\times B}$, $t_B = (l_B, b, g_B, R_B, l'_B)$ *is a transition of* $\mathcal{B}^+$ *and there exists a transition* $t_A = (l_A, a, g_A, R_A, l'_A)$ *is a transition of* $\mathcal{A}^+$ *with* , $b = a$, *then* $((l_B, l_A, 0), b, g_B \wedge \bigwedge \neg g_A, R_B, \bot)$ *is a transition of* $T_{A\times B}$.

Intuitively, the product advances time in $\mathcal{A}$ and $\mathcal{B}$ jointly until a fault occurs without considering the actions (steps 1-2), and for the next K (steps 3-4), and becomes a synchronous product of $\mathcal{A}$ and $\mathcal{B}$ K steps after a fault (steps 5-6), i.e., matches actions of $\mathcal{A}$ and $\mathcal{B}$.

As $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$ is a timed automaton, we can easily compute its region automaton $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$, whose states are of the form (l_B, l_A, c, r), where c is a value between -1 and K, and r a region. For a state (l_B, l_A, c, r) of this region automaton, we can decompose region r into r_B and r_A, denoting respectively the restriction of r to clocks of $\mathcal{B}^+$ and $\mathcal{A}^+$. Notice that regions of $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$ and regions of $\mathcal{R}(\mathcal{B}^+)$, $\mathcal{R}(\mathcal{A}^+)$ differ, as enlarging $\mathcal{A}^+$ with a fault model changes the constants used. A second remark is that $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$ contains locations of the form $(l_B, l_A, -1, r)$ depicting a pair of locations that are accessible in $\mathcal{B}^+$ and $\mathcal{A}^+$ without a faulty transition. Let us define a region projection operator $\Pi_X : Reg(Y) \to Reg(X)$ that projects a region r_Y on a set of clocks Y to a region r_X of a set of clocks $X \subseteq Y$.

Proposition 2. *Let* (l_B, l_A, c, r) *be a reachable state of* $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$. *Then there exists a duration* δ *and a pair of runs* ρ_1, ρ_2 *of* $\mathcal{B}^+$ *and* $\mathcal{A}^+$, *of common duration* δ *ending in configuration* (l_B, ν_B) *and* (l_A, ν_A) *such that* $\nu_B \in [\Pi_{X_B}(r)]$ *and* $\nu_A \in [\Pi_{X_A}(r)]$.

The region automaton $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$ contains several types of states that will be of particular interest later in the next section. We will say that a state is synchronizing if it is of the form $S = (l_B, l_A, 0, r)$ and it has a predecessor of the form $S' = (l'_B, l'_A, 1, r)$. That is, it is state when K steps after a fault have been completed. We will use these states as starting points to show that a faulty behavior can (or cannot) mimic a suffix of $\mathcal{L}(\mathcal{A})$. Further, we will say that a state $S = (l_B, l_A, n, r)$ of $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$ is accepting if l_B is a final location of $\mathcal{B}$ and l_A a final location of $\mathcal{A}$. Finally, we say that a state $S = (l_B, l_A, n, r)$ is

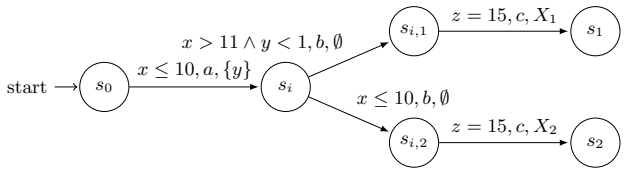


Fig. 4. The gadget automaton $\mathcal{G}_{und}$.

safe if there is an accepting state reachable from S in the region automaton of the product.

If $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$ has only safe synchronizing states, then obviously $\mathcal{A}$ is $K - \exists - resilient$. Notice that the converse is not true: it is not sufficient to show that $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$ has a synchronizing state that is unsafe to prove that $\mathcal{A}$ is not $K - \exists - resilient$, as a word not accepted from $S = (l_B, l_A, 0, r)$ might be accepted from another state $S = (l_B, l'_A, 0, r')$ with $\Pi_{X_B}(r') = \Pi_{X_B}(r')$.

6 Timed Resilience

When resilience considers timed behaviors, instead of untimed languages, we can show that the timed $\forall - resilience$ is undecidable. The undecidability result is achieved by reducing the language inclusion problem of timed automata, that is undecidable [2] to a K-$\forall$-resilience problem for timed automata.

Theorem 3. *K-$\forall$-resilience of timed automata is undecidable.*

Proof (Proof Sketch). The proof proceeds by reduction from a timed language inclusion problem. Taking two timed automata $\mathcal{A}_1 = (L_1, \{l_{0_1}\}, X_1, \Sigma_1, T_1, F_1)$ and $\mathcal{A}_2 = (L_2, \{l_{0_2}\}, X_2, \Sigma_2, T_2, F_2)$, we can define a gadget $\mathcal{G}_{und}$ as shown in Fig 4, and build an automaton $\mathcal{B}$ that behaves as $\mathcal{A}_1$ after 15 time units in no fault occurs, and and $\mathcal{A}_2$ after 15 time units if a fault occurs. This is done by collapsing state s_1 in the gadget with the initial state of $\mathcal{A}_1$ and state s_2 with the initial state of $\mathcal{A}_2$. Then, $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ iff $\mathcal{B}$ is $2 - \forall - resilient$. A complete proof of this theorem can be found in Appendix C

Theorem 4. *K-$\exists$-resilience of timed automata is in EXPSPACE.*

Proof. The proof proceeds in three steps.

1. We first prove that region automata for the product $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$ gives a sound abstraction of prefixes of timed words, i.e., that runs with close durations and identical sequences of transitions end in the same regions.
2. We then build an automaton $\mathcal{R}(2^{\mathcal{A}\times\mathcal{B}})$ whose states are sets of "local states" of the form (l^i_B, l^i_A, c^i, r^i) representing a state of $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$, such that every local state in a state has identical $\mathcal{B}$ component, and is reachable in the same duration as the other local states in that state.
3. Finally, we show that $\mathcal{A}$ is not $K - \exists - resilient$ iff there exists a state S of $\mathcal{R}(2^{\mathcal{A}\times\mathcal{B}})$ such that from none of the local states (l^i_B, l^i_A, c^i, r^i) in S one can reach an accepting state of $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$.

6.1 Reaching the same region

Let us first show an interesting properties of region automaton, saying that sequences of transitions can always be retimed to occur later, without changing acceptance of the considered run.

Lemma 2. *Let $\sigma = \tau_1 \cdots \tau_k$ a sequence of timed transitions on a set of clocks X. Let $w = a_1 \dots a_k$ be a timed word following σ from timed configuration 0^X of duration $d + e$, with $d \in \mathbb{N}$ and $e \in (0, 1)$. Then for all $e' \in (0, 1)$, there exists w' a timed word following σ from timed configuration 0^X and of duration $d + e'$.*

Proof (Proof Sketch). We refer readers to Appendix C for a complete proof. Briefly, we show that we can retime all actions in a timed word while keeping the same ordering on fractional parts. This retiming can be achieved by first changing date t_k to $d + e'$ and then changing all fractional parts that are greater than e in increasing order, with growing values in $(0, 1)$ and fractional parts that are smaller than e in decreasing order, replacing them with decreasing values. This retiming suffices to obtain a timed word of duration $d + e'$ that satisfies all constraints on transitions along the considered word, and hence is still accepted by the automaton.

The region automaton of $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$ has similar properties.

Lemma 3. *Let $\sigma = \tau_1 \cdots \tau_k$ be a sequence of timed transitions on set of clocks X, and $\sigma_B = \tau_1^B \cdots \tau_\ell^B$ be a set of timed transition on set of clocks Y. Let w, w_B be timed words following σ, σ_B from timed configuration $0, \cdots, 0$ of duration $d + e$, with $d \in \mathbb{N}$ and $e \in (0, 1)$. These two timed words are not synchronized on events, they only have the same duration.*

Assume that $e \in (0, 1)$ (the other case is simpler). Let w'_B be a timed word following σ_B from timed configuration $0, \cdots, 0$ of duration $d+e'$, with $e' \in (0, 1)$, passing through the exact same sequence of regions as w_B. Then there exists w' following σ of duration $d + e'$ such that (w'_B, w') reaches the same region of $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$ as (w_B, w).

Proof. For every $i \in \{1, \cdots, k\}$ we want to modify the fractional part $frac(t_i)$ of dates t_i in word w to get a new word w' such that (w'_B, w') reaches the same region of $A \times B$ as (w, w_B).

Let t_x, t_y be the dates at which the clocks $x \in X$ and $y \in Y$ have been last reset in w, and in w_B respectively. Similarly, let t'_y be the dates at which the clocks $y \in Y$ have been last reset in w'_B. The date t_x corresponds to some date t_i where a clock $x \in X$ have been reset in a transition τ_i $i \in \{1, \cdots, k\}$ in w. We use e to represent the fractional part of the date at which in w and w_B ends. Let e_x be the fractional part of the date at which clock x is reset for the last time in w. We order $\{e\} \cup \{frac(t_x) \mid x \in X\} \cup \{frac(t_y) \mid y \in Y\}$, the fractional values in (0,1). If we keep the same relations in w'_B, w' , then we will keep the same region of $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$ as for (w_B, w). For each $x \in X$ (resp $frac(t_i)$) we will change e_x (resp $frac(t_i)$) , to design w' and w'_B : Let e_k to represent the

fractional part of the date when the last transition τ_k and τ_l^B fired in w' and w'_B.

We set $e_k = e'$ the fractional part of the date of the last transaction in w'_B and in w', and set the other t_i accordingly (and in particular the date t_x):

1. all $frac(t_i), frac(t_x)$ that are equal to $frac(t_k) = e_k$ are set to e',
2. Choose $i \in \{1, \cdots, k\}$ such that, $frac(t_i) > frac(t_k)$ and there is no $j \in \{1, \cdots, k\}$ with $frac(t_k) < frac(t_j) < frac(t_i)$.
   - If there exists a $y \in Y$ such that $frac(t_i) = frac(t_y)$, then set $e'_i = frac(t'_y)$, i.e., if the next date in the ordering is a reset of a clock $y \in Y$, then we will copy the fractional part from the word w'_B
   - Otherwise, if the next date in the ordering is a reset of a clock $x \in X$ then we need to choose a value such that the fractional value maintains the order. We will choose a $e'_i \in (e_k, 1)$ such that, $e'_i > frac(t'_y)$ for all $y \in \{y' \in Y \mid frac(t_i) > frac(t_{y'})\}$.
   - all $frac(t_j) = frac(t_i)$ are set to e'_i.
3. Repeat step (2) replacing $frac(t_k)$ by $frac(t_i)$ till there is no j with $frac(t_j) > frac(t_i)$.
4. Do the same iteration of (2) and (3) for $frac(t_i) < frac(t_k)$, setting fractional parts to decreasing values in $(0, e_k)$.

Note from Lemma 2 it is clear that this construction respects all the constraints of σ. We need to prove that this also respects the constraints of the regions of $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$ which is done by showing this new construct keeps the same ordering of the fractions of $(w', w'_B) \in \{e\} \cup \{frac(t_x) \mid x \in X\} \cup \{frac(t_y) \mid y \in Y\}$ as in (w, w_b). It is clear that, the ordering of $\{frac(t_x) \mid \forall x \in X\}$ is preserved because of Lemma 2 and the ordering of $\{frac(t_y) \mid \forall y \in Y\}$ is preserved as both of w_B and w'_B goes through exact same sequence of regions. But, we also have to assure that the ordering among every $frac(t_x)$ and $frac(t_y)$ $\forall x \in X, \forall y \in Y$ is also maintained. Every time we choose a new assignment for a $frac(t_x)$ we make sure that, the chosen value respects the previous ordering of $frac(t_x)$ along with the ordering $frac(t_x) > frac(t_y)$. Every time we choose a new fraction for a $frac(t_x)$ we make sure that it maintains the orderings and hence in the end it maintains the ordering of $(w, w_B) \in \{e\} \cup \{frac(t_x) \mid x \in X\} \cup \{frac(t_y) \mid y \in Y\}$. Now, we did not change the integer values of the clocks and also maintained the ordering of the fractional parts, hence, this construction preserves the constraints of regions of $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$ and both (w_B, w) and (w'_B, w') end in the same region of $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$.

Lemma 3 shows that in the region of the product automaton $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$, if the synchronization of a normal sequence σ_A and faulty sequence of transitions σ_B reaches a region r within a duration $d+e$, then the same pair of sequences of transitions can be retimed to reach any accessible neigbour in the region. This has several consequences: First if σ_A, σ_B end respectively in locations l_A, l_B then any retiming will end in the same locations. As shown by the lemma, these retimings also end in the same location. Hence using the properties of regions, one can ensure that for any move from r to r' (symbolizing firing of a pair of

transitions in $\mathcal{A}^+$ and $\mathcal{B}^+$), there exists a proper duration that guarantees that this move is feasible from any point in the region.

6.2 A subset-region construction

If a synchronizing state of the product region automaton is of the form $(l_A, l_B, c = 0, r)$, and is a safe state, then any faulty word w_B ending in that region can be extended to mimic a suffix of a non-faulty word. If not, then there might be another word of $\mathcal{A}$ of the same duration as w_B allowing to end in a safe state of $\mathcal{R}(\mathcal{A}^+ \times \mathcal{B}^+)$. We hence build a subset-region automaton, whose states are the result of a subset construction on possible regions and locations of $\mathcal{A}^+ \bigotimes_K \mathcal{B}^+$ that are accessible within an identical duration.

The subset-region automaton for $\mathcal{A}^+ \times \mathcal{B}^+$, denoted $\mathcal{R}(2^{\mathcal{A}\times\mathcal{B}})$, will maintain a set of local states composed of a location of $\mathcal{B}^+$, a number of steps since a fault occurred, a location of $\mathcal{A}^+$, and a region over a set of clocks $X_A \uplus X_B$ where X_A and X_B are disjoint copies of clocks of $\mathcal{A}$. Each state is hence of the form $S = \big((l_B, l_A^1, c, r_1), \dots (l_B, l_A^k, c, r_k)\big)$ with $l_B \in L_B$, $l_A^i \in L_A$.

We start from the configuration $S_0 = \big((l_0^B, l_0^A, -1, r)\big)$ For each state $S = \big((l_B, l_A^1, c, r_1), \dots (l_B, l_A^k, c, r_k)\big)$, we can compute successors. As we will define subset of states of the form (l_i, r_i), where l_i is a location and r_i a region, we will consider that each region defines constraints on its own copy of clocks of $\mathcal{A}$. We can hence define products of regions of the form $r_i \times r_j$ as the conjunction of their constraints on their respective clocks (this product generalizes to an arbitrary number of regions). We say $S' = \Big((l'_B, {l'}_A^1, c', {r'}_1), \dots ({l'}_B, {l'}_A^q, c', {r'}_q)\Big)$ is a successor of S iff

- there exists a transition $t = (l_B, g, a, R, {l'}_B)$ of $\mathcal{B}$, $c' = K$ if $c = -1$ and t is a faulty transition, $c' = c - 1$ if $c > 0$
- for every local state (l_B, l_A^i, c, r_i) there exists a maximal set of transitions $t_{i,1} \dots t_{i,m_i}$ where every $t_{i,j}$ is of the form $t_{i,j} = (l_A^i, g_{i,j}, a_{i,j}, R_{i,j}, {l'}_A^{i,j})$ and a region ${r''}_{i,j}$ such that ${r''}_{i,j} \models g \wedge g_{i,j}$

  $$ {r''}_{1,1} \times {r''}_{1,2} \cdots \times {r''}_{k,1} \times \cdots \times {r''}_{k,m_k} $$

  is a time successor of $(r_1)^{m_1} \times (r_2)^{m_2} \cdots \times (r_k)^{m_k}$,

  $$ S' = \{({l'}_B, {l'}_A^{i,j}, c, {r'}_{i,j}) \mid r_{i,j} \text{ is obtained from } {r''}_{i,j} \text{ by resetting } R_{i,j}\} $$

In each local state (l_B, l_A^i, c, r_i), the region reached by $\mathcal{B}$ (resp. $\mathcal{A}$) is the projection of r_i on the copy of clocks owned by $\mathcal{B}$ (resp. $\mathcal{A}$). This local state is a region of the product $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$, so, using proposition 2, we know that there exists a pair of runs in $\mathcal{A}$ and $\mathcal{B}$ of common duration reaching this region. Further, as we take a common time successor for all r_i's this duration is common to all local states in each S built this way. Last, following lemma 3, each local state is a sound abstraction of configurations reached by pairs of runs of identical duration ending in that region. The construction stops when the value of c is 0, i.e. it ends

with a location l_B and a region r_B for $\mathcal{B}$, and a set of possible location and regions for $\mathcal{A}$ that can be accessed with the same duration K steps after a fault. One can immediately notice that by choosing time successors from products of regions, this guarantees that at every move, there exists a common duration d allowing to move from local states of S to local states of S'. Choosing a different value for d results in different time successor regions, and hence possibly in different states. One can remark that for every local state (l_B, l_A, b, r) of S, there exists necessarily one transition of $\mathcal{A}^+$ that can be used, as its guard is true. Notice also that this construction cannot be achieved as a determinization or subset construction from $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$, as subset construction would assemble local states that are accessible after distinct durations, while our construction finds time successors for products of regions.

6.3 A check for non-resilience

Lemma 4. *$\mathcal{A}$ is not $K-\exists-$resilient iff there exists a faulty state of the form $S = \big((l_B, l_A^1, K, r_1), \dots (l_B, l_A^k, K, r_k)\big)$ in $\mathcal{R}(2^{\mathcal{A}\times\mathcal{B}})$ from which all synchronizing states*
$S' = \big((l_B, l_A^1, 0, r_1), \dots (l_B, l_A^k, 0, r_k)\big)$ reachable in K steps from S contain only local states that are not safe.

Proof. Assume that a state $S_n = \big((l_B, l_A^1, K, r_1), \dots (l_B, l_A^k, K, r_k)\big)$ is reachable in $\mathcal{R}(2^{\mathcal{A}\times\mathcal{B}})$ (a fault has occurred), and that all states $S_{n+K} =$
$\big((l_B, l_A^1, 0, r_1), \dots (l_B, l_A^k, 0, r_k)\big)$ reachable in K steps after S_n contain only unsafe local states. Then there exists a sequence of moves from S_0 to S_{n+K}, where each move is a synchronization of transitions from the $\mathcal{A}$ component of local states with a transition of $\mathcal{B}$ at a date d_n. From the construction of the automaton $\mathcal{R}(2^{\mathcal{A}\times\mathcal{B}})$, the sequence $\rho_B = (t_B^1, d_1) \dots (t_B^n, d_n)$ of transitions of $\mathcal{B}$ is a run of $\mathcal{B}$, and every sequence $\rho_A = (t_A^1, d_1) \dots (t_A^n, d_n)$ connecting local states $S_0, S_1, \dots S_n$ has the same duration as ρ_B. Each move $S_i \longrightarrow S_{i+1}$ in the region automaton gives the whole set of successors of local states in S_i for which there exists a common duration. Hence, S_n contains all regions that are accessible by synchronizing ρ_B with a run that has the same duration. As none of the local states reached K steps later is safe $(l_B, l_A^i, 0, r_i)$, there is no way to extend ρ_B is such a way that it synchronizes with a sequence of transitions of $\mathcal{A}$ K steps later, and hence there exists at least one run with a fault such that, K steps after the fault, one cannot recognize a suffix of a word of $\mathcal{A}$, so $\mathcal{A}$ is not $K-\exists-$ resilient.

Now assume that $\mathcal{A}$ is not $K-\exists-$ resilient. Then, there exists a faulty run ρ_B of duration d such that K steps after a fault, no extension of ρ_B matches a suffix of a word in $\mathcal{L}(\mathcal{A})$. Hence, for every run ρ_A of duration d, the synchronization of $\mathcal{A}^+$ and $\mathcal{B}^+$ allows to reach a configuration $C = (l_A, l_B, c = 0, \nu)$ where ν is a valuation over $X_A \uplus X_B$ which belongs to some region r, but any sequence of synchronized transitions of $\mathcal{A}$ and $\mathcal{B}$ with common letters and with any choosen dates leads to a configuration that is not accepting in $\mathcal{A}$ or in $\mathcal{B}$. Hence, as ν is a valuation of r, $(l_A, l_B, c = 0, r)$ is not a safe state of $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$. This is true

for any ρ'_A that has the same duration as ρ_B leading to a region r'. Now, by contradiction, assume that the path of $\mathcal{R}(2^{\mathcal{A}\times\mathcal{B}})$ corresponding to ρ_B reaches a configuration containing a location and region of $\mathcal{B}^+ \bigotimes_K \mathcal{A}^+$ from which a safe state can be reach K steps after the fault. This means there exist two paths (ρ'_b, ρ'_a) of same duration d', with ρ'_b passing through the exact same regions in ρ_b (hence d and d' belong to the same integral region), which can reach a safe state after K steps. We can thus invoke Lemma 3 to obtain a path ρ''_a that have duration d and such that ρ''_a, ρ_b reaches the same region as ρ'_a, ρ'_b. In particular, there exists a continuation of ρ''_a, ρ_b that is BTN after K steps, a contradiction.

Let us now consider the size of $\mathcal{R}(2^{\mathcal{A}\times\mathcal{B}})$. It builds subsets of states of $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$, so it is of doubly exponential size. Deciding whether a particular state S of $\mathcal{R}(2^{\mathcal{A}\times\mathcal{B}})$ is reachable is decidable in EXPSPACE, deciding if S' is a successor of S K steps later is also in NEXPSPACE=EXPSPACE, and deciding if S' contains only unsafe local states requires up to an exponential number of PSPACE tests using $\mathcal{R}(\mathcal{B}^+ \bigotimes_K \mathcal{A}^+)$. Hence, the overall decision procedure for $K-\exists-$resilience is in EXPSPACE.

Theorem 5. *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 automaton $\mathcal{G}_{und}$ of Figure 4. We take any automaton $\mathcal{A}$ and collapse its inital 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, $t_e = (s_2, \sigma, true, \emptyset, s_2)$ 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 $\mathcal{A}$ has no accepting word, there is no timed word after a fault which is a suffix of a word in $\mathcal{L}(\mathcal{A})$, and conversely, if $\mathcal{L}(\mathcal{A}) \neq \emptyset$, 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.

7 Integer Reset Automata

Integer reset automata [15] is a syntactic subclass of timed automata where resets happen only when clocks have integral values. This is guaranteed syntactically by requiring that the guard of every transition that resets a clock contains a constraint that compares a clock value to an integer.

Definition 12 (IRTA). *An integer reset timed automaton (IRTA) $\mathcal{A} = (L, L_0, X, \Sigma, T, F)$ is a timed automaton in which every transition $t = (l, a, \phi, R, l') \in T$ is such that $R \neq \emptyset$ only if ϕ contains at least one atomic constraint of the form $x = c$ where $c \in \mathbb{N}$.*

If furthermore the automaton contains unobservable transitions (transitions labeled by ϵ), then the automaton is called an $\epsilon-$IRTA. From the definition it is

clear that the clocks can only be reset at integer dates. The class of IRTA is closed under union, intersection and complement. For a given IRTA $\mathcal{A}$, one can effectively compute a deterministic IRTA $\mathcal{B}$ that recognizes the same language [15]. However, the size of $\mathcal{B}$ is doubly exponential. Last, for a timed automaton $\mathcal{A}$ and an $\epsilon-$IRTA B the problem $\mathcal{L}(A) \subseteq \mathcal{L}(B)$ is decidable in EXPSPACE [15]. Let us recall some elements used to prove decidability of this inclusion.

For a given IRTA $\mathcal{A}$ we can define a map $f : \rho \to w_{unt}$ that maps every run ρ of $\mathcal{A}$ 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 : $dt(x) = (\delta.\checkmark)^k$ 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 $w = (a_1, d_1) \dots (a_n, d_n)$ the word $f(w) = w_1.a_1.w_2.a_2 \dots w_n.a_n$ 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 $\checkmark$ and δ but keeps actions unchanged. Consider for example, a word $w = (a, 1.6).(b, 2.7).(c, 3.4)$ the map will be $f(w) = \delta\checkmark\delta a\checkmark\delta b\checkmark\delta c$. It is shown in [15] for two timed words ρ_1, ρ_2 with $f(\rho_1) = f(\rho_2)$ then $\rho_1 \in \mathcal{L}(\mathcal{A})$ if and only if $\rho_2 \in \mathcal{L}(\mathcal{A})$. It is also shown that we can construct a Marked Timed Automata(MA) from $\mathcal{A}$ with one extra clock and polynomial increase in the number of locations such that $untime(\mathcal{L}(MA)) = f(\mathcal{L}(\mathcal{A}))$. The MA of $\mathcal{A}$ duplicates transitions of $\mathcal{A}$ to differentiates firing at integral/non integral dates, plus transitions that makes time elapsing visible using the additional clock which is reset at each global integral time stamp.

Definition 13. *Given a timed automata $\mathcal{A} = (L, L_0, X, \Sigma, T, F)$ the Marked Timed Automata of $\mathcal{A}$ is a tuple$MA = (L', L'_0, X \cup \{n\}, \Sigma \cup \{\checkmark, \delta\}, T', F')$ such that (i) $n \in X$ (ii) $L' = L^0 \cup L^+$ where for $\alpha \in \{0, +\}, L^\alpha = \{l^\alpha \mid l \in L\}$(iii) $L'_0 = \{l^0 \mid l \in L_0\}$, (iv) $F' = \{l^0, l^+ \mid l \in F\}$ and (v) E' is defined as follows,*

$$\begin{aligned} E' = &\{(l^0, a, \phi \wedge n = 0?, R, l'^0) \mid (l, a, \phi, R, l') \in E\} \\ &\cup \{(l^+, a, \phi \wedge 0 < n < 1?, R, l'^+) \mid (l, a, \phi, R, l') \in E\} \\ &\cup \bigcup_{l \in L} (l^0, \delta, 0 < n < 1, \emptyset, l^+) \cup \bigcup_{l \in L} (l^+, \checkmark, n = 1?, \{n\}, l^0) \end{aligned}$$

Theorem 6 ([15]Thm.5). *Let $\mathcal{A}$ be a timed automaton and MA be its marked automaton. Then $Unt(\mathcal{L}(MA)) = f(\mathcal{L}(\mathcal{A}))$*

Remark 3. The marked automaton of an IRTA is also an IRTA.

Theorem 7. *Untimed K-$\forall$-resilience for IRTA is EXPSPACE Complete.*

Proof. EXPSPACE membership follows easily from the space complexity of untimed $\forall-resilience$ in the general case, as proved in Theorem 1. For the hardness part, we reduce the language inclusion problem of IRTA which is known to be EXPSPACE-Complete[5] to a untimed $K-\forall-resilience$ problem of IRTA. The crux of the proof lies in the fact that we can reduce the timed language inclusion problem of IRTA to untimed language inclusion problem of the corresponding marked automata(c.f. Lemma 6). Also, we know that the marked automaton

of an IRTA is also an IRTA. Now, if we show that we can reduce the untimed language inclusion of marked automata to untimed $K-\forall-resilience$ of IRTA then we are done. To show the latter we will use a gadget similar to Figure 5.

Theorem 8. *Timed K-$\forall$-resilience for IRTA is EXPSPACE-Complete.*

Proof. We first prove that the timed K-$\forall$-resilience is in EXPSPACE. From [15] that inclusion of $L(\mathcal{B})$ in $L(\mathcal{A})$ is in EXPSPACE for IRTA, and even for ϵ-IRTA. Let us first prove that the timed suffix language of an IRTA $\mathcal{A}$ can be recognized by an ϵ-IRTA $\mathcal{A}^S$.

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

We can now prove that the suffixes of timed words that contain a fault are also recognized by an $\epsilon-$IRTA and start immediately K steps after a fault are recognized by an $\epsilon-$IRTA. Given a timed automaton $\mathcal{A}$ and a fault model $\mathcal{P}$, we build an automaton $\mathcal{B}^S$ which states remember if a fault has occurred, and how many transitions have been taken since a fault occurred (see the construction in Theorem 1). Then, we re-label every transition occurring before a fault by ϵ, keeping the same locations, guards and resets. We relabel every transition that occurs less than K steps after a fault by ϵ similarly, and leave transitions occurring more than K steps after a fault unchanged. Accepting locations are locations occurring after a fault and that are accepting in the original automaton. Then, every accepted faulty run word of $\mathcal{B}^S$ is of the form $\rho = (t_1, d_1) \dots (t_f, d_f).(t_{f+1}, d_{f+1}) \dots (t_{f+K}, d_{f+K}). \dots . (t_n, d_n)$ where $t_1, \dots t_{f+K}$ are ϵ transitions. A run ρ is BTN if and only if $(a_{f+K+1}, d_{f+K+1}) \dots (a_n, d_n)$ is a suffix of a timed word of ta, i.e., is recognized by $\mathcal{A}^S$.

One can then apply the results of [15] to check that every word in $\mathcal{B}$ (reading only ϵ before that fault) is recognized by the suffix automaton $\mathcal{A}^S$. The size of automaton $\mathcal{A}^S$ is twice the size of $\mathcal{A}$, and the size of $\mathcal{B}^S$ is in $O(K.|\mathcal{A}|)$. Checking $K-\forall-resilience$ is hence in EXPSPACE.

We can prove the hardness part, by a reduction from the inclusion problem of IRTA, known to be EXPSPACE-complete [5]. The idea of the proof follows the same lines as for the untimed $K-\forall-resilience$ of timed automata. But, IRTAs are not closed under fault enlargement, we construct a gadget such that we can construct an equivalent IRTA of the faulty automata. More details of this proof can be found in the Theorem 13 in Appendix E.

Theorem 9. *The $K-\exists$-resilience of IRTAs is PSPACE-Complete*

Proof (sketch). The proof builds on the properties of the untimed languages recognized by marked automata. We first show that existential resilience in IRTA can be brought back to an untimed property called $K - prefix - resilience$. This property compares words of $\mathcal{A}$ and words of $\mathcal{A}_{\mathcal{P}}$ the following way. The projection of the image of prefix of a faulty word in $\mathcal{A}_{\mathcal{P}}$ up to K steps after a fault to the set $\{\delta, \checkmark\}$ via map $f()$ followed by the projection operator$\downarrow_{\{\delta,\checkmark\}}$ should be equal to the projection of the image of a prefixes of a word in $\mathcal{A}$ via map $f()$ followed by the projection operator $\downarrow_{\{\delta,\checkmark\}}$, and then the untimed marked languages of remaining suffixes should be identical. Existential timed resilience in IRTA holds if and only if $K - prefix - resilience$ holds. This property is a property of the region automaton of the marked automaton MA, and can be checked in PSPACE. PSPACE-Hardness is already proved in Theorem 5. For a complete proof, we refer readers to appendix E.

Remark 4. Considering untimed existential resilience for IRTAs does not change the complexity of the problem. Indeed, one can use the gadget of Fig. 5 to encode a language emptiness problem as a resilience problem on IRTAs. As language emptiness for IRTAs is PSPACE-Complete, and as the problem is shown in PSPACE by Thm. 2, untimed K-$\exists$-resilience of IRTAs is PSPACE-complete.

8 Conclusion

Resilience is a new technique to verify how robust a timed system is to unspecified delays. A universally resilient timed system can recover from any delay in some fixed number of steps. Checking if a timed system is universally resilient is undecidable, except for the restricted subclass of IRTA, where it is EXPSPACE-Complete. A slightly relaxed notion of resilience is the existential one, where we guarantee the existence of a controller that can bring back the system to a normal behaviour after an unexpected delay. Existential resilience is decidable and in EXPSPACE in general. As we only have a PSPACE lower bound (from timed language emptiness), the question of whether this is tight is open. For IRTAs, however, we are able to show a tight bound, yielding PSPACE-Completeness of existential timed resilience. On the other hand, untimed resilience is decidable, both for the universal and existential cases. Somewhat surprisingly, we do not get an improvement here by moving to IRTA, where the complexity remains the same.

As futurework, it would be interesting to find the precise boundaries of decidability for universal timed resilience. The undecidability proof for universal timed resilience does not apply to deterministic timed automata or event recording automata (ERA) [3]. ERA are an interesting class orthogonal to IRTA in expressiveness, and whose language inclusion is in PSPACE [3, 4] and it would be interesting to investigate resilience for ERA and other determinizable classes. Two natural extensions of this work are to consider resilience control, that is, the design of timed controllers that allow a system to be resilient within the smallest possible number of steps, and that of continuous resilience, where the system recovers in some fixed duration rather than within some number of steps.

References

1. S. Akshay, B. Bollig, P. Gastin, M. Mukund, and K. Narayan Kumar. Distributed timed automata with independently evolving clocks. Fundam. Informaticae, 130(4):377–407, 2014.
2. R. Alur and D.L. Dill. The theory of timed automata. In Real-Time: Theory in Practice, REX Workshop, volume 600 of LNCS, pages 45–73, 1991.
3. R. Alur, L. Fix, and T.A. Henzinger. Event-clock automata: A determinizable class of timed automata. Theor. Comput. Sci., 211(1-2):253–273, 1999.
4. R. Alur and P. Madhusudan. Decision problems for timed automata: A survey. In Formal Methods for the Design of Real-Time Systems SFM-RT 2004, volume 3185 of LNCS, pages 1–24, 2004.
5. C. Baier, N. Bertrand, P. Bouyer, and T. Brihaye. When are timed automata determinizable? In Proc. of ICALP'09, volume 5556 of LNCS, pages 43–54, 2009.
6. P. Bouyer, F. Chevalier, and D. D'Souza. Fault diagnosis using timed automata. In Proc. of FOSSACS 2005, pages 219–233, 2005.
7. P. Bouyer, N. Markey, and O. Sankur. Robust model-checking of timed automata via pumping in channel machines. In Proc. of FORMATS 2011, volume 6919 of LNCS, pages 97–112, 2011.
8. P. Bouyer, N. Markey, and O. Sankur. Robustness in timed automata. In International Workshop on Reachability Problems, volume 8169 of LNCS, pages 1–18, 2013.
9. R. Brenguier, S. Göller, and O. Sankur. A comparison of succinctly represented finite-state systems. In Proc. of CONCUR 2012, volume 7454 of LNCS, pages 147–161. Springer, 2012.
10. M. De Wulf, L. Doyen, N. Markey, and J-F Raskin. Robust safety of timed automata. Formal Methods Syst. Des., 33(1-3):45–84, 2008.
11. Catalin Dima. Dynamical properties of timed automata revisited. In Formal Modeling and Analysis of Timed Systems, 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007, Proceedings, volume 4763 of Lecture Notes in Computer Science, pages 130–146. Springer, 2007.
12. V. Gupta, T.A. Henzinger, and R. Jagadeesan. Robust timed automata. In Proc. Of HART'97, Hybrid and Real-Time Systems, volume 1201 of LNCS, pages 331–345, 1997.
13. A. Puri. Dynamical properties of timed automata. In DEDS, 10(1-2):87–113, 2000.
14. N. Rampersad, J. Shallit, and Z. Xu. The computational complexity of universality problems for prefixes, suffixes, factors, and subwords of regular languages. Fundam. Informaticae, 116(1-4):223–236, 2012.
15. P. V. Suman, P.K. Pandya, S.N. Krishna, and L. Manasa. Timed automata with integer resets: Language inclusion and expressiveness. In Proc. of FORMATS'08, volume 5215 of LNCS, pages 78–92, 2008.
16. M. Swaminathan, M. Fränzle, and J-P. Katoen. The surprising robustness of (closed) timed automata against clock-drift. In Fifth IFIP International Conference On Theoretical Computer Science - TCS 2008, IFIP 20th World Computer Congress, TC 1, Foundations of Computer Science, September 7-10, 2008, Milano, Italy, volume 273 of IFIP, pages 537–553. Springer, 2008.

A Ordering of Regions

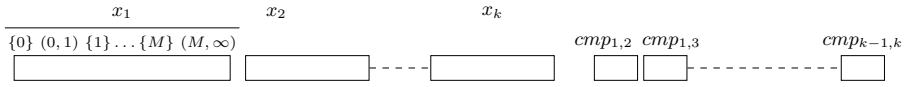
Without loss of generality we can assume that clocks in a set $X = x_1, \ldots x_k$ are indexed by $\mathbb{N}$, and ordered according to this index. Similarly, we can assume that

all guards have integer constants (if this is not the case, one can get back to an integer setting by a multiplication of all values by the least common multiple of all constants). Then for a clock x_i we can easily define an order $\prec_R^i$ on integer intervals and integer points depicting possible values for x as follows:

$\{0\} \prec_R^i]0,1[\prec_R^i \{1\} \prec_R^i \cdots \prec_R^i]M,\infty)$ where, M is the maximum constant of the underlying timed automata. For a constant M, a clock can be in any of these $2.M+2$ intervals or points

Now regions also compare fractional parts of clocks. Given two clocks x_i, x_j, and letting $frac(x_i) = x_i - \lfloor x_i \rfloor$ denote the fractional part of x_i, we define the following integer $cmp_{i,j}$, set to -1 if $frac(x_i) < frac(x_j)$, to 0 if $frac(x_i) = frac(x_j)$, and to 1 if $frac(x_i) > frac(x_j)$.

To define a region, it is hence sufficient to memorize, for each clock x_i, in which interval or point the clock lays (this can be done with a vector of size $2 \cdot M + 2$), and for each pair of clocks x_i, x_j an integer in $\{-1, 0, 1\}$ (this can be done with a vector of size $|X^2|$). We can hence represent a region r as a vector V_r of size $(2 \cdot M + 2) \cdot |X| + |X|^2$. Then, we can define an ordering $\prec_R$ on regions as the componentwise comparison of their respective vectors. Accordingly, one can also compute in PTIME the next region according to that order.



Thus given a timed automata $\mathcal{A}$ we can enumerate the state space of the region automata assuming the locations and clocks are ordered. We can index the states of the region automata as per the enumeration. We use the notation $Enum(\mathcal{A})$ to represent the enumeration of the state space.

B Lemmas for section 4

LEMMA 1 Let ρ be a run that leads to an accepting state of $\mathcal{B}$. Then, the word recognized by ρ is not BTN.

Proof. Consider run ρ is of the form

$\rho = (l_0, r_0, K, \emptyset) \xrightarrow{t_0} (l_1, r_1, K, \emptyset) \ldots (l_i, r_i, K, \emptyset)$
$\xrightarrow{t_i} (\overset{\bullet}{l}_i, r_i, K, \emptyset) \ldots (\overset{\bullet}{l}_{i+K}, r_{i+K}, 0, L \times Reg(X)) \ldots (\overset{\bullet}{l}_n, r_n, 0, X_n)$

Let a_i denote the letter read by transition t_i. Run ρ recognizes a word $w = a_0 \ldots a_{n-1}$, and this word is faulty, i.e., it belongs to $\mathcal{L}(\mathcal{A}_\mathcal{P})$. The suffix of w that appears K steps after the fault in w is $w' = a_{i+K} \ldots a_n$. If w' is a suffix of the untiming of a word in $\mathcal{L}(\mathcal{A})$ then there must be a sequence of moves in $\mathcal{R}(\mathcal{A})$ recognizing a word of the form $w_1.w'$, as $\mathcal{R}(\mathcal{A})$ recognizes the untimed language of $\mathcal{A}$. Hence, one can reach a particular state (l, r) of $\mathcal{R}(\mathcal{A})$, and recognize w' in $|w'|$ steps. However, at step $i + K$ of the run, we start recognizing w' from all states of $\mathcal{R}(\mathcal{A})$, and in particular (l, r), and at the end of the run X_n contains

no accepting state of $\mathcal{R}(\mathcal{A})$. This leads to a contradiction so w' is not a suffix of $Unt(\mathcal{L}(\mathcal{A}))$, and w is not BTN.

C Proofs for section 6

THEOREM 3 The $K-\forall-resilience$ problem for timed automata is undecidable.

Proof. Let $\mathcal{A}_1 = (L_1, L_{0_1}, X_1, \Sigma_1, T_1, F_1)$ and
$\mathcal{A}_2 = (L_2, L_{0_2}, X_2, \Sigma_2, T_2, F_2)$ be two timed automata. We first define a gadget $\mathcal{G}_{und}$ that allows to reach a state s_1 at an arbitrary date $d_1 = 15$ when a fault happens, and a state s_2 at date $d_2 = d_1 = 15$ when no fault occur. This gadget is shown in Fig 4. $\mathcal{G}_{und}$ has 6 locations $s_0, s_i, s_{i,1}, s_1, s_2 \notin L_1 \cup L_2$, three new clocks $x, y, z \notin X_1 \cup X_2$, three new actions $a, b, c \notin \Sigma_1 \cup \Sigma_2$, and 5 transitions $t_0, t_1, t_2, t_3, t_4 \notin T_1 \cup T_2$ defined as: $t_0 = (s_0, a, g_0, \{y\}, s_i)$ with $g_0 ::= x \leq 10$, $t_1 = (s_i, b, g_1, \emptyset, s_{i,1})$ with $g_1 ::= x > 11 \wedge y < 1$, $t_2 = (s_i, b, g_2, \emptyset, s_{i,2})$ with $g_2 ::= x \leq 10$, $t_3 = (s_{i,1}, c, g_3, X_1, s_1)$ with $g_3 ::= z = 15$, and $t_4 = (s_{i,2}, c, g_4, X_2, s_2)$ with $g_4 ::= z = 15$. Clearly, in this gadget, transition t_1 can never fire, as a configuration with $x > 11$ and $y < 1$ is not accessible.

We build a timed automaton $\mathcal{B}$ that contains all transitions of $\mathcal{A}_1$ and $\mathcal{A}_2$, but preceded by $\mathcal{G}_{und}$ by collapsing the initial location of $\mathcal{A}_1$ with s_1 and the initial Location of and $\mathcal{A}_2$ with s_2. We also use a fault model $\mathcal{P} : a \to [0, 2]$, that can delay transitions t_0 with action a by up to 2 time units. The language $\mathcal{L}(\mathcal{B})$ is the set of words:

$$\begin{aligned}\mathcal{L}(\mathcal{B}) = \{&a, d_1).(b, d_2).(c, 15).(\sigma_1, d_3) \dots (\sigma_n, d_{n+2}) \mid (d_1 \leq 10)\\ &\wedge (d_2 \leq 10) \wedge (d_2 - d_1 < 1)\\ &\wedge \exists w = (\sigma_1, d'_3) \dots (\sigma_n, d'_{n+2}) \in \mathcal{L}(\mathcal{A}_2), \forall i \in 3..n+2, d'_i = d_i - 15\}\end{aligned}$$

The enlargement of $\mathcal{B}$ is denoted by $\mathcal{B}_{\mathcal{P}}$. The words in $\mathcal{L}(\mathcal{B}_{\mathcal{P}})$ is the set of words in $\mathcal{L}(\mathcal{B})$ (when there is no fault) plus the set of words in:

$$\begin{aligned}\mathcal{L}^F(\mathcal{B}_{\mathcal{P}})\{&a, d_1).(b, d_2).(c, 15).(\sigma_1, d_3) \dots (\sigma_n, d_{n+2}) \mid (10 < d_1 \leq 12)\\ &\wedge d_2 > 11 \wedge (d_2 - d_1 < 1)\\ &\wedge \exists w = (\sigma_1, d'_3) \dots (\sigma_n, d'_{n+2}) \in \mathcal{L}(\mathcal{A}_1), \forall i \in 3..n+2, d'_i = d_i - 15\}\end{aligned}$$

Now, $\mathcal{B}$ is $K - \forall-$resilient for $K = 2$ iff every word $w = (a, d_1).(b, d_2).(c, 15).(\sigma_1, d_3) \dots (\sigma_n, d_{n+2})$ in $\mathcal{L}^F(\mathcal{B}_{\mathcal{P}})$ is BTN after 2 steps, $K = 2$, i.e. if there exists a word $w = (a, d'_1).(b, d'_2).(c, 15).(\sigma_1, d_3) \dots (\sigma_n, d_{n+2})$ in $\mathcal{L}(\mathcal{B})$. This means that every word of $\mathcal{A}_1$ is a word of $\mathcal{A}_2$. So $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ iff $\mathcal{B}$ is $2 - \forall-$resilient.

LEMMA 2 Let $\sigma = \tau_1 \cdots \tau_k$ a sequence of timed transitions on the same set of clocks X. Let $w = a_1 \dots a_k$ be a timed word following σ from timed configuration 0^X of duration $d + e$, with $d \in \mathbb{N}$ and $e \in (0, 1)$. Then for all $e' \in (0, 1)$, there exists w' a timed word following σ from timed configuration 0^X and of duration $d + e'$.

Proof. Let $t_1 \cdots t_k$ be the dates at which $\tau_1 \cdots \tau_k$ are executed in w, with initial time $t_0 = 0$, and similarly $t'_1 \cdots t'_k$ be the dates at which $\tau_1 \cdots \tau_k$ are executed in w'.

All dates of transitions $t_1, \dots t_k$ are of the for $t_i = \lfloor t_i \rfloor + e_i$, with $e_i = frac(t_i) \in [0, 1)$ We define an ordering on values of fractional parts of dates, i.e. for all i, j, we know whether $frac(t_i) > frac(t_k)$, $frac(t_i) < frac(t_k)$, or $frac(t_i) = frac(t_k)$. The relation $<$ on fractional part of t_i's is acyclic/conflict free.

Now, let us prove that as long as w' respects ordering and equalities of fractional parts in w, plus the integral part of dates at which $\tau_1 \cdots \tau_k$ are done (only the fractional part can change), we are free to set the values of $frac(t'_1) \cdots frac(t'_k)$ as we want:

As $frac(t_k) = e > 0$ is possible (see w), it means that the constraints between $frac(t_k)$ and $frac(t_0)$ is $frac(t_k) > frac(t_0)$ ($frac(t_k) = frac(t_0) = 0$ is not possible, and neither is $frac(t_k) < frac(t_0) = 0$).

Hence we can set $frac(t'_k) = e'$, and set the other $frac(t'_i)$ accordingly:

1. for every i such that $frac(t_i) = frac(t_k)$, we set $frac(t'_i)$ to e', to get $frac(t'_i) = frac(t'_k)$.
2. Choose $i \in 1..k$ such that $frac(t_i) > frac(t_k)$ and there is no $j \in 1..k$ with $frac(t_k) < frac(t_j) < frac(t_i)$. Set e'_i to any value in $(e', 1)$,.
3. For every j such that $frac(t_j) = frac(t_i)$, set $e'_j = e'_i$.
4. Repeat steps 2-3, restarting from fractional part $frac(t_k)$ instead of $frac(t_i)$ and taking e_i instead of e' as minimal maximal value of a fractional part seen so far, till there is no j with $frac(t_j) > frac(t_i)$.
5. Do the same fractional values update for $frac(t_i) < frac(t_k)$, but setting every e'_i to decreasing values in $(0, e)$ at every step.

We can now show that if fractional parts of dates in w' have the same ordering as fractional parts of dates in w and the integral parts of dates in w and $w'$are the same, then the dates $t'_1, \dots t'_k$ in w' respect time constraints of every transition τ_i:

Case 1: Take any τ_i with some constraint $x = n$, with $n \in \mathbb{N}$. Take the latest τ_j such that $j < i$ resetting x. It means that $frac(t_i) = frac(t_j)$ in w, and as τ_i happens exactly n time units after τ_j, we have $e_i = e_j$, and hence $e'_i = e'_j$. As integral parts of dates are preserved, we have $t'_i - t'_j = n$, and hence the constraint $x = n$ is preserved in w'.

Case 2: Otherwise, the constraint is of the form $x \in (a, b)$. Take the latest τ_j such that $j < i$ resetting x. The constraint holds when firing τ_i iff $t_j - t_i \in (a, b)$. Let $n = \lfloor t_i \rfloor - \lfloor t_i \rfloor$ We necessarily have $n \in [a, b]$, and $t_i - t_j = n + e_i - e_j$. Consider $[t_i]$ and $[t_j]$.

- Assume that in w, we have $frac(t_i) > frac(t_j)$. It implies that $1 > e_i > e_j > 0$. After calculus of fractional values, we have $1 > e'_i > e'_j > 0$, and furthermore, $1 > e'_i - e'_j > e_i - e_j$. So, $t'_i - t'_j \in (n, n+1)$ and the constraint $x \in (a, b)$ is satisfied.
- Assume that in w, we have $frac(t_j) > frac(t_i)$. It implies that $1 > e_j > e_i > 0$. After calculus of fractional values, we still have $1 > e'_j > e'_i > 0$, hence and furthermore, $e_i - e_j > e'_i - e'_j > 0$. So, $t'_i - t'_j \in (n, n+1)$ and the constraint $x \in (a, b)$ is satisfied

As, for every transition t_i of σ, any constraint of the form $x = c$ or $x \in (a, b)$ appearing in guard g_i is satisfied by the clock valuations appearing in w', one can claim that $w' = (a_1, \lfloor t_1 \rfloor + e'1) \dots (a_k, \lfloor t_k \rfloor + e') \in \mathcal{L}(\mathcal{A})$.

D Integer Reset Automata

In integer reset automata(IRTA) the clocks can only reset on integer time stamps. Due to this restrictions, at any point of time the fractional values of all clocks must be same i.e., they can either be 0 or in (0,1). For a given IRTA $\mathcal{A}$ we can define a map $f : \rho \to w_{unt}$ that maps every run ρ of $\mathcal{A}$ 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 : $dt(x) = (\delta.\checkmark)^k$ 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 $w = (a_1, d_1) \dots (a_n, d_n)$ the word $f(w) = w_1.a_1.w_2.a_2 \dots w_n.a_n$ 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 $\checkmark$ and δ but keeps actions unchanged. Consider for example a timed word $\rho = (a, 1.5).(b, 2.6).(c, 2.6).(d, 3.4).(e, 3.5)$ then we can represent it as a word $f(\rho) = w = \delta\checkmark\delta a\checkmark\delta bc\checkmark\delta de$. It is shown in [15] for two timed words ρ_1, ρ_2 with $f(\rho_1) = f(\rho_2)$ then $\rho_1 \in \mathcal{L}(\mathcal{A})$ if and only if $\rho_2 \in \mathcal{L}(\mathcal{A})$. It is also shown that we can construct a Marked Timed Automata(MA)(Section 3) from $\mathcal{A}$ with one extra clock and polynomial increase in the number of locations such that $untime(\mathcal{L}(MA)) = f(\mathcal{L}(\mathcal{A}))$. The MA of $\mathcal{A}$ duplicates transitions of $\mathcal{A}$ to differentiates firing at integral/non integral dates, plus transitions that makes time elapsing visible using the additional clock which is reset at each global integral time stamp.

Lemma 5. *Given a timed automata $\mathcal{A}$ and a fault model P, $Mark(\mathcal{A})_P = Mark(\mathcal{A}_P)$.*

Proof. This proof is straight forward and comes from the construction of a Marked Automaton from a given automaton and the definition of enlargement of the automaton upon a fault model.

Definition 14. *For a given word $w \in \Sigma^*$ and a subset $\Sigma' \subseteq \Sigma$, w_s is a projection of w on Σ' if w_s contains all the letters from Σ' in same order as in w and nothing else. We will use $w \downarrow_{\Sigma'}$ to represent a projection of the word $w \in \Sigma$ to $\Sigma' \subseteq \Sigma$.*

Example 1. Assume $w = \delta\checkmark\delta a\checkmark\delta bc\checkmark\delta de$, and $\Sigma' = \{\checkmark, \delta\}$ then the projection of w to Σ' will be $w \downarrow_{\{\checkmark,\delta\}} = \delta\checkmark\delta\checkmark\delta\checkmark\delta$.

Lemma 6. *For two IRTA $\mathcal{A}$ and $\mathcal{B}$ and their corresponding marked automata $\mathcal{A}_M$ and $\mathcal{B}_M$, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ if and only if $untime(\mathcal{L}(\mathcal{A}_M)) \subseteq untime(\mathcal{L}(\mathcal{B}_M))$.*

Proof. ($\Rightarrow$) Assume, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ and assume there exists a word $w \in untime(\mathcal{L}(\mathcal{A}_M))$, but $w \notin untime(\mathcal{L}(\mathcal{B}_M))$. Now, there exists a timed word $\rho \in \mathcal{L}(\mathcal{A})$ such that,

$f(\rho) = w$. Clearly, $\rho \in \mathcal{L}(\mathcal{B})$, then clearly $f(\rho) = w \in \mathit{untimed}(\mathcal{L}(\mathcal{B}_M))$ a contradiction. So, $\mathit{untime}(\mathcal{L}(\mathcal{A}_m)) \subseteq \mathit{untime}(\mathcal{L}(\mathcal{B}_m))$.

($\Leftarrow$)Assume, $\mathit{untime}(\mathcal{L}(\mathcal{A}_M)) \subseteq \mathit{untime}(\mathcal{L}(\mathcal{B}_M))$, and $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. Then, there exists a timed word $\rho \in \mathcal{L}(\mathcal{A})$ such that $\rho \notin \mathcal{L}(\mathcal{B})$. Assume $f(\rho) = w$, then clearly, $w \in \mathit{untime}(\mathcal{L}(\mathcal{A}_M))$ and $w \in \mathit{untime}(\mathcal{L}(\mathcal{B}_M))$. So, there exists a timed word $\rho' \in \mathcal{L}(\mathcal{A})$ such that, $f(\rho') = w = f(\rho)$. According to the Theorem 3 of [15] we can conclude that, $\rho \in \mathcal{L}(\mathcal{B})$ a contradiction.

Remark 5. The Lemma 6 shows that the timed and untimed language inclusion problem for IRTA are same problem. So, we can solve the timed language inclusion problem by solving an untimed language inclusion problem of IRTA and vice-versa hence, the untimed language inclusion is also EXPSPACE-Complete.

E Resilience of IRTA

The proofs of resilience for IRTA will rely on the following property proved in [15]

Theorem 10 (Thm.3, [15]). *If $\mathcal{A}$ is an IRTA and $f(w) = f(w')$, then $w \in \mathcal{L}(\mathcal{A})$ if and only if $w' \in \mathcal{L}(\mathcal{A})$*

Definition 15. *For two automata $\mathcal{A}$ and $\mathcal{B}$ with common set of action Σ we define a synchronized product automata $\mathcal{A} \times \mathcal{B}$ such that there exists a transition $(l_a, l_b) \xrightarrow{a} (l'_a, l'_b)$ in $\mathcal{A} \times \mathcal{B}$ if and only if there exists a transition $l_a \xrightarrow{a} l'_a$ in $\mathcal{A}$ and a transition $l_b \xrightarrow{a} l'_b$ in $\mathcal{B}$ for all $a \in \Sigma$.*

Definition 16. *For two automata $\mathcal{A}$ and $\mathcal{B}$ with a common set of actions we define semi-synchronized product automata $\mathcal{A} \times_{\Sigma'} \mathcal{B}$ such that the transitions are synchronized over only a subset of actions $\Sigma' \subseteq \Sigma$. For every transitions of $\mathcal{A} \times_{\Sigma'} \mathcal{B}$ can then be partitioned in three categories,*

1. *$(l_a, l_b) \xrightarrow{a} (l'_a, l'_b)$ in $\mathcal{A} \times_{\Sigma'} \mathcal{B}$ for all $a \in \Sigma'$ if and only if there exists a transition $l_a \xrightarrow{a} l'_a$ in $\mathcal{A}$ and a transition $l_b \xrightarrow{a} l'_b$ in $\mathcal{B}$.*
2. *$(l_a, l_b) \xrightarrow{b} (l'_a, l_b)$ in $\mathcal{A} \times_{\Sigma'} \mathcal{B}$ for all $b \in \Sigma \setminus \Sigma'$ if and only if there exists a transition $l_a \xrightarrow{b} l'_a$ in $\mathcal{A}$*
3. *$(l_a, l_b) \xrightarrow{b} (l_a, l'_b)$ in $\mathcal{A} \times_{\Sigma'} \mathcal{B}$ for all $b \in \Sigma \setminus \Sigma'$ if and only if there exists a transition $l_b \xrightarrow{b} l'_b$ in $\mathcal{B}$*

Definition 17. *Just like Definition 16 we can define $n-$semi-synchronized product where the number of automata in that product is equal to $n \in \mathbb{N}$.*

Remark 6. We can easily extend the n-semi-synchronized product automata to count the number of steps after the fault. For that we will have to include a counter in each state that counts the number of steps in the faulty automata till the fault happened. The value of the counter is -1 in the start state, and resets to 0 when a faulty transition fired in $\mathcal{R}(\mathcal{A}_P)$. Then, it continues to increase the

count for each transition fired by the faulty automata. And stops when it reaches the count $K+1$, and stays in this counter value for the rest of the transitions. For example the states of 2-semi-synchronized product automata can be represented as $((l,r)(l',r'),C)$ where, C represents the counter value, and for a state the counter value $C = K$ when the state is reachable from the initial state after K steps after a fault. We represent the 2-semi-synchronized product automata with a counter as $\mathcal{A}_f \times_{C-\Sigma'} \mathcal{A}$ where, C is the steps after the fault and Σ' is the subset of action on which we are synchronizing the product.

Definition 18. *Let $Act \subseteq \Sigma$ be a set of actions, ρ be a faulty timed word accepted by $\mathcal{A}_P$ of length n, such that $w = untime(\rho) = w_1.w_2$ with $w_1 \in \mathcal{L}^{1F}(\mathcal{A}_\mathcal{P})$. We say that ρ is untimed $K-prefix-BTN$ (w.r.t. Act) if there exists a time word ρ_i in $\mathcal{A}$, where, , $w_i = untime(\rho_i) = w_{pre}.w_{suff}$, and $w_{[|w_1|+K+1,n]} = w_{suff}$, such that, $w_{pre} \downarrow_{Act} = w_{[1,|w_1|+k]} \downarrow_{Act}$.*

Definition 19. *An automata $\mathcal{A}$ along with a fault model $\mathcal{P}$, is untimed $K-prefix-resilient$ if every faulty word $\rho \in \mathcal{A}_P$ is untimed $K-prefix-BTN$.*

Theorem 11. *For a timed automata $\mathcal{A}$, and a fault model $\mathcal{P}$ checking if the automata is untimed $K-prefix-resilient$ is in PSPACE.*

Proof. Let us assume a timed word ρ that ends after K steps of a fault in $\mathcal{A}_P$ i.e, $\rho \in \mathcal{L}^{1F}(\mathcal{A}_\mathcal{P})$. Also without the loss of generality assume, that ρ ends in a state $q_\rho = (l_\rho, r_\rho)$ of $\mathcal{R}(\mathcal{A}_P)$. To check if this run is untimed $K-prefix-BTN$, we need to first check if we can find a run ρ' in $\mathcal{A}$ with $untime(\rho') \downarrow_{Act} = untime(\rho) \downarrow_{Act}$, where Act is the set of actions for which we are checking untimed $K-prefix-resilience$. This can be solved by constructing a semi-synchronized product automaton(c.f., Definition 16) of the region automata $\mathcal{R}(\mathcal{A}_P) \times_{Act} \mathcal{R}(\mathcal{A})$ such that they are only synchronized over the actions $a \in Act$. And check if we can reach any state $((l_\rho, r_\rho)(l', r'))$ in that semi-synchronzed product automaton, where, (l', r') is a state of $\mathcal{R}(\mathcal{A})$. Once we find such a state pair $((l_\rho, r_\rho)(l', r'))$, we need to check if from this state pair both automata follows same sequence of action to reach a pair of their respective final states e.g., $\Big((l_f, r)(l'_f, r'_f)\Big)$ where, l_f, l'_f are the final states of the automata $\mathcal{A}_P$ and $\mathcal{A}$ respectively. This checking can be done using a reachability query in synchronized product automata(c.f, Definition 15). If, we can reach such final states this means the corresponding fault is untimed $K-prefix-BTN$, but, if we can not reach such a final state, then it is possible that there exists another state (l'', r'') in $\mathcal{R}(\mathcal{A})$ which can be reached in $\mathcal{R}(\mathcal{A})$ following a run ρ'' of $\mathcal{A}$ such that $untime(\rho'') \downarrow_{Act} = untime(\rho) \downarrow_{Act}$. We can find such states by a $3-$semi-synchronized product automata $\mathcal{R}(\mathcal{A}_p) \times_{K-Act} \mathcal{R}(\mathcal{A}) \times_{K-Act} \mathcal{R}(\mathcal{A})$ and checking reachability of the state $((l_\rho, r_\rho)(l', r')(l'', r''), K)$(c.f., Algorithm 1). If we still can not reach a final state for all such possible $((l_\rho, r_\rho)(l'', r''))$ then the fault is not $K-prefix-BTN$, and there exists a fault which is not $K-prefix-BTN$ and the automata $\mathcal{A}$ is not untimed $K-prefix-resilient$. Using an ordering on regions (such as the order proposed in Appendix A), one can enumerate states of $\mathcal{R}^K(\mathcal{A}_\mathcal{P}) \times \mathcal{R}(\mathcal{A})$ in PSPACE. For each of the state pairs $((l_\rho, r_\rho)(l', r'))$, one can check if this pair

is reachable from the initial state of the semi-synchronized product automata as discussed earlier. This is a general timed reachability query and known to be in PSPACE. Then, we need to check if any final state pair $\Big((l_f, r)(l'_f, r'_f)\Big)$ is reachable from the state pair $((l_\rho, r_\rho)(l', r'))$ in the fully synchronized product automation. This is also a timed reachability query and can be done in PSPACE. And if this is not reachable then we need to call the joint reachability query to for all possible states which are reachable by a run ρ'' with $untimed(\rho'') \downarrow_{Act} = untimed(\rho) \downarrow_{Act}$. Algorithm 1 solves the problem of joint reachability and it is easy to verify that this algorithm also is in PSPACE. Hence, the problem untimed $K - prefix\_rsilience$ can be solved in PSPACE.

Lemma 7. *An IRTA $\mathcal{A}$ is $K-\exists-resilient$ if and only if the Marked Timed Automata of $\mathcal{A}$ is untimed $K-prefix-resilient$(c.f. Definition 19 in Appendix D) with respect to the actions $\{\checkmark, \delta\}$*

Proof. ($\Rightarrow$)Assume a timed word $\rho \in \mathcal{L}(\mathcal{A}_\mathcal{P})$ that comes back to normal within K steps of the fault (where steps are only counted for the transitions that belongs to the original automata $\mathcal{A}$,) and matches with the suffix of a timed word $\rho' \in \mathcal{L}(\mathcal{A})$. From the definition of marked automata[15] $f(\rho) = w \in untimed(\mathcal{L}(MA_p))$ and $f(\rho') = w' \in untimed(\mathcal{L}(MA))$. We can get a suffix w_{suff} of w that start with the action of $(K+1)^{th}$ transition of $\mathcal{A}_P$ after fault. We define prefix w_k such that $w = w_k.w_{suff}$. Similarly, we can partition $f(\rho') = w'$ in to a suffix $w'_{suff} = w_{suff}$ and a prefix w'_{pre}, clearly both the words ρ and ρ' elapse same time in the respective automata before coming back to normal hence it is easy to check that $w_K \downarrow_{\{\checkmark,\delta\}} = w'_{suff} \downarrow_{\{\checkmark,\delta\}}$. Hence, if all words of $\mathcal{L}(\mathcal{A}_\mathcal{P})$ come back to normal within K steps after a fault, the marked automaton is untimed $K - prefix - resilient$.
($\Leftarrow$) Assume a word $w \in untimed(\mathcal{L}(MA_p))$ which is $K - prefix - BTN$ with respect to the set of actions $\{\checkmark, \delta\}$ and the steps measured as the number of $\mathcal{A}_P$ transitions fired along the run (i.e., we ignore the step count of other transitions). We can split w in two parts, one till K steps after the fault (denoted by w_k hereafter) and the remaining part, w_e such that, $w = w_k.w_e$. Then, according to the definition of $K - Prefix - BTN$, there exists a word $w' = w'_{pre}.w'_{suff} \in untimed(\mathcal{L}(MA))$ such that, $w_e = w'_{suff}$ and $w'_{pre} \downarrow_{\checkmark,\delta} = w_e \downarrow_{\checkmark,\delta}$. Note that the suffix w_e must start with an action $a \notin \{\delta, \checkmark\}$ because, we are counting the steps in terms of the actions of the automaton $\mathcal{A}_P$ not in terms of the actions of marked automata MA_P and δ and $\checkmark$ actions are part of marked automata, hence, the $K + 1^{th}$ step must be an action from the automata $\mathcal{A}$. Because, $w \in untimed(\mathcal{L}(MA_P))$ then, there exists a timed word $\rho \in \mathcal{L}(\mathcal{A}_P)$ such that, $f(\rho) = w$. Similarly, there also exists a timed word $\rho' \in \mathcal{L}(\mathcal{A})$ such that, $f(\rho') = w' = w'_{pre}.w'_{suff}$. Without loss of generality we can assume a timed word $\rho_f \in \mathcal{L}(\mathcal{A}_P)$ such that, $f(\rho_f) = w$ and ρ_{suff} represents the suffix of ρ_f starting K steps after the fault. Note that if the first event of ρ_{suff} is (a', τ'), then the first letter of $w_e = w'_{suff}$ will also be a', because in both cases, a' is the action $K + 1^{th}$ action of $\mathcal{A}_P$ after the fault. Given all this, we can construct a timed word ρ'' such that, (1) $f(\rho'') = w'$ (2) There exists a suffix ρ''_{suff} of ρ''

such that $\rho''_{suff} = \rho_{suff}$. We can start by selecting the suffix $\rho''_{suffix} = \rho_{suff}$ including the time stamps and then we will construct a prefix ρ''_{pre} from w'_{pre}. Recall that $w'_{pre} \downarrow_{\checkmark,\delta} = w_e \downarrow_{\checkmark,\delta}$ and the next event after the prefix is the first event of ρ_{suffix}, assume it be (a', τ'). It is easy to assign the time stamps to the actions of w'_{pre} except the actions after the last ✓ of w'_{pre} (1) We will start from the first non δ,✓ action of w'_{pre} and assign it a time stamp $M \in \mathbb{N}$ if there are M occurrences of ✓ before it and no δ between the action and the M^{th} ✓ (2) assign $M + \epsilon$ where $\epsilon \in (0, 1)$ if there are M ✓ before it and if there exists a δ between the M^{th} ✓ and the action. Assure that, every actions between a ✓ and its nearest δ has same time stamps. Now, we need to take special attention before assigning the time stamps to the actions after the last ✓ of w'_{pre}. If, this actions does not have any δ between the last ✓ and itself, then it's time stamp is integer and equal to the number of ✓ in w'_{pre}. But, it has a δ between the ✓ and the action in w'_{pre} then, its time stamp can be equal to the timestamp of the first event of ρ_{suff}, which is τ' here. It is easy to check that $f(\rho''_{pre}) = w'_{pre}$. Hence, the newly created run will be $\rho'' = \rho''_{pre}.\rho''_{suff}$. Clearly, $f(\rho'') = w'$ and because there exists a run $\rho' \in \mathcal{L}(\mathcal{A})$ such that $f(\rho') = w'$, $\rho'' \in \mathcal{L}(\mathcal{A})$(Theorem 10) and hence proved.

Now, suppose that all words in $untime(\mathcal{L}(MA_P))$ are $K - prefix - BTN$, but there exist a faulty word $w \in \mathcal{L}(\mathcal{A}_\mathcal{P})$ which is not BTN K steps after a fault. Let $w = w_1.w_2$ where w_2 is the suffix of w $K-$steps after the fault in w. By construction of MA, we know that $f(w) \in untime(\mathcal{L}(MA_P))$, and $f(w)$ is $K - prefix - BTN$. Hence, one can build a timed word $w'_1.w'_2 \in \mathcal{L}(\mathcal{A})$ such that $f(w'_1) = f(w_1)$ and $f(w'_2) = f(w_2)$. Following Theorem 10, one can choose dates for w'_1, w_2 such that the duration of w'_1 is exactly that of w_2 and such that $w'_2 = w_2$, which contradicts the fact that w is not BTN.

Theorem 12. *The $K - \exists$ timed resilience for IRTA is PSPACE Complete.*

Proof. Containment: From Lemma 7 we can conclude that by solving untimed $K - prefix - resilience$ of Marked Timed Automata(MA) we can solve, timed existential resilience of IRTA. The size of MA is polynomial to the size of the underlying IRTA $\mathcal{A}$ and it is already shown that solving untimed $K - prefix - resilience$ is in PSPACE(Appendix D, Theorem 11). Hence, the problem of timed existential resilience is in PSPACE for Integer Reset Automata.

Hardness: The hardness proof is similar to the hardness proof of Theorem 2, but we need to modify the gadget so that it satisfies the IRTA properties. Then, we can reduce the problem of emptiness checking of IRTA to a existential resilience problem of IRTA.

Theorem 13. *The timed $K - \forall - resilience$ for IRTA is EXPSPACE-hard.*

Proof. The proof is obtained by a reduction from the inclusion problem of IRTA, known to be EXPSPACE-Complete[5]. The idea of the proof follows the same lines as the untimed $K - \forall - resilience$ of timed automata. Assume we are given IRTA $\mathcal{A}_1, \mathcal{A}_2$. a, b, c are symbols not in the alphabets of $\mathcal{A}_1, \mathcal{A}_2$. It is easy to

Algorithm 1: semi-joint-reach$(((l_B, r_B)(l_A, r_A)), \mathcal{A}_P, \mathcal{A}, Act)$

```
1  S = ((l_0, r_0)(l'_0, r'_0)(l'_0, r'_0), −1);
2  n=0;
3  while n < Enum(A) do
4      Select state (l_i, r_i) indexed by n.;
5      if ((l_B, r_B)(l_A, r_A)(l_i, r_i), K)is reachable from S in
         A_P ×_{K−Act} A ×_{K−Act} A then
6          if ((l_f, r_f)(l'_f, r'_f)) is reachable from ((l_B, r_B)(l_i, r_i)) in A_P × A then
7              return True;
8      else
9          n=n+1;
10 return False;
```

Algorithm 2: K-prefix-resil$(\mathcal{A}_P, \mathcal{A}, Act)$

```
1  S = ((l_0, r_0)(l'_0, r'_0), −1);
2  n=0;
3  while n < Enum(A_P × A × {−1, 0, 1...K + 1}) do
4      Select state ((l_ρ, r_ρ)(r', l'), C) indexed by n.;
5      if C = K then
6          if ((l_ρ, r_ρ)(r', l'), K)is reachable from S in A_P ×_{K−Act} A then
7              if ((l_f, r_f)(l'_f, r'_f)) is reachable from ((l_ρ, r_ρ)(l', r')) in A_P × A
                 then
8                  n=n+1;
9              else
10                 if semi-joint-reach(((l_ρ, r_ρ)(l', r')), A_P, A, Act) then
11                     n=n+1;
12                 else
13                     return False;
14         else
15             n=n+1;
16 return True;
```

see that $L(\mathcal{B})$=$(a,1).(b,1).(c,11).(L(\mathcal{A}_1)+11)$, where $L(\mathcal{A}_1)+k = \{(a_1,d_1+k)(a_2,d_2+k)\ldots(a_n,d_n+k) \mid (a_1,d_1)\ldots(a_n,d_n) \in L(\mathcal{A}_1)\}$. Associate a fault model $\mathcal{P}(a)=1$, where the fault of a is 1. We construct an IRTA $\mathcal{B}_\mathcal{P}$ as shown in Figure 6. Notice that in general, IRTAs are not closed under the fault operation; the enlarged guard in $\mathcal{B}$ 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 $\mathcal{B}_\mathcal{P}$, we prune away that transition resulting in $\mathcal{B}_\mathcal{P}$ as in Figure 6. Indeed, this resulting faulty automaton is an IRTA.

The language accepted by $\mathcal{B}_\mathcal{P}$ is $L(\mathcal{B}) \cup (a,2).(b,2).(c,11).(L(\mathcal{A}_2)+11)$. Considering $K=2$, $\mathcal{B}_\mathcal{P}$ is BTN in 2 steps after the fault iff $L(\mathcal{A}_2) \subseteq L(\mathcal{A}_1)$. The EXPSPACE hardness of the timed $K-\forall-resilience$ of IRTA follows from the EXPSPACE completeness of the inclusion of IRTA.

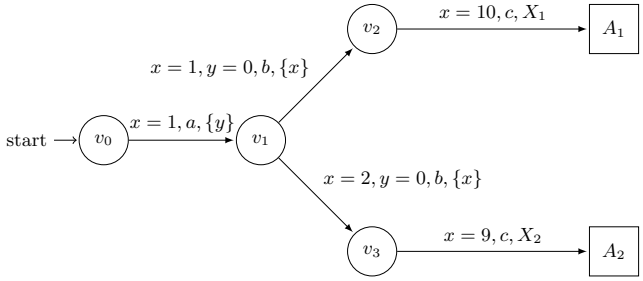


Fig. 5. The automaton B

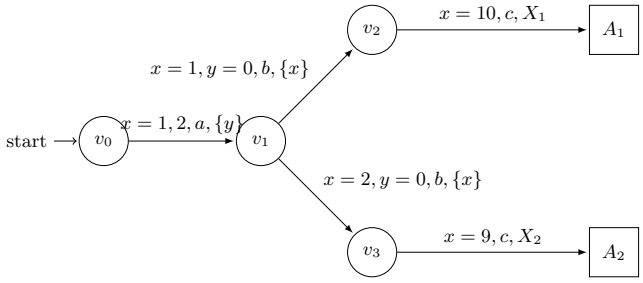


Fig. 6. The faulty automaton $B_\mathcal{P}$.

Theorem 14. *The timed $K-\exists-resilience$ for IRTA is PSPACE-hard.*

Proof. Consider an IRTA $\mathcal{A}$ with alphabet Σ and the construction of an automata that uses a gadget shown below in Figure 7. Let us call this automaton

$\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}$. It is easy to see that the $L(\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}})$=$(a,1).(b,1).(c,11).((\Sigma \times \mathbb{R})^* + 11)$, where $L(A_1) + k = \{(a_1, d_1 + k)(a_2, d_2 + k) \dots (a_n, d_n + k) \mid (a_1, d_1) \dots (a_n, d_n) \in L(A_1)\}$. The Σ loop on a particular accepting state q_e is responsible for acceptance of all timed word. Now, associate a fault model $\mathcal{P}(a) \to 1$ with $\mathcal{B}$, where the fault of a is 1. Let us call this enlarged automaton $\mathcal{B}_{(\Sigma^* \subseteq \mathcal{A})_P}$. 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 8. The language accepted by $\mathcal{B}_{(\Sigma^* \subseteq \mathcal{A})_P}$ is $L(\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}) \cup (a,2).(b,2).(c,11), (L(\mathcal{A}) + 11)$. The accepting states are $q_e \cup F$, where F is the set of final states of $\mathcal{A}$. Then $\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}$ is $K - \exists - resilient$ if and only if $L(\mathcal{A}) \neq \emptyset$.

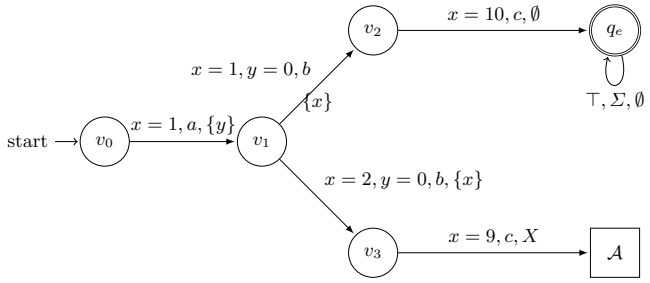


Fig. 7. The automaton $\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}$

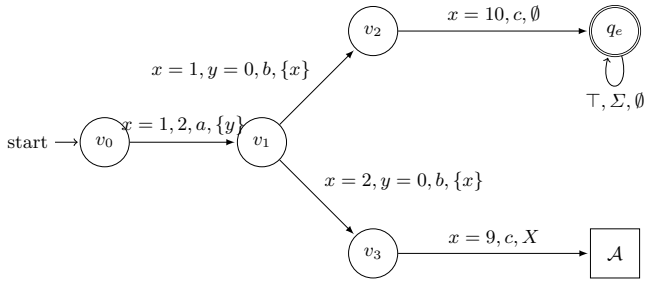


Fig. 8. The faulty automaton $\mathcal{B}_{(\Sigma^* \subseteq \mathcal{A})_P}$

Remark 7. The untimed language inclusion problem is shown to be EXPSPACE-Complete in Remark 5. 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 $K - \exists - resilient$ 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 used in Theorem 13 and Theorem 14 respectively.