Resilience of Timed Systems

S. Akshay
IIT Bombay, Mumbai, India

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

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

S. Krishna
IIT Bombay, Mumbai, India

Sparsa Roychowdhury
IIT Bombay, Mumbai, India

Abstract

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

2012 ACM Subject Classification Theory of computation → Timed and hybrid models

Keywords and phrases Timed automata; Fault tolerance; Integer-resets; Resilience

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.33

Funding This work was supported by the DST/CEFIPRA/INRIA associated team EQuaVE

1 Introduction

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

Several works have addressed timing errors as a question of *robustness* [10, 8, 7], to guarantee that a property of a system is preserved for some small imprecision of up to ϵ time units. Timed automata have an ideal representation of time: if a guard of a transition contains a constraint of the form $x = 12$, it means that this transition occurs *exactly* when

41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2021).
Editors: Mikołaj Bojańczyk and Chandra Chekuri; Article No. 33; pp. 33:1–33:22
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

	Universal Resilience	Existential Resilience
Timed	Undecidable for TA (Prop. 18) EXPSPACE-C for IRTA (Thm. 20)	EXPSPACE (Thm. 14) PSPACE-Hard (Thm. 15, Thm. 32)
Untimed	EXPSPACE-C (Thm. 21)	PSPACE-C (Thm. 16, Rmk. 17)

Table 1 Summary of results for resilience.

the value of clock x is 12. Such an arbitrary precision is impossible in an implementation [10]. One way of addressing this is through guard enlargement, i.e., by checking that there exists a small value $\epsilon > 0$ such that after replacing guards of the form $x \in [a, b]$ by $x \in [a - \epsilon, b + \epsilon]$, the considered property is still valid, as shown in [7] for ω-regular properties. In [15], robust automata are defined that accept timed words and their neighbors i.e., words whose timing differences remain at a small distance, while in [16, 12, 19, 1], the authors consider robustness via modeling clock drifts. Our goal is different: rather than being robust w.r.t. to slight imprecisions, we wish to check the capacity to recover from a *possibly large* time deviation. Thus, for a bounded number of steps, the system can deviate arbitrarily, after which, it must return to its specified timed behavior.

The first contribution of this paper is a formalization of resilience in timed automata. We capture delayed events with *faulty transitions*. These occur at dates deviating from the original specification and may affect clock values for an arbitrarily long time, letting the system diverge from its expected behavior. A system is *resilient* if it recovers in a finite number of steps after the fault. More precisely, we define two variants. A timed automaton is K-$\forall$-*resilient* if for *every* faulty timed run, the behavior of the system K steps after the fault cannot be distinguished from a non-faulty behavior. In other words, the system *always* repairs itself in at most K steps after a fault, whenever a fault happens. This means that, after a fault happens, *all* the subsequent behaviors (or extensions) of the system are restored to normalcy within K steps. A timed automaton is K-$\exists$-*resilient* if for every timed run ending with a fault, there exists *an* extension in which, the behavior of the system K steps after the fault cannot be distinguished from a non-faulty behavior. There can still be some extensions which are beyond repair, or take more than K steps after fault to be repaired, but there is a guarantee of at least one repaired extension within K steps after the fault. In the first case, the timed automaton is fully self-resilient, while in the second case, there exist controllers choosing dates and transitions so that the system gets back to a normal behavior. We also differentiate between timed and untimed settings: in timed resilience recovered behaviors must be indistinguishable w.r.t. actions and dates, while in untimed resilience recovered behaviors only need to match actions.

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

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

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

2 Preliminaries

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

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

A *valuation* of a set of clocks X is a map $\nu : X \to \mathbb{R}_{\geq 0}$ that associates a non-negative real value to each clock in X. For every clock x, $\nu(x)$ has an integral part $\lfloor \nu(x) \rfloor$ and a fractional part $\mathsf{frac}(\nu(x)) = \nu(x) - \lfloor \nu(x) \rfloor$. We will say that a valuation ν on a set of clocks X satisfies a guard g, denoted $\nu \models g$ if and only if replacing every $x \in X$ by $\nu(x)$ in g yields a tautology. We will denote by $[g]$ the set of valuations that satisfy g. Given $\delta \in \mathbb{R}_{\geq 0}$, we denote by $\nu + \delta$ the valuation that associates value $\nu(x) + \delta$ to every clock $x \in X$. A *configuration* is a pair $C = (l, \nu)$ of a location of the automaton and valuation of its clocks. The semantics of a timed automaton is defined in terms of discrete and timed moves from a configuration to the next one. A *timed move of duration δ* lets $\delta \in \mathbb{R}_{\geq 0}$ time units elapse from a configuration $C = (l, \nu)$ which leads to configuration $C' = (l, \nu + \delta)$. A *discrete move* from configuration

$C = (l, \nu)$ consists of taking one of the transitions leaving l, i.e., a transition of the form $t = (l, g, a, R, l')$ where g is a guard, $a \in \Sigma$ a particular action name, R is the set of clocks reset by the transition, and l' the next location reached. A discrete move with transition t is allowed only if $\nu \models g$. Taking transition t leads the automaton to configuration $C' = (l', \nu')$ where $\nu'(x) = \nu(x)$ if $x \notin R$, and $\nu'(x) = 0$ otherwise.

▶ **Definition 2** (Runs, Maximal runs, Accepting runs)**.** *An (infinite)* run *of a timed automaton $\mathcal{A}$ is a sequence $\rho = (l_0, \nu_0) \xrightarrow{(t_1,d_1)} (l_1, \nu_1) \xrightarrow{(t_2,d_2)} \cdots$ where every pair (l_i, ν_i) is a configuration, and there exists an (infinite) sequence of timed and discrete moves $\delta_1.t_1.\delta_2.t_2 \ldots$ in $\mathcal{A}$ such that $\delta_i = d_{i+1} - d_i$, and a timed move of duration δ_i from (l_i, ν_i) to $(l_i, \nu_i + \delta_i)$ and a discrete move from $(l_i, \nu_i + \delta_i)$ to (l_{i+1}, ν_{i+1}) via transition t_i. A run is* maximal *if it is infinite, or if it ends at a location with no outgoing transitions. A finite run is* accepting *if its last location is final, while an infinite run is* accepting *if it visits accepting locations infinitely often.*

We assume that all runs start from a configuration (l_0, ν_0), where $l_0 \in I$ and ν_0 is the initial valuation, assigning value 0 to every clock of X. One can associate a finite/infinite *timed word* w_ρ to every run ρ of $\mathcal{A}$ by letting $w_\rho = (a_1, d_1)\ (a_2, d_2) \ldots (a_n, d_n) \ldots$, where a_i is the action in transition t_i and d_i is the time stamp of t_i in ρ. A (finite/infinite) *timed word* w is accepted by $\mathcal{A}$ if there exists a (finite/infinite) accepting run ρ such that $w = w_\rho$. The *timed language* of $\mathcal{A}$ is the set of all timed words accepted by $\mathcal{A}$, and is denoted by $\mathcal{L}(\mathcal{A})$. The *untimed language* of $\mathcal{A}$ is the language $Unt(\mathcal{L}(\mathcal{A})) = \{Unt(w) \mid w \in \mathcal{L}(\mathcal{A})\}$. As shown in [2], the untimed language 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 in an atomic constraint of a guard of $\mathcal{A}$ involving x. Two valuations ν, ν' of clocks in X 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) $\forall x, y \in X$ with $\nu(x) \leq c_x$ and $\nu(y) \leq c_y$, $\mathsf{frac}(\nu(x)) \leq \mathsf{frac}(v(y))$ iff $\mathsf{frac}(\nu'(x)) \leq \mathsf{frac}(\nu'(y))$
iii) For all $x \in X$ with $\nu(x) \leq c_x$, $\mathsf{frac}(\nu(x)) = 0$ iff $\mathsf{frac}(\nu'(x)) = 0$.

A *region* r of $\mathcal{A}$ is the equivalence class induced by $\sim$. For a valuation ν, we denote by $[\nu]$ the region of ν, i.e., its equivalence class. We will also write $\nu \in r$ (ν is a valuation in region r when $r = [\nu]$. 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 for a clock constraint ψ that, if $\nu \sim \nu'$, then $\nu \models \psi$ if and only if $\nu' \models \psi$. A region r' is a *time successor* of another region r if for every $\nu \in r$, there exists $\delta \in \mathbb{R}_{>0}$ such that $\nu + \delta \in r'$. We denote by $Reg(X)$ the set of all possible regions of the set of clocks X. A region r satisfies a guard g if and only if there exists a valuation $\nu \in r$ such that $\nu \models g$. The region automaton of a timed automaton $\mathcal{A} = (L, I, X, \Sigma, T, F)$ is the untimed automaton $\mathcal{R}(\mathcal{A}) = (S_R, I_R, \Sigma, T_R, F_R)$ that recognizes the untimed language $Unt(\mathcal{L}(\mathcal{A}))$. States of $\mathcal{R}(\mathcal{A})$ are of the form (l, r), where l is a location of $\mathcal{A}$ and r a region, i.e., $S_R \subseteq L \times Reg(X), I_R \subseteq I \times Reg(X)$, and $F_R \subseteq F \times Reg(X)$. The transition relation T_R is such that $\big((l, r), a, (l', r')\big) \in T_R$ if there exists a transition $t = (l, g, a, R, l') \in T$ such that there exists a time successor region r'' of r such that r'' satisfies the guard g, and r' is obtained from r'' by resetting values of clocks in R. The size of the region automaton is the number of states in $\mathcal{R}(\mathcal{A})$ and is denoted $|\mathcal{R}(\mathcal{A})|$. For a region r defined on a set of clocks Y, we define a projection operator $\Pi_X(r)$ to represent the region r projected on the set of clocks $X \subseteq Y$. Let $\rho = (l_0, \nu_0) \xrightarrow{(t_1,d_1)} (l_1, \nu_1) \cdots$ be a run of $\mathcal{A}$, where every t_i is of the form $t_i = (l_i, g_i, a_i, R_i, l'_i)$. The *abstract run* $\sigma_\rho = (l_0, r_0) \xrightarrow{a_1} (l_1, r_1) \cdots$ of ρ is a path in the region automaton $\mathcal{R}(\mathcal{A})$ such that, $\forall i \in \mathbb{N}, r_i = [\nu_i]$. We represent runs using variables ρ, π and the corresponding abstract runs with σ_ρ, σ_π respectively. The automaton $\mathcal{R}(\mathcal{A})$ can be used to prove non-emptiness of $\mathcal{L}(\mathcal{A})$, as $\mathcal{L}(\mathcal{A}) \neq \emptyset$ iff $\mathcal{R}(\mathcal{A})$ accepts some word.

3 Resilience Problems

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

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

For simplicity, we consider only executions in which a single timing error occurs. The perturbed semantics defined below easily adapts to a setting with multiple timing errors. With a fault model, we can define a new timed automaton, for which every run $\rho = (l_0, \nu_0) \xrightarrow{(t_1,d_1)} (l_1, \nu_1) \xrightarrow{(t_2,d_2)} \cdots$ contains *at most* one transition $t_i = (l, g, a, r, l')$ occurring later than allowed by guard g, and agrees with a run of $\mathcal{A}$ until this faulty transition is taken.

▶ **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 $\phi_{\rhd\delta}$ 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 $g_{\rhd\delta} = \bigwedge_{i \in 1..m} \phi_{i_{\rhd\delta}} \wedge \bigwedge_{j \in 1..q} \phi_j$*

For every transition $t = (l, g, a, R, l')$ with enlarged guard

$$g_{\rhd\mathcal{P}(a)} = \bigwedge_{i \in 1..m} \phi_i = a_i \bowtie_{l_i} x_i \leq b_i + \mathcal{P}(a) \wedge \bigwedge_{j \in 1..q} \phi_j = a_j \bowtie_{l_j} x_j - y_j \bowtie_{u_j} b_j,$$

we can create a new transition $t_{f,\mathcal{P}} = (l, g_{f,\mathcal{P}}, a, R, \overset{\bullet}{l'})$ called a faulty *transition such that,*

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

Diagonal constraints remain unchanged under enlargement, as the difference between clocks x and y is preserved by time elapsing, and operator $\bar{\bowtie}_{l_i}$ guarantee that normal and faulty behaviors occur at different dates. From now, we fix a fault model $\mathcal{P}$ and 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 $g_{\rhd\delta}$. We take this particular definition of enlargement to consider late events as faults. We can easily adapt the definition to handle early events, or any variation where non-specified faulty transitions can be identified through a guard g_f disjoint from g, without harming the results shown in the rest of the paper.

▶ **Definition 5** (Enlargement of automata). *Let $\mathcal{A} = (L, I, X, \Sigma, T, F)$ be a timed automaton. The enlargement of $\mathcal{A}$ by a fault model $\mathcal{P}$ is the automaton $\mathcal{A}_\mathcal{P} = (L_\mathcal{P}, I, X, \Sigma, T_\mathcal{P}, F_\mathcal{P})$, where*

- *$L_\mathcal{P} = L \cup \{\overset{\bullet}{l} \mid l \in L\}$ and $F_\mathcal{P} = F \cup \{\overset{\bullet}{l} \mid l \in F\}$. A location $\overset{\bullet}{l}$ indicates that an unexpected delay has occurred.*
- *$T_\mathcal{P} = T \cup \overset{\bullet}{T}$ such that, $\overset{\bullet}{T} = \{(l, g_f, a, R, \overset{\bullet}{l'}) \mid (l, g, a, R, l') \in T\} \cup \{(\overset{\bullet}{l}, g, a, R, \overset{\bullet}{l'}) \mid (l, g, a, R, l') \in T\}$ i.e., $\overset{\bullet}{T}$ is the set of transitions occurring after a fault.*

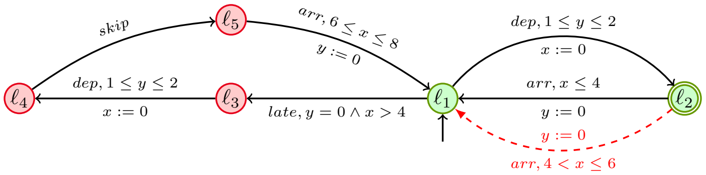

Figure 1 Model of a train system with a mechanism to recover from delays

A run of $\mathcal{A}_\mathcal{P}$ is *faulty* if it contains a transition of $\dot{T}$. It is *just faulty* if its last transition belongs to $\dot{T}$ and all other transitions belong to T. Note that while faulty runs can be finite or infinite, *just faulty* runs are always finite prefix of a faulty run, and end in a location $\dot{l}$.

▶ **Definition 6** (Back To Normal (BTN)). *Let $K \geq 1$, $\mathcal{A}$ be a timed automaton with fault model $\mathcal{P}$. Let $\rho = (l_0, \nu_0) \xrightarrow{(t_1,d_1)} (l_1, \nu_1) \xrightarrow{(t_2,d_2)} \cdots$ be a (finite or infinite) faulty accepting run of $\mathcal{A}_\mathcal{P}$, with associated timed word $(a_1, d_1)(a_2, d_2)\ldots$ and let $i \in \mathbb{N}$ be the position of the faulty transition in ρ. Then ρ is* back to normal (BTN) *after K steps if there exists an accepting run $\rho' = (l'_0, \nu'_0) \xrightarrow{(t'_1,d'_1)} (l'_1, \nu'_1) \xrightarrow{(t'_2,d'_2)} \cdots$ of $\mathcal{A}$ with associated timed word $(a'_1, d'_1)(a'_2, d'_2)\ldots$ and an index $\ell \in \mathbb{N}$ such that $(a'_\ell, d'_\ell)(a'_{\ell+1}, d'_{\ell+1}) \cdots = (a_{i+K}, d_{i+K})(a_{i+K+1}, d_{i+K+1}) \ldots$.*
ρ is untimed back to normal (untimed BTN) *after K steps if there exists an accepting run $\rho' = (l'_0, \nu'_0) \xrightarrow{(t'_1,d'_1)} (l'_1, \nu'_1) \xrightarrow{(t'_2,d'_2)} \cdots$ of $\mathcal{A}$ and an index $\ell \in \mathbb{N}$ s.t. $a'_\ell a'_{\ell+1} \cdots = a_{i+K} a_{i+K+1} \cdots$*

In other words, if w is a timed word having a faulty accepting run (i.e., $w \in \mathcal{L}(\mathcal{A}_\mathcal{P})$), the suffix of w, K steps after the fault, matches with the suffix of some word $w' \in \mathcal{L}(\mathcal{A})$. Note that the accepting run of w' in $\mathcal{A}$ is not faulty, by definition. The conditions in untimed BTN are simpler, and ask the same sequence of actions, but not equality on dates. Words w and w' need not have an identical prefix: this means that a BTN run has returned to *some* normal behavior, but not necessarily *the* behavior originally planned before the fault.

Our current definition of back-to-normal in K steps means that a system recovered from a fault (a primary delay) in $\leq K$ steps and remained error-free. We can generalize our definition, to model real life situations where more than one fault happens due to time delays, but the system recovers from each one in a small number of steps and eventually achieves its fixed goal (a reachability objective, some ω-regular property...). A classical example of this is a metro network, where trains are often delayed, but nevertheless recover from these delays to reach their destination on time. This motivates the following definition of resilience.

▶ **Definition 7** (Resilience). *A timed automaton $\mathcal{A}$ is*

- (untimed) *K-$\forall$-resilient if every finite faulty accepting run is (untimed) BTN in K steps.*
- (untimed) *K-$\exists$-resilient if every just faulty run ρ_{jf} can be extended into a maximal accepting run ρ_f which is (untimed) BTN in K steps.*

Intuitively, a faulty run of $\mathcal{A}$ is BTN if the system has definitively recovered from a fault, i.e., it has recovered and will follow the behavior of the original system after its recovery. The definition of existential resilience considers maximal (infinite, or finite but ending at a location with no outgoing transitions) runs to avoid situations where an accepting faulty run ρ_f is BTN, but all its extensions i.e., suffixes ρ' are such that $\rho_f.\rho'$ *is not* BTN.

▶ **Example 8.** We model train services to a specific destination such as an airport. On an average, the distance between two consecutive stations is covered in ≤ 4 time units. At each stop in a station, the dwell time is in between 1 and 2 time units. To recover from a delay, the train is allowed to skip an intermediate station (as long as the next stop is not the destination). Skipping a station is a choice, and can only be activated if there is a delay. We

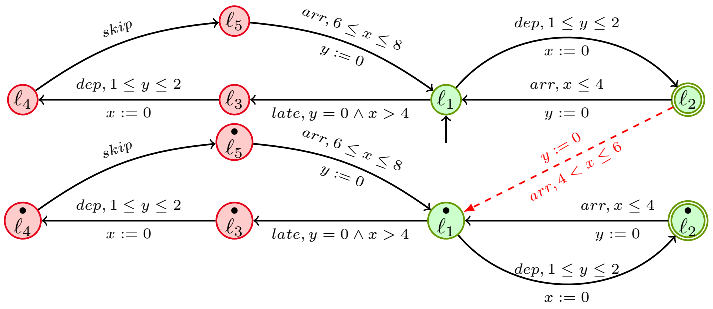

Figure 2 Enlarged automaton for the train system (with recovery) model of Figure 1

model this system with the timed automaton of Figure 1. There are 5 locations: ℓ_1, and ℓ_2 represent the normal behavior of the train and ℓ_3, ℓ_4, ℓ_5 represent the skipping mechanism. These locations can only be accessed if the faulty transition (represented as a red dotted arrow in Figure 1) is fired. A transition t_{ij} goes from ℓ_i to ℓ_j, and $\dot{t}_{21}$ denotes the faulty transition from ℓ_2 to $\dot{\ell}_1$. The green locations represent the behavior of the train without any delay, and the red locations represent behaviors when the train chooses to skip the next station to recover from a delay. This mechanism is invoked once the train leaves the station where it arrived late (location ℓ_3). When it departs, x is reset as usual; the next arrival to a station (from location ℓ_4) happens after skipping stop at the next station. The delay can be recovered since the running time since the last stop (covering 2 stations) is between 6 and 8 units of time. Formally, verifying that this system can recover from a delay within K steps can be done by setting as fault model $\mathcal{P}(arr) = 2$, and then checking a K-$\exists$-*resilience* problem. It then amounts to asking if the enlarged automaton of Figure 2 can recognize a suffix of a word recognized by the automaton of Figure 1, K steps after visiting location $\dot{\ell}_1$.

Consider the faulty run $\rho_f = (\ell_1, 0|0) \xrightarrow{(t_{12},2)} (\ell_2, 0|2) \xrightarrow{(\dot{t}_{21},8)} (\ell_1, 6|0) \xrightarrow{(t_{13},8)} (\dot{\ell}_3, 6|0) \xrightarrow{(t_{34},10)} (\dot{\ell}_4, 0|2) \xrightarrow{(t_{45},10)} (\dot{\ell}_5, 0|2) \xrightarrow{(t_{51},18)} (\dot{\ell}_1, 8|0) \xrightarrow{(t_{12},19)} (\dot{\ell}_2, 0|1)$ reading $(dep, 2)(arr, 8)(late, 8)(dep, 10)$ $(skip, 10)(arr, 18)(dep, 19)$. Run ρ_f is BTN in 4 steps. It matches the non-faulty run $\rho = (\ell_1, 0|0) \xrightarrow{(t_{12},2)} (\ell_2, 0|2) \xrightarrow{(t_{21},6)} (\ell_1, 4|0) \xrightarrow{(t_{12},8)} (\ell_2, 0|2) \xrightarrow{(t_{21},12)} (\ell_1, 4|0) \xrightarrow{(t_{12},14)} (\ell_2, 0|2) \xrightarrow{(t_{21},18)} (\ell_1, 4|0) \xrightarrow{(t_{12},19)} (\ell_2, 0|1)$ reading $(dep, 2)(arr, 6)(dep, 8)(arr, 12)(dep, 14)(arr, 18)(dep, 19)$. This automaton is K-$\exists$-*resilient* for $K = 4$ and fault model $\mathcal{P}$, as skipping a station after a delay of $\leq$2 time units allows to recover the time lost. It is not K-$\forall$-*resilient*, for any K, as skipping is not mandatory, and a train can be late for an arbitrary number of steps. In Appendix A we give another example that is 1-$\forall$-*resilient*.

K-$\forall$-*resilience* always implies K-$\exists$-*resilience*. In case of K-$\forall$-*resilience*, every faulty run ρ_w has to be BTN in $\leq K$ steps after the occurrence of a fault. This implies K-$\exists$-*resilience* since, any just faulty run ρ_w that is the prefix of an accepting run ρ of $\mathcal{A}_{\mathcal{P}}$ is BTN in less than K steps. The converse does not hold: $\mathcal{A}_{\mathcal{P}}$ can have a pair of runs ρ_1, ρ_2, sharing a common just faulty run ρ_f as prefix such that ρ_1 is BTN in K steps, witnessing existential resilience, while ρ_2 is not. Finally, an accepting run $\rho = \rho_f \rho_s$ in $\mathcal{A}_{\mathcal{P}}$ s.t., ρ_f is *just faulty* and $|\rho_s| < K$, is BTN in K steps since ε is a suffix of a run accepted by $\mathcal{A}$.

4 Existential Resilience

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

Existential Timed Resilience. As the first step, we define a product automaton $\mathcal{B} \otimes_K \mathcal{A}$

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

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

- *$n \neq 0$ (no fault has occurred, or less than K steps of $\mathcal{B}$ have occurred), the action is $< x, y >=< a, * >$, we have transition $t_B = (l_B, g, a, R, l'_B) \in T_B$, $l_A = l'_A$ (the location of $\mathcal{A}$ is unchanged) and either: $n = -1$, the transition t_B is faulty and $n' = K$, or $n = -1$, the transition t_B is non faulty and $n' = -1$, or $n > 0$ and $n' = n - 1$.*
- *$n = n' \neq 0$ (no fault has occurred, or less than K steps of $\mathcal{B}$ have occurred), the action is $< x, y >=< *, a >$, we have the transition $t_A = (l_A, g, a, R, l'_A) \in T_A$, $l_B = l'_B$ (the location of $\mathcal{B}$ is unchanged).*
- *$n = n' = 0$ (at least K steps after a fault have occured), the action is $< x, y >=< a, a >$ and there exists two transitions $t_B = (l_B, g, a, R_B, l'_B) \in T_B$ and $t_A = (l_A, g_A, a, R_A, l'_A) \in T_A$ with $g = g_A \wedge g_B$, and $R = R_B \cup R_A$ (t_A and t_B occur synchronously).*

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

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

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

▶ **Proposition 10.** *Let ρ_f be a faulty accepting run of $\mathcal{A}_{\mathcal{P}}$. The following are equivalent:*

i *ρ_f is BTN in K-steps*

ii *there is an accepting run $\rho^{\otimes}$ of $\mathcal{A}_{\mathcal{P}} \otimes_K \mathcal{A}$ s.t., the projection on its first component is ρ_f*

Let ρ be a finite run of $\mathcal{A}_{\mathcal{P}}$. We denote by $T_{\rho}^{\otimes_K}$ the set of configurations of $\mathcal{A}_{\mathcal{P}} \otimes_K \mathcal{A}$ such that there exists a run $\rho^{\otimes}$ of $\mathcal{A}_{\mathcal{P}} \otimes_K \mathcal{A}$ ending in this configuration, whose projection on the first component is ρ. We then define $S_{\rho}^{\otimes_K}$ as the set of states of $\mathcal{R}(\mathcal{A}_{\mathcal{P}} \otimes_K \mathcal{A})$

corresponding to $T_\rho^{\otimes_K}$, i.e., $S_\rho^{\otimes_K} = \{(s,[\nu]) \in \mathcal{R}(\mathcal{A}_\mathcal{P} \otimes_K \mathcal{A}) \mid (s,\nu) \in T_\rho^{\otimes_K}\}$. If we can compute the set $\mathbb{S} = \{S_\rho^{\otimes_k} \mid \rho \text{ is a finite run of } \mathcal{A}_\mathcal{P}\}$, we would be able to solve *timed* universal resilience, because from this set, one can check existence of a run accepted by $\mathcal{A}_\mathcal{P}$ and not by $\mathcal{A}$. Proposition 18 shows that universal resilience is undecidable. Hence, computing $\mathbb{S}$ is impossible. Roughly speaking, it is because this set depends on the exact timing in a run ρ, and in general one cannot use the region construction.

We can however show that in some restricted cases, we can use a *modified* region construction to build $S_\rho^{\otimes_K}$, which will enable decidability of timed existential resilience. First, we restrict to *just faulty runs*, i.e., consider runs of $\mathcal{A}_\mathcal{P}$ and $\mathcal{A}$ of equal durations, but that did not yet synchronize on actions in the product $\mathcal{A}_\mathcal{P} \otimes_K \mathcal{A}$. For a timed run ρ, by its duration, we mean the time-stamp or date of occurrence of its last event. Second, we consider abstract runs $\widetilde{\sigma}$ through a so-called *strong region automaton*, as defined below. Intuitively, $\widetilde{\sigma}$ keeps more information than in the usual region automaton to ensure that for two timed runs $\rho_1 = (t_1,d_1)(t_2,d_2)\dots$, and $\rho_2 = (t_1,e_1)(t_2,e_2)\dots$ associated with the same run of the strong region automaton, we have $\lfloor e_i \rfloor = \lfloor d_i \rfloor$ for all i. Formally, we build the strong region automaton $\mathcal{R}_{\text{strong}}(\mathcal{B})$ of a timed automaton $\mathcal{B}$ as follows. We add a virtual clock x_ι to $\mathcal{B}$ which is reset at each integral time point, add constraint $x_\iota < 1$ to each transition guard, and add a virtual self loop transition with guard $x_\iota = 1$ resetting x_ι on each state. Standard regions are equivalence classes for clock values, but not for elapsed time. Adding a virtual clock resetting at every integral time point allows to consider the fractional part of elapsed global time in regions. Lemma 12 below shows that if two abstract runs σ_1, σ_2 visit the same sequence of strong regions, then there are two runs of identical duration that have σ_1, σ_2 as abstractions. We then make the usual region construction on this extended timed automaton to obtain $\mathcal{R}_{\text{strong}}(\mathcal{B})$. The strong region construction thus has the same complexity as the standard region construction. Let $\mathcal{L}(\mathcal{R}_{\text{strong}}(\mathcal{B}))$ be the language of this strong region automaton, where these self loops on the virtual clock are projected away. These additional transitions capture ticks at integral times, but do not change the behavior of $\mathcal{B}$, i.e., we have $Unt(\mathcal{L}(\mathcal{B})) \subseteq \mathcal{L}(\mathcal{R}_{\text{strong}}(\mathcal{B})) \subseteq \mathcal{L}(\mathcal{R}(\mathcal{B})) = Unt(\mathcal{L}(\mathcal{B}))$ so $Unt(\mathcal{L}(\mathcal{B})) = \mathcal{L}(\mathcal{R}_{\text{strong}}(\mathcal{B}))$.

For a finite abstract run $\widetilde{\sigma}$ of the *strong* region automaton $\mathcal{R}_{\text{strong}}(\mathcal{A}_\mathcal{P})$, we define the set $S_{\widetilde{\sigma}}^{\otimes_K}$ of states of $\mathcal{R}_{\text{strong}}(\mathcal{A}_\mathcal{P} \otimes_K \mathcal{A})$ (the virtual clock is projected away, and our region is w.r.t original clocks) such that there exists a run $\widetilde{\sigma}^\otimes$ through $\mathcal{R}_{\text{strong}}(\mathcal{A}_\mathcal{P} \otimes_K \mathcal{A})$ ending in this state and whose projection on the first component is $\widetilde{\sigma}$. Let $\widetilde{\sigma}_\rho$ be the run of $\mathcal{R}_{\text{strong}}(\mathcal{A}_\mathcal{P})$ associated with a run ρ of $\mathcal{A}_\mathcal{P}$. It is easy to see that $S_{\widetilde{\sigma}}^{\otimes_K} = \bigcup_{\rho \mid \widetilde{\sigma}_\rho = \widetilde{\sigma}} S_\rho^{\otimes_K}$. For a *just faulty* timed run ρ of $\mathcal{A}_\mathcal{P}$, we have a stronger relation between $S_\rho^{\otimes_K}$ and $S_{\widetilde{\sigma}_\rho}^{\otimes_K}$:

▶ **Proposition 11.** *Let ρ be a* just faulty run *of $\mathcal{A}_\mathcal{P}$. Then $S_\rho^{\otimes_K} = S_{\widetilde{\sigma}_\rho}^{\otimes_K}$.*

Proof. First, notice that given a just faulty timed run ρ of $\mathcal{A}_\mathcal{P}$ and a timed run ρ' of $\mathcal{A}$ of same duration, the timed run $\rho \otimes \rho'$ (the run of $\mathcal{A}_\mathcal{P} \otimes_K \mathcal{A}$ such that ρ is the projection on the first component and ρ' on the second component) exists.

To show that $S_\rho^{\otimes_K} = S_{\widetilde{\sigma}_\rho}^{\otimes_K}$, we show that for any pair of just faulty runs ρ_1, ρ_2 of $\mathcal{A}_\mathcal{P}$ with $\widetilde{\sigma}_{\rho_1} = \widetilde{\sigma}_{\rho_2}$, we have $S_{\rho_1}^{\otimes_K} = S_{\rho_2}^{\otimes_K}$, which yields the result as $S_{\widetilde{\sigma}_\rho}^{\otimes_K} = \bigcup_{\rho' \mid \widetilde{\sigma}_{\rho'} = \widetilde{\sigma}_\rho} S_{\rho'}^{\otimes_K}$. Consider ρ_1, ρ_2, two just faulty timed runs of $\mathcal{A}_\mathcal{P}$ with $\widetilde{\sigma}_{\rho_1} = \widetilde{\sigma}_{\rho_2}$ and let $(l_{\mathcal{A}_\mathcal{P}}, l_\mathcal{A}, K, r) \in S_{\rho_1}^{\otimes_K}$. Then, this implies that there exists $\nu_1 \models r$ and a timed run ρ_1' of $\mathcal{A}$ with the same duration as ρ_1, such that $\rho_1 \otimes \rho_1'$ ends in state $(l_{\mathcal{A}_\mathcal{P}}, l_\mathcal{A}, K, \nu_1)$. The following lemma completes the proof:

▶ **Lemma 12.** *There exists $\nu_2 \models r$ and a timed run ρ_2' of $\mathcal{A}$ with the same duration as ρ_2, such that $\rho_2 \otimes \rho_2'$ ends in state $(l_{\mathcal{A}_\mathcal{P}}, l_\mathcal{A}, K, \nu_2)$.*

The main idea of the proof is to show that we can construct ρ'_2 which will have the same transitions as ρ'_1, with same integral parts in timings (thanks to the information from the strong region automaton), but possibly different timings in the fractional parts, called a re-timing of ρ'_1. Notice that ρ_2 *is* a re-timing of ρ_1, as $\widetilde{\sigma}_{\rho_1} = \widetilde{\sigma}_{\rho_2}$. We translate the requirement on ρ'_2 into a set of constraints (which is actually a partial ordering) on the fractional parts of the dates of its transitions, and show that we can indeed set the dates accordingly. This translation follows the following idea: the value of a clock x just before firing transition t is obtained by considering the date d of t minus the date d^x of the latest transition t^x at which x has been last reset before t. In particular, the difference $x - y$ between clocks x, y just before firing transition t is $(d - d^x) - (d - d^y) = d^y - d^x$. That is, the value of a clock or its difference can be obtained by considering the difference between two dates of transitions. A constraint given by $x - y \in (n, n+1)$ is equivalent with the constraint given by $d^y - d^x \in (n, n+1)$, and similar constraints on the fractional parts can be given.

Proof. Let $t_1, \ldots, t_n$ be the sequence of transitions of ρ_1, ρ_2 taken respectively, at dates $d_1, \ldots, d_n$ and $e_1, \ldots, e_n$. Similarly, we will denote by $t'_1, \ldots, t'_k$ the sequence of transitions of ρ'_1, taken at dates $d'_1, \ldots, d'_k$. Run ρ'_2 will pass by the same transitions $t'_1, \ldots, t'_k$, but with possibly different dates $e'_1, \ldots, e'_k$ such that:

- the duration of ρ'_2 is the same as the duration of ρ_2,
- $\widetilde{\sigma}_{\rho'_2}$ follows the same sequence of states of $\mathcal{R}_{\text{strong}}(\mathcal{A})$ as $\widetilde{\sigma}_{\rho'_1}$ (in particular, ρ'_2 is a valid run as it fullfils the guards of its transitions, which are the same as those of ρ'_1).
- $\widetilde{\sigma}_{\rho_2 \otimes \rho'_2}$ reaches the same state of $\mathcal{R}_{\text{strong}}(\mathcal{A}_{\mathcal{P}} \otimes_K \mathcal{A})$ as $\widetilde{\sigma}_{\rho_1 \otimes \rho'_1}$.

We translate these into three requirements on the dates $(e'_i)_{i \leq k}$ of ρ'_2:

R1. We have $e'_k = e_n$,

R2. For every $i \leq k$, the integral part $\lfloor e'_i \rfloor = \lfloor d'_i \rfloor$. Remark that we already have $\lfloor e'_k \rfloor = \lfloor e_n \rfloor = \lfloor d_n \rfloor = \lfloor d'_k \rfloor$ by $R1$ and by the hypothesis,

R3. Fractional parts $(\mathsf{frac}(e'_i))_{i \leq k}$ satisfy a set of constraints, defined hereafter as a partial ordering on $(\mathsf{frac}(e'_i))_{i \leq k} \cup (\mathsf{frac}(e_i))_{i \leq n}$.

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

We then characterize the conditions required for the run $\rho_2 \otimes \rho'_2$ to reach the same region r of $\mathcal{R}_{\text{strong}}(\mathcal{A}_{\mathcal{P}} \otimes_K \mathcal{A})$ which was reached by $\rho_1 \otimes \rho'_1$. These conditions are described as on region r in the following equivalent ways:

1. A set of constraints C on the disjoint union $X'' = X_{\mathcal{A}_{\mathcal{P}}} \uplus X_{\mathcal{A}}$ of clocks of $\mathcal{A}_{\mathcal{P}}$ and $\mathcal{A}$, of the form $x - y \in (n, n+1)$ or $x - y = n$ or $x - y > Max$ (possibly considering a null clock y) for $n \in \mathbb{Z}$,
2. The associated set of constraints $C' = \{d(c) \mid c \in C\}$ on $D = \{d_x \mid x \in X_{\mathcal{A}_{\mathcal{P}}}\} \uplus \{d'_{x'} \mid x' \in X_{\mathcal{A}}\}$, with d_x the date of the latest transition $t_j^{\otimes}$ that resets the clock $x \in X_{\mathcal{A}_{\mathcal{P}}}$, and $d'_{x'}$ the date of the latest transition $t_l^{\otimes}$ that resets clock $x' \in X_{\mathcal{A}}$,
3. An ordering $\leq'$ over $FP = \{\mathsf{frac}(\tau) \mid \tau \in D\}$ defined as follows: for each constraint $\tau - \tau' \in (n, n+1)$ of C', if $\lfloor \tau \rfloor = \lfloor \tau' \rfloor + n$ then $\mathsf{frac}(\tau) <' \mathsf{frac}(\tau')$, and if $\lfloor \tau \rfloor = \lfloor \tau' \rfloor + n + 1$ then $\mathsf{frac}(\tau') <' \mathsf{frac}(\tau)$.

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

For each constraint $\tau - \tau' > \mathsf{c}_{\max}$ of C' such that $\lfloor \tau \rfloor = \lfloor \tau' \rfloor + \mathsf{c}_{\max}$, we have $\mathsf{frac}(\tau') >' \mathsf{frac}(\tau)$ (if $\lfloor \tau \rfloor \geq \lfloor \tau' \rfloor + \mathsf{c}_{\max} + 1$, then we dont need to do anything), where $\mathsf{c}_{\max} = \max(\{c_x \mid x \in X\})$.

Further, path ρ'_2 needs to visit the regions $r_1, \dots r_k$ visited by ρ'_1. For each i, visiting region r_i is characterized by a set of constraints C_i, which we translate as above as an ordering $\leq'_i$ on $FP' = \{\mathsf{frac}(d'_i) \mid i \leq k\}$.

Thus, finally, we can collect all the requirements for having ρ' with required properties by defining $\leq''$ over $FP' \cup FP$ (notice that it is not a disjoint union) as the transitive closure of the union of all $\leq'_i$ and of $\leq'$. As the union of constraints on C'_i and on C' is satisfied by the dates $(d_i)_{i \leq n}$ and $(d'_i)_{i \leq k}$ of ρ_1 and ρ'_1, the union of constraints is satisfiable. Equivalently, $\leq''$ is a partial ordering, respecting the total natural ordering $\leq$ on $FP \cup FP'$. We will denote $\tau ='' \tau'$ whenever $\tau \leq'' \tau'$ and $\tau' \leq'' \tau$, and $\tau <'' \tau'$ if $\tau \leq'' \tau'$ but we dont have $\tau ='' \tau'$. Because $\leq''$ is a partial ordering, there is no τ, τ' with $\tau <'' \tau' <'' \tau$.

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

We will start from the fixed value of $\mathsf{frac}(e'_{k-1})$ and work backwards. Let us assume inductively that $\mathsf{frac}(e'_{k-1}) \dots \mathsf{frac}(e'_{i+1})$ have been fixed. We now describe how to obtain $\mathsf{frac}(e'_i)$. If $\mathsf{frac}(d'_i) ='' \mathsf{frac}(d'_j)$, $j > i$ then we set $\mathsf{frac}(e'_i) = \mathsf{frac}(e'_j)$. If $\mathsf{frac}(d'_i) ='' \mathsf{frac}(d_j)$, then we set $\mathsf{frac}(e'_i) = \mathsf{frac}(e_j)$. Otherwise, consider the sets $L_i = \{\mathsf{frac}(e_j) \mid j \leq n, \mathsf{frac}(d_j) <'' \mathsf{frac}(d'_i)\} \cup \{\mathsf{frac}(e'_j) \mid i < j \leq n, \mathsf{frac}(d'_j) <'' \mathsf{frac}(d'_i)\}$. Also, consider $U_i = \{\mathsf{frac}(e_j) \mid j \leq n, \mathsf{frac}(d_j) >'' \mathsf{frac}(d'_i)\} \cup \{\mathsf{frac}(e'_j) \mid i < j \leq n, \mathsf{frac}(d'_j) >'' \mathsf{frac}(d'_i)\}$. We let $l_i = max(L_i)$ and $u_i = min(U_i)$. We then set $\mathsf{frac}(e'_i)$ to any value in (l_i, u_i). It remains to show that we always have $l_i < u_i$, which will show that such a choice of value for the fractional part of e'_i is indeed possible.

By contradiction, consider that there exists i such that $l_i \geq u_i$, and consider the maximal (first) such i. First, assume that both l_i and u_i are of the form $\mathsf{frac}(e_j), \mathsf{frac}(e_k)$ respectively, i.e. corresponds to clock values in the last regions of ρ_2. The contradiction hypothesis is $l_i = \mathsf{frac}(e_j) \geq u_i = \mathsf{frac}(e_k)$. By definition of L_i and U_i, we also have $\mathsf{frac}(d_j) <'' \mathsf{frac}(d'_i) <'' \mathsf{frac}(d_k)$. In particular, $\mathsf{frac}(d_j) < \mathsf{frac}(d_k)$. This is a contradiction with $\widetilde{\sigma}_{\rho_1} = \widetilde{\sigma}_{\rho_2}$, as the strong region reached by ρ_1 and ρ_2 are the same. A contradiction.

Otherwise, at least one of l_i, u_i is of the form $\mathsf{frac}(e'_j)$, with $j > i$ (consider j minimal if both are of this form). By symetry, let say $l_i = \mathsf{frac}(e'_j) \geq u_i$. Let say $u_i = \mathsf{frac}(e_k)$, as $u_i = \mathsf{frac}(e'_k)$ with $k > j$ is similar since it has been fixed before $\mathsf{frac}(e'_j)$. We have $\mathsf{frac}(d'_j) <'' d'_i <'' \mathsf{frac}(d_k)$ by definition of L_i, U_i. In particular $\mathsf{frac}(d'_j) <'' \mathsf{frac}(d_k)$: That is, $k \in U_j$, and by construction, and as $j > i$, we have $l_i = \mathsf{frac}(e'_j) < \mathsf{frac}(e_k) = u_i$, a contradiction. ◀

Lemma 12 completes the proof of Proposition 11 immediately. Indeed, the lemma implies that $(l_{\mathcal{A}_\mathcal{P}}, l_A, K, r) \in S^{\otimes_K}_{\rho_2}$ from which we infer that $S^{\otimes_K}_{\rho_1} \subseteq S^{\otimes_K}_{\rho_2}$. By a symmetric argument we get the other containment also, and hence we conclude that $S^{\otimes_K}_{\rho_1} = S^{\otimes_K}_{\rho_2}$. ◀

Lemma 12, which is crucial for our decidability results for existential timed resilience, shows that a timed run can be re-timed, i.e., it shows the existence of a timed run with the same transitions but possibly different timestamps. For this, the global time-stamps (d_j) of actions need to be fixed, and in particular the ordering between their fractional parts $\mathsf{frac}(d_j)$. The normal region automaton only ensures ordering between the differences of

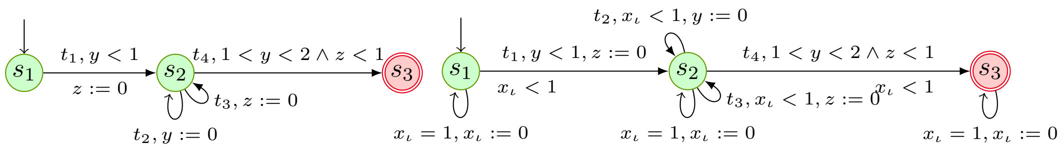
s_1
$t_1, y < 1$
$z := 0$
s_2
$t_4, 1 < y < 2 \wedge z < 1$
s_3
$t_3, z := 0$
$t_2, y := 0$
$t_2, x_\iota < 1, y := 0$
s_1
$t_1, y < 1, z := 0$
$x_\iota < 1$
s_2
$t_4, 1 < y < 2 \wedge z < 1$
$x_\iota < 1$
s_3
$t_3, x_\iota < 1, z := 0$
$x_\iota = 1, x_\iota := 0$
$x_\iota = 1, x_\iota := 0$
$x_\iota = 1, x_\iota := 0$

Figure 3 Example timed automaton (left) and its strong timed automaton (right)

(d_j)'s, but not (d_j) themselves. Let us illustrate this with an concrete example of a TA c.f., Figure 3 (left), having 3 locations s_1, s_2, s_3, 2 clocks y, z and transitions $t_1 = (y < 1, z := 0), t_2 = (y := 0), t_3 = (z := 0), t_4 = (1 < y < 2, z < 1)$ such that t_1 goes from location s_1 to s_2, t_2, t_3 are loops at s_2 and t_4 goes from s_2 to s_3. We can see the run in the standard region automaton $\sigma = (s_1, [\{0\}, \{0\}]) \xrightarrow{t_1} (s_2, [(0,1), \{0\}]) \xrightarrow{t_2} (s_2, [\{0\}, (0,1)]) \xrightarrow{t_3} (s_2, [(0,1), \{0\}]) \xrightarrow{t_4} (s_3, [(1,2), (0,1), \mathsf{frac}(y) < \mathsf{frac}(z)])$. The following two timed runs $\rho_1 = (t_1, d_1 = 0.8)(t_2, d_2 = 1.2)(t_3, d_3 = 1.9)(t_4, d_4 = 2.4)$ and $\rho_2 = (t_1, d'_1 = 0.9)(t_2, d'_2 = 1.89)(t_3, d'_3 = 2.69)(t_4, d'_4 = 3.39)$ correspond to abstract run σ. Note that $\mathsf{frac}(d_2) < \mathsf{frac}(d_3)$ but $\mathsf{frac}(d'_2) > \mathsf{frac}(d'_3)$.

We build the strong region automaton by adding a virtual clock x_ι reset at all integer points (reset x when $x_\iota = 1$) c.f., Figure 3 (right). As explained above, concrete runs ρ_1 and ρ_2 have the same abstract run σ in the standard region automaton. Now, if we consider abstract runs in the strong region automaton (i.e. with the addition of a clock x_ι reset at integral time points), the concrete run ρ_1 will correspond to abstract run $\sigma_1 = (s_1, [\{0\}, \{0\}, \{0\}]) \xrightarrow{t_1} (s_2, [(0,1), (0,1), \{0\}, \mathsf{frac}(x_\iota) = \mathsf{frac}(y)]) \xrightarrow{t_2} (s_2, [(0,1), \{0\}, (0,1), \mathsf{frac}(x_\iota) < \mathsf{frac}(z)]) \xrightarrow{t_3} (s_2, [(0,1), (0,1), \{0\}, \mathsf{frac}(y) < \mathsf{frac}(x_\iota)]) \xrightarrow{t_4} (s_3, [(0,1), (1,2), (0,1), \mathsf{frac}(y) < \mathsf{frac}(x_\iota) < \mathsf{frac}(z)])$, and the concrete run ρ_2 will correspond to abstract run $\sigma_2 = (s_1, [\{0\}, \{0\}, \{0\}]) \xrightarrow{t_1} (s_2, [(0,1), (0,1), \{0\}, \mathsf{frac}(x_\iota) = \mathsf{frac}(y)]) \xrightarrow{t_2} (s_2, [(0,1), \{0\}, (0,1), \mathsf{frac}(x_\iota) < \mathsf{frac}(z)]) \xrightarrow{t_3} (s_2, [(0,1), (0,1), \{0\}, \mathsf{frac}(x_\iota) < \mathsf{frac}(y)]) \xrightarrow{t_4} (s_3, [(0,1), (1,2), (0,1), \mathsf{frac}(x_\iota) < \mathsf{frac}(y) < \mathsf{frac}(z)])$. The abstract run σ_1 ends with a relation $\mathsf{frac}(y) < \mathsf{frac}(x_\iota) < \mathsf{frac}(z)$ on fractional parts of clocks x_ι, y, z, the abstract runs σ_2 end with the relation $\mathsf{frac}(x_\iota) < \mathsf{frac}(y) < \mathsf{frac}(z)$. Thus, ρ_1 and ρ_2, do not have the same abstract "strong" run.

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

▶ **Lemma 13.** *There exists a maximal accepting extension of a just faulty run ρ_f that is BTN in K-steps iff $S^{\otimes_K}_{\rho_f}$ is safe. Further, deciding if $S^{\otimes_K}_{\rho_f}$ is safe can be done in PSPACE.*

Proof. Let ρ_f a just faulty run. By Proposition 10, there exists an extention ρ of ρ_f that is BTN in K steps if and only if there exists an accepting run $\rho^{\otimes_K}$ of $\mathcal{A}_\mathcal{P} \otimes_K \mathcal{A}$ such that ρ_f is a prefix of the projection of $\rho^{\otimes_K}$ on its first component, if and only if there exists a just faulty run $\rho_f^{\otimes_K}$ of $\mathcal{A}_\mathcal{P} \otimes_K \mathcal{A}$ such that its projection on the first component is ρ_f, and such that an accepting state of $\mathcal{A}_\mathcal{P} \otimes_K \mathcal{A}$ can be reached after $\rho_f^{\otimes_K}$, if and only if $S^{\otimes_K}_{\rho_f}$ is safe.

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

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

▶ **Theorem 14.** *K-$\exists$-resilience of timed automata is in EXPSPACE.*

Proof. Lemma 13 implies that $\mathcal{A}$ is not K-timed existential resilient if and only if there exists a just faulty run ρ_f such that $S^{\otimes_K}_{\rho_f}$ is not safe. This latter condition can be checked. Let us denote by $\mathcal{R}_{\mathrm{strong}}(\mathcal{A}_\mathcal{P}) = (S_{\mathcal{R}(\mathcal{A}_\mathcal{P})}, I_{\mathcal{R}(\mathcal{A}_\mathcal{P})}, \Sigma, T_{\mathcal{R}(\mathcal{A}_\mathcal{P})}, F_{\mathcal{R}(\mathcal{A}_\mathcal{P})})$ the strong region automaton associated with $\mathcal{A}_\mathcal{P}$. We also denote $\mathcal{R}_{\otimes_K} = (S_{\mathcal{R}_{\otimes_K}}, I_{\mathcal{R}_{\otimes_K}}, \Sigma, T_{\mathcal{R}_{\otimes_K}}, F_{\mathcal{R}_{\otimes_K}})$ the strong region automaton $\mathcal{R}_{\mathrm{strong}}(\mathcal{A}_\mathcal{P} \otimes_K \mathcal{A})$. Let ρ_f be a just faulty run, and let $\sigma = \widetilde{\sigma}_{\rho_f}$ denote the run of $\mathcal{R}_{\mathrm{strong}}(\mathcal{A}_\mathcal{P})$ associated with ρ_f. Thanks to Proposition 11, we have $S^{\otimes_K}_{\rho_f} = S^{\otimes_K}_{\sigma}$, as $S^{\otimes_K}_{\rho_f}$ does not depend on the exact dates in ρ_f, but only on their regions, i.e., on σ. So it suffices to find a reachable witness $S^{\otimes_K}_{\sigma}$ of $\mathcal{R}_{\otimes_K}$ which is not safe, to conclude that $\mathcal{A}$ is not existentially resilient. For that, we build an (untimed) automaton $\mathfrak{B}$. Intuitively, this automaton follows σ up to a fault of the region automaton $\mathcal{R}_{\mathrm{strong}}(\mathcal{A}_\mathcal{P})$, and maintains the set $S^{\otimes_K}_{\sigma}$ of regions of $\mathcal{R}_{\otimes_K}$. This automaton stops in an accepting state immediately after occurrence of a fault. Formally, the product subset automaton $\mathfrak{B}$ is a tuple $(S_\mathfrak{B}, I, \Sigma, T, F)$ with set of states $S_\mathfrak{B} = S_{\mathcal{R}_{\mathrm{strong}}(\mathcal{A}_\mathcal{P})} \times 2^{S_{\mathcal{R}_{\otimes_K}}} \times \{0,1\}$, set of initial states $I = I_{\mathcal{R}_{\mathrm{strong}}(\mathcal{A}_\mathcal{P})} \times \{I_{\mathcal{R}_{\otimes_K}}\} \times \{0\}$, and set of final states $F = S_{\mathcal{R}_{\mathrm{strong}}(\mathcal{A}_\mathcal{P})} \times 2^{S_{\mathcal{R}_{\otimes_K}}} \times \{1\}$. The set of transitions $T \subseteq S_\mathfrak{B} \times \Sigma \times S_\mathfrak{B}$ is defined as follows,

- $\big((l,r,S,0), a, (l',r',S',\flat)\big) \in T$ if and only if $t_R = \big((l,r), a, (l',r')\big) \in T_{\mathcal{R}_{\mathrm{strong}}(\mathcal{A}_\mathcal{P})}$ and $\flat = 1$ if and only if t_R is faulty and $\flat = 0$ otherwise.
- S' is the set of states s' of $\mathcal{R}_{\mathrm{strong}}(\mathcal{A}_\mathcal{P} \otimes_K \mathcal{A})$ whose first component is (l',r') and such that there exists $s \in S, (s,a,s') \in T_{\mathcal{R}(\otimes_K)}$.

Intuitively, 0 in the states means no fault has occurred yet, and 1 means that a fault has just occurred, and thus no transition exists from this state. We have that for every prefix σ of a just faulty abstract run of $\mathcal{R}_{\mathrm{strong}}(\mathcal{A}_\mathcal{P})$, ending on a state (l,r) of $\mathcal{R}_{\mathrm{strong}}(\mathcal{A}_\mathcal{P})$ then, there exists a unique accepting path $\sigma^\otimes$ in $\mathfrak{B}$ such that σ is the projection of $\sigma^\otimes$ on its first component. Let $(l,r,S,1)$ be the state reached by $\sigma^\otimes$. Then $S^{\otimes_K}_{\sigma} = S$. Thus, non-existential resilience can be decided by checking reachability of a state $(l,r,S,1)$ such that S is not safe in automaton $\mathfrak{B}$. Recall (from Lemma 13) that checking safety of S is in PSPACE. As $\mathfrak{B}$ is of doubly exponential size, reachability can be checked in EXPSPACE. As EXPSPACE is closed under complement, checking existential resilience is in EXPSPACE. ◀

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

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

Proof. We proceed by reduction from the language emptiness problem, which is known to be PSPACE-Complete for timed automata. We can reuse the gadget $\mathcal{G}_{und}$ of Figure 4. We take any automaton $\mathcal{A}$ and collapse its initial state to state s_1 in the gadget. We recall that s_1 is accessible at date 15 only after a fault. We add a self loop with transition, $t_e = (s_2, \sigma, true, \emptyset, s_2)$ 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 2-$\exists$-*resilience*. ◀

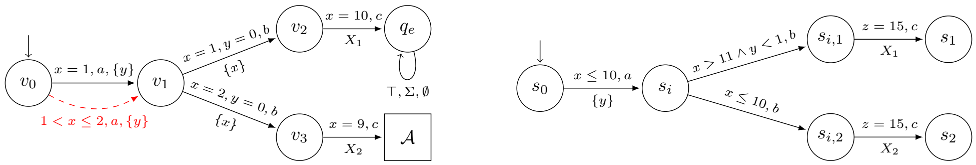

Figure 4 The gadget automaton $\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}$ (left) and the gadget $\mathcal{G}_{und}$ (right)

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

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

Proof (sketch). *Membership :* $\mathcal{A}$ is untimed K-$\exists$-*resilient* if and only if for all states $q = (l, r)$ reached by a just faulty run of $\mathcal{R}(\mathcal{A}_{\mathcal{P}})$, there exists a maximal accepting path σ from q such that its suffix σ_s after K steps is also the suffix of a path of $\mathcal{R}(\mathcal{A})$. This property can be verified in PSPACE. A detailed proof is provided in Appendix B.
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 in Figure 4 (left). 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 in dotted red) then it resets all clocks of $\mathcal{A}$ and behaves as $\mathcal{A}$. The accepting states are $q_e \cup F$. Then, $\mathcal{A}$ has an accepting word if and only if $\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}$ is untimed 2-$\exists$-*resilient*. Since the emptiness problem for timed automata is PSPACE-Complete, the result follows. ◀

▶ Remark 17. The hardness reduction in the proof of Theorem 16 holds even for deterministic timed automata. It is known [2] that PSPACE-Hardness of emptiness still holds for deterministic TAs. Hence, considering deterministic timed automata will not improve the complexity of K-$\exists$-*resilience*. Considering IRTAs does not change complexity either, as the gadget used in Theorem 16 can be adapted to become an IRTA (as shown in Appendix C).

5 Universal Resilience

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

▶ **Proposition 18.** *Language inclusion for timed automata can be reduced in polynomial time to K-$\forall$-resilience. Thus, K-$\forall$-resilience is undecidable in general for timed automata.*

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 with only one initial state (w.l.o.g). We build a timed automaton $\mathcal{B}$ such that $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ if and only if $\mathcal{B}$ is 2-$\forall$-*resilient*.

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(right). $\mathcal{G}_{und}$ 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 $\mathcal{A}_1$ i.e., l_{0_1} with s_1 and the initial location of $\mathcal{A}_2$ i.e., l_{0_2} with s_2. We also use a fault model $\mathcal{P} : a \rightarrow [0, 2]$, 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:

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

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:

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

Now, $\mathcal{B}$ is K-$\forall$-*resilient* for $K = 2$ if and only if every word in $\mathcal{L}^F(\mathcal{B}_{\mathcal{P}})$ is BTN after 2 steps ($K = 2$), i.e., for every word $w = (a, d_1)(b, d_2)(c, 15)(\sigma_1, d_3) \dots (\sigma_n, d_{n+2})$ in $\mathcal{L}^F(\mathcal{B}_{\mathcal{P}})$, 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 $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ if and only if $\mathcal{B}$ is 2-$\forall$-*resilient*.

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

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

▶ **Proposition 19.** *K-$\forall$-resilience can be reduced in polynomial time to language inclusion for timed automata with ε-transitions.*

Proof. Given a timed automaton $\mathcal{A} = (L, I, X, \Sigma, T, F)$, we can build a timed automaton $\mathcal{A}^S$ that recognizes all suffixes of timed words recognized by $\mathcal{A}$ (see Appendix B, Figure 7 for an example). Formally, $\mathcal{A}^S$ contains the original locations and transitions of $\mathcal{A}$, a copy of all location, a copy of all transitions where letters are replaced by ε, and a transition from copies to original locations labeled by their original letters.

We have $\mathcal{A}^S = (L^S, I^S, X, \Sigma \cup \{\varepsilon\}, T^S, F)$, where $L^S = L \cup \{l' \mid l \in L\}$, $I^S = \{l' \in L_S, l \in I\}$ $T^S = T \cup \{(l'_1, g, \varepsilon, R, l'_2) \mid \exists (l_1, g, \sigma, R, l_2) \in T\} \cup \{(l'_1, g, \sigma, R, l_2) \mid \exists (l_1, g, \sigma, R, l_2) \in T\}$. Obviously, for every timed word $(a_1, d_1)(a_2, d_2) \dots (a_n, d_n)$ recognized by $\mathcal{A}$, and every index $k \in 1..n$, the words $(\varepsilon, d_1)(\varepsilon, d_k)(a_{k+1}, d_{k+1}) \dots (a_n, d_n) = (a_{k+1}, d_{k+1}) \dots (a_n, d_n)$ is recognized by $\mathcal{A}^S$.

Given a timed automaton $\mathcal{A}$ and a fault model $\mathcal{P}$, we build an automaton $\mathcal{B}^{\mathcal{P}}$ which remembers if a fault has occurred, and how many transitions have been taken since a fault (see Definition 9 in Appendix B). Then, we can build an automaton $\mathcal{B}^{\mathcal{P},\varepsilon}$ by re-labeling every transition occurring before a fault and until K steps after the fault by ε, keeping the same locations, guards and resets, and leave transitions occurring more than K steps after a fault unchanged. The relabeled transitions are transitions starting from a location (l, n) with $n \neq 0$. Accepting locations of $\mathcal{B}^{\mathcal{P},\varepsilon}$ are of the form $(l, 0)$ where l is an accepting locations of $\mathcal{A}$ occurring after a fault in $\mathcal{B}^{\mathcal{P}}$. Then, every faulty run accepted by $\mathcal{B}^{\mathcal{P},\varepsilon}$ is associated with a word 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 $\mathcal{A}$, i.e., is recognized by $\mathcal{A}^S$.

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

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

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

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

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

Proof. Recall that untimed language inclusion of timed automata is EXPSPACE-Complete [9]. The lower bound is readily obtained by using the reduction of Proposition 18.

For the upper bound, we will use the construction of automata $\mathcal{A}^S$ and $\mathcal{B}^{\mathcal{P},\varepsilon}$ built during the reduction of Proposition 19. We however need inclusion of TA with ε transitions, and thus we adapt the EXPSPACE algorithm in the presence of ε transitions:

We can consider ε transitions as transitions labeled by any letter, and build the region automata $\mathcal{A}_\sharp = \mathcal{R}(\mathcal{A}^S)$ and $\mathcal{B}_\sharp = \mathcal{R}(\mathcal{B}^{\mathcal{P},\varepsilon})$. The size of these untimed automata is exponential in the number of clocks, with ε transitions. We can perform an ε reduction on $\mathcal{A}_\sharp$ to obtain an automaton $\mathcal{A}^S_U$ with the same number of states as $\mathcal{A}_\sharp$ that recognizes untimed suffixes of words of $\mathcal{A}$. Similarly, we can perform an ε reduction on $\mathcal{B}_\sharp$ to obtain an automaton $\mathcal{B}^{\mathcal{P}}_U$ with the same number of states as $\mathcal{B}_\sharp$ that recognizes suffixes of words played K steps after a fault. We then check $\mathcal{L}(\mathcal{B}^{\mathcal{P}}_U) \subseteq \mathcal{L}(\mathcal{A}^S_U)$ with an usual PSPACE inclusion algorithm, which yields the EXPSPACE upper bound, as $\mathcal{A}^S_U, \mathcal{B}^{\mathcal{P}}_U$ have an exponential number of states w.r.t. $|\mathcal{A}|$. ◀

6 Conclusion

Resilience allows to check robustness of a timed system to unspecified delays. A universally resilient timed system recovers from any delay in some fixed number of steps. Existential resilience guarantees the existence of a controller that can bring back the system to a normal behavior within a fixed number of steps after an unexpected delay. Interestingly, we show that existential resilience enjoys better complexities/decidability than universal resilience. Universal resilience is decidable only for well behaved classes of timed automata such as IRTA, or in the untimed setting. A future work is to investigate resilience for other determinizable classes of timed automata, and a natural extension of resilience called *continuous resilience*, where a system recovers within some fixed duration rather than within some number of steps. Another natural question is to consider resilience questions when K is not fixed, i.e., check existence of a value for K such that $\mathcal{A}$ is K-$\exists$-*resilient* (resp. K-$\forall$-*resilient*).

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. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

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 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.

5 Roderick Bloem, Peter Gjøl Jensen, Bettina Könighofer, Kim Guldstrand Larsen, Florian Lorber, and Alexander Palmisano. It's time to play safe: Shield synthesis for timed systems. *CoRR*, abs/2006.16688, 2020.

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 V. Diekert, P. Gastin, and A. Petit. Removing epsilon-transitions in timed automata. In *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '97, page 583–594, Berlin, Heidelberg, 1997. Springer-Verlag.

12 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.

13 D. D'Souza, M. Gopinathan, S. Ramesh, and P. Sampath. Conflict-tolerant real-time features. In *Fifth International Conference on the Quantitative Evaluaiton of Systems (QEST 2008)*, pages 274–283. IEEE Computer Society, 2008.

14 Rüdiger Ehlers and Ufuk Topcu. Resilience to intermittent assumption violations in reactive synthesis. In Martin Fränzle and John Lygeros, editors, *17th International Conference on Hybrid Systems: Computation and Control (part of CPS Week), HSCC'14, Berlin, Germany, April 15-17, 2014*, pages 203–212. ACM, 2014.

15 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.

16 A. Puri. Dynamical properties of timed automata. *In DEDS*, 10(1-2):87–113, 2000.

17 Matthieu Renard, Yliès Falcone, Antoine Rollet, Thierry Jéron, and Hervé Marchand. Optimal enforcement of (timed) properties with uncontrollable events. *Math. Struct. Comput. Sci.*, 29(1):169–214, 2019.

18 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.

19 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.

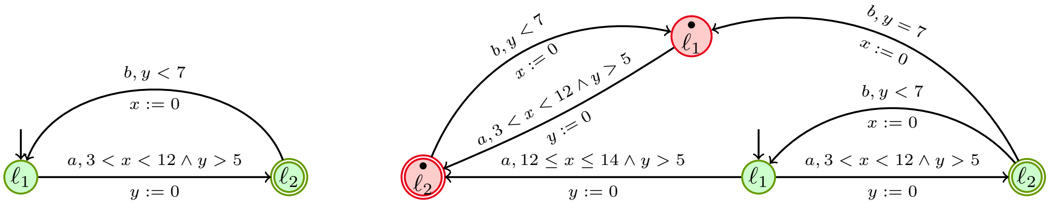

Figure 5 $\mathcal{A}$ on the left; Enlargement $\mathcal{A}_\mathcal{P}$ on the right, $\mathcal{P}(a) = 2, \mathcal{P}(b) = 0$.

A Example for Universal Resilience

▶ **Example 22.** Consider the automaton $\mathcal{A}$ in Figure 5, with two locations ℓ_1 and ℓ_2, a transition t_{12} from ℓ_1 to ℓ_2 and a transition t_{21} from ℓ_2 to ℓ_1. The enlarged automaton $\mathcal{A}_\mathcal{P}$ has two extra locations $\dot{\ell}_1, \dot{\ell}_2$, extra transitions between $\dot{\ell}_1$ and $\dot{\ell}_2$, and from ℓ_1 to $\dot{\ell}_2$ and from ℓ_2 to $\dot{\ell}_1$ respectively. We represent a configuration of the automata with a pair $\big(\ell, \nu(x)|\nu(y)\big)$ where, ℓ belongs to the set of the locations and $\nu(x)$ (resp. $\nu(y)$) represents the valuation of clock x (resp. clock y). Let $\rho_f = (\ell_1, 0|0) \xrightarrow{(t_{12},6)} (\ell_2, 6|0) \xrightarrow{(\dot{t}_{21},13)} (\dot{\ell}_1, 0|7) \xrightarrow{(\ddot{t}_{12},19)} (\dot{\ell}_2, 4|0)$ be a *faulty run* reading the faulty word $(a, 6)(b, 13)(a, 19) \in \mathcal{L}(\mathcal{A}_\mathcal{P})$. This run is 1-BTN since the run $\sigma = (\ell, 0|0) \xrightarrow{(t_{12},6)} (\ell_2, 6|0) \xrightarrow{(t_{21},12)} (\ell_1, 0|6) \xrightarrow{(t_{12},19)} (\ell_2, 7|0)$ is an accepting run of $\mathcal{A}$, reading timed word $w_\sigma = (a, 6)(b, 12)(a, 19) \in \mathcal{L}(\mathcal{A})$. Similarly, the run $\rho' = (\ell, 0|0) \xrightarrow{(\dot{t}_{12},14)} (\dot{\ell}_2, 14|0) \xrightarrow{(\ddot{t}_{21},20)} (\dot{\ell}_1, 0|6) \xrightarrow{(\ddot{t}_{12},31)} (\dot{\ell}_2, 11|0)$ of $\mathcal{A}_\mathcal{P}$ reading word $(a, 14)(b, 20)(a, 31)$ is 1-BTN because of run $\sigma' = (\ell_1, 0|0) \xrightarrow{(t_{12},10)} (\ell_2, 10|0) \xrightarrow{(t_{21},15)} (\ell_1, 0|5) \xrightarrow{(t_{12},19)} (\ell_2, 4|0) \xrightarrow{(t_{21},20)} (\ell_1, 0|1) \xrightarrow{(t_{12},31)} (\ell_2, 11|0)$ reading the word $w_{\sigma'} = (a, 10)(b, 15)(a, 19)(b, 20)(a, 31)$. One can notice that ρ' and σ' are of different lengths. In fact, we can say something stronger, namely it is 1-$\forall$-*resilient* (and hence 1-$\exists$-*resilient*) as explained below.

The example consists of a single $(a.b)^*$ loop, where action a occurs between 3 and 12 time units after entering location ℓ_1, and action b occurs less than 7 time units after entering ℓ_2. A fault occurs either from ℓ_1, in which case action a occurs $12 + d$ time units after entering ℓ_1, with $d \in [0, 2]$, or from ℓ_2, i.e., when b occurs exactly 7 time units after entering ℓ_2. Once a fault has occurred, the iteration of a and b continues on $\dot{\ell}_1$ and $\dot{\ell}_2$ with non-faulty constraints. Consider a just faulty run ρ_f where fault occurs on event a. The timed word generated in ρ_f is of the form $w_f = (a, d_1).(b, d_2) \dots (a, d_k).(b, d_{k+1}).(a, d_{k+2})$, where $d_{k+2} = d_{k+1} + 12 + x$ with $x \in [0, 2]$. The word $w = (a, d_1).(b, d_2) \dots (a, d_k).(b, d_{k+1}).(a, d_{k+1} + 5).(b, d_{k+1} + 5 + x).(a, d_{k+1} + 5 + x + 7)$ is also recognized by the normal automaton, and ends at date $d_{k+1} + 12 + x$. Hence, for every just faulty word w_f which delays action a, there exists a word w such for every timed word v, if $w_f.v$ is accepted by the faulty automaton, $w.v$ is accepted by the normal automaton. Now, consider a fault occurring when playing action b. The just faulty word ending with a fault is of the form $w_f = (a, d_1).(b, d_2) \dots (a, d_k).(b, d_k + 7)$. All occurrences of a occur at a date between $d_j + 3$ and $d_j + 12$ for some date d_j at which location ℓ_1 is reached, (except the first time stamp $d_1 \in (5, 12)$) and all occurrences of b at a date strictly smaller than $d_i + 7$, where d_i is the date of last occurrence of a. Also, for any value $\epsilon \leq 7$ the word $w_\epsilon = (a, d_1).(b, d_2) \dots (a, d_k).(b, d_k + 7 - \epsilon)$ is non-faulty. Let $v_1 = 12 - d_1$, recall that $d_1 \in (5, 12)$. If we choose $\epsilon < v_1$ then the run $w_\epsilon^+ = (a, d_1 + \epsilon).(b, d_2 + \epsilon) \dots (a, d_k + \epsilon).(b, d_k + 7)$ is also non-faulty because $5 < d_1 + \epsilon < d_1 + v_1 = 12$. Clearly, we can extend w_ϵ^+ to match transitions fired from w_ϵ hence, the automaton of the example is 1-$\forall$-*resilient*.

B K-$\exists$-*resilience* and untimed K-$\exists$-*resilience*

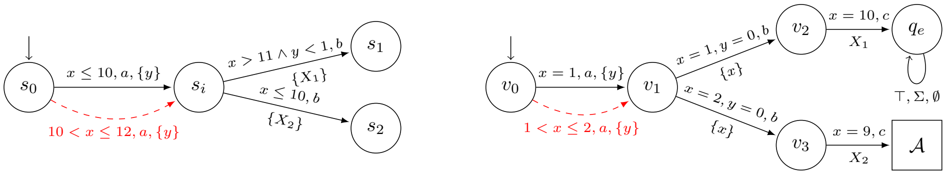

Figure 6 The gadgets $\mathcal{G}$ (left) and $\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}$ (right) which is untimed 2-$\exists$-*resilient* iff $\mathcal{L}(\mathcal{A}) \neq \emptyset$.

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

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

Let $q = (l, r)$ be a state of $\mathcal{R}(\mathcal{A}_\mathcal{P})$ reached after a just faulty run. K steps after reaching $q = (l, r)$ of $\mathcal{R}(\mathcal{A}_\mathcal{P})$, one can check in PSPACE, if there exists a path σ_s whose sequence of actions is the same as the suffix of an accepting path σ of $\mathcal{R}(\mathcal{A})$. That is, either both these end in a pair of accepting states from which no transitions are defined (both paths are maximal), or visit a pair of states twice such that the cyclic part of the path contains both an accepting state of $\mathcal{R}(\mathcal{A}_\mathcal{P})$ and an accepting state of $\mathcal{R}(\mathcal{A})$. To find these paths σ, σ_s, one just needs to guess them, i.e., build them synchronously by adding a pair of transitions to the already built path only if they have the same label. One needs to remember the current pair of states reached, and possibly guess a pair of states $(s_\mathcal{A}, s_{\mathcal{A}_\mathcal{P}})$ on which a cycle starts, and two bits $b_\mathcal{A}$ (resp. $b_{\mathcal{A}_\mathcal{P}}$) to remember if an accepting state of $\mathcal{A}$ (resp. $\mathcal{A}_\mathcal{P}$) has been seen since $(s_\mathcal{A}, s_{\mathcal{A}_\mathcal{P}})$. A maximal finite path or a lasso can be found on a path of length smaller than $|\mathcal{R}(\mathcal{A}_\mathcal{P})| \times |\mathcal{R}(\mathcal{A})|$, and the size of the currently explored path can be memorized with $\log_2(|\mathcal{R}(\mathcal{A}_\mathcal{P})| \times |\mathcal{R}(\mathcal{A})|)$ bits. This can be done in PSPACE. The complement of this, i.e., checking that no maximal path originating from q with the same labeling as a suffix of a word recognized by $\mathcal{R}(\mathcal{A})$ K steps after a fault exists, is in PSPACE too.

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

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 in Figure 6 (right). 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 in dotted red) then it resets all clocks of $\mathcal{A}$ and behaves as $\mathcal{A}$. The accepting states are $q_e \cup F$. Then, $\mathcal{A}$ has an accepting word if and only if $\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}$ is untimed 2-$\exists$-*resilient*. Since the emptiness problem for timed automata is PSPACE-Complete, the result follows. ◀

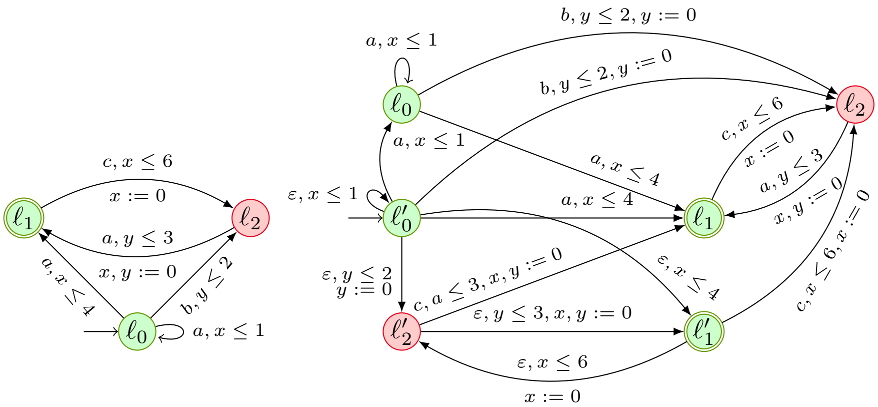

Figure 7 An example automaton $\mathcal{A}$ (left) and its suffix automaton $\mathcal{A}^S$ (right)

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

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

C Resilience of Integer Reset Timed Automata

Let us recall some elements used to prove decidability of language inclusion in IRTA. 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. We define another map $f_\downarrow : w \to \{\checkmark, \delta\}^*$ that maps every word w of $\mathcal{A}$ to a word in $\{\checkmark, \delta\}^*$ dropping the actions from $f(w)$. Consider for example, a word $w = (a, 1.6)(b, 2.7)(c, 3.4)$ then, $f(w) = \delta\checkmark\delta a\checkmark\delta b\checkmark\delta c$, and $f_\downarrow(w) = \delta\checkmark\delta\checkmark\delta\checkmark\delta$. It is shown in [18] 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 Automaton (MA) from $\mathcal{A}$ with one extra clock and polynomial increase in the number of locations such that $Unt(\mathcal{L}(MA)) = f(\mathcal{L}(\mathcal{A}))$. The MA of $\mathcal{A}$ duplicates transitions of $\mathcal{A}$ to differentiate firing at integral/non integral dates, plus transitions that make time elapsing visible using the additional clock which is reset at each global integral time stamp.

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

ii) $L' = L^0 \cup L^+$ *where for* $\alpha \in \{0, +\}, L^\alpha = \{l^\alpha \mid l \in L\}$

iii) $L'_0 = \{l^0 \mid l \in L_0\}$, $F' = \{l^0, l^+ \mid l \in F\}$ *and*

iv) T' *is defined by*
$$\begin{aligned} T' &= \{(l^0, a, g \wedge n = 0?, R, l'^0) \mid (l, a, g, R, l') \in E\} \\ &\cup \{(l^+, a, g \wedge 0 < n < 1?, R, l'^+) \mid (l, a, g, R, l') \in E\} \\ &\cup \bigcup_{l \in L} (l^0, \delta, 0 < n < 1, \emptyset, l^+) \cup \bigcup_{l \in L} (l^+, \checkmark, n = 1?, \{n\}, l^0) \end{aligned}$$

Then we have the following results.

▶ **Theorem 25** ([18]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 26. The marked timed automaton of an IRTA is also an IRTA.

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

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

▶ **Lemma 28.** *The timed suffix language of an IRTA $\mathcal{A}$ can be recognized by an ε-IRTA $\mathcal{A}^S$*

Proof. Let $\mathcal{A} = (L, X, \Sigma, T, \mathcal{G}, F)$ be a timed automaton. We create an automaton $\mathcal{A}^S = (L^S, X, \Sigma \cup \{\varepsilon\}, T^S, \mathcal{G}, F)$ as follows. We set $L^S = L \cup L_\varepsilon$, where $L_\varepsilon = \{l_\varepsilon \mid l \in L\}$ i.e., L^S contains a copy of locations in $\mathcal{A}$ and another "silent" copy. The initial location of $\mathcal{A}^S$ is $l_{0,\varepsilon}$. We set $T^S = T \cup T_\varepsilon \cup T'_\varepsilon$, where $T_\varepsilon = \{(l_\varepsilon, \varepsilon, true, \emptyset, l) \mid l \in L\}$ and $T'_\varepsilon = \{(l_\varepsilon, \varepsilon, g, R, l'_\varepsilon) \mid \exists (l, a, g, R, l') \in T\}$. 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 $w' = (\varepsilon, d_1) \dots . (\varepsilon, d_i)(a_{i+1}, d_{i+1}) \dots (a_n, d_n) = (a_{i+1}, d_{i+1}) \dots (a_n, d_n)$ is a recognized by $\mathcal{A}^S$, and it is easy to verify that $\mathcal{A}^s$ is an ε-IRTA. ◀

▶ **Lemma 29.** *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 $f(\rho) = w \in untimed(\mathcal{L}(\mathcal{B}_M))$ a contradiction. So, $untime(\mathcal{L}(\mathcal{A}_m)) \subseteq untime(\mathcal{L}(\mathcal{B}_m))$.

($\Leftarrow$)Assume, $untime(\mathcal{L}(\mathcal{A}_M)) \subseteq untime(\mathcal{L}(\mathcal{B}_M))$, and $\mathcal{L}(\mathcal{A}) \nsubseteq \mathcal{L}(\mathcal{B})$. 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, $w \in untime(\mathcal{L}(\mathcal{A}_M))$ and $w \in untime(\mathcal{L}(\mathcal{B}_M))$. So, there exists a timed word $\rho' \in \mathcal{L}(\mathcal{A})$ such that, $f(\rho') = w = f(\rho)$. According to Theorem 27 we can conclude that, $\rho \in \mathcal{L}(\mathcal{B})$ a contradiction. ◀

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

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

Proof. We proceed by a reduction from the language inclusion problem of IRTA, known to be EXPSPACE-Complete [4]. The idea of the proof follows the same lines as the untimed *K-$\forall$-resilience* of timed automata. Assume we are given IRTA $\mathcal{A}_1, \mathcal{A}_2$. a, b, c are symbols not in the alphabets of $\mathcal{A}_1, \mathcal{A}_2$. Consider $\mathcal{B}$ in Figure 8 (left). It is easy to see that

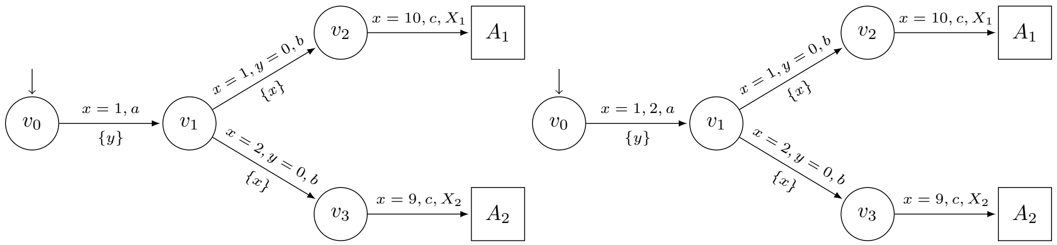

Figure 8 The automaton B (left) and the faulty automaton $B_{\mathcal{P}}$ (right)

$L(\mathcal{B})$=$(a,1)(b,1)(c,11)(L(\mathcal{A}_1)$+11), where $L(\mathcal{A}_1)+k = \{(a_1, d_1+k)(a_2, d_2+k)\dots(a_n, d_n+k) \mid (a_1, d_1)\dots(a_n, d_n) \in L(\mathcal{A}_1)\}$. 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 8 (right). Notice that in general, IRTAs are not closed under the fault insertion; the enlarged transition in $\mathcal{B}$ has guard $1 \le x \le 2$, and resets y. This violates the integer reset condition; however, since a value $1 < x < 2$ when 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 8 (right). 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 if and only if $L(\mathcal{A}_2) \subseteq L(\mathcal{A}_1)$. The EXPSPACE hardness of the timed K-$\forall$-*resilience* of IRTA follows from the EXPSPACE completeness of the inclusion of IRTA. ◀

▶ **Theorem 32.** *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 9 (left). Let us call this automaton $\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}$. It is easy to see that the $L(\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}})$=$(a,1)(b,1)(c,11)\big((\Sigma \times \mathbb{R})^* + 11\big)$, 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 9 (right). 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 $L(\mathcal{A}) \neq \emptyset$. ◀

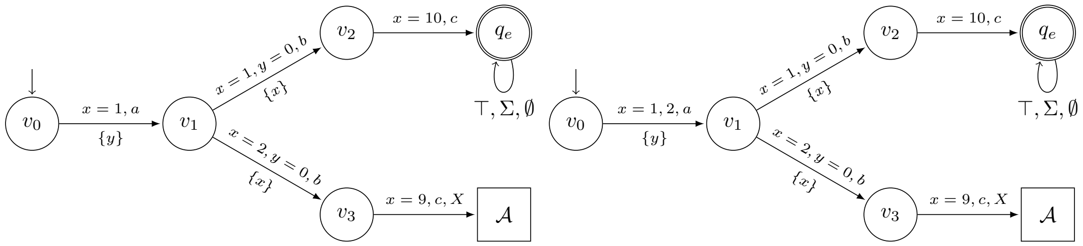

Figure 9 The IRTA $\mathcal{B}_{\Sigma^* \subseteq \mathcal{A}}$ (left) and the faulty IRTA $\mathcal{B}_{(\Sigma^* \subseteq \mathcal{A})_P}$ (right)

▶ Remark 33. The untimed language inclusion problem is shown to be EXPSPACE-Complete in Remark 30. The emptiness checking of timed automata is done by checking the emptiness of its untimed region automaton. So, to show the hardness of untimed K-$\forall$-*resilient* or K-$\exists$-*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 as shown in Theorem 31 and Theorem 32 respectively.