

# Resiliency

**Alain Finkel** Author: Please enter affiliation as second parameter of the author macro

**Mathieu Hilaire** ✉

Université Paris-Saclay

CNRS

ENS Paris-Saclay

Laboratoire Méthodes Formelles (LMF)

Gif-sur-Yvette, France

---

## Abstract

Here goes the abstract.

**2012 ACM Subject Classification** Theory of computation → Automata over infinite objects; Theory of computation → Automata extensions

**Keywords and phrases** Author: Please fill in \keywords macro

**Digital Object Identifier** 10.4230/LIPIcs...70

**Funding** *Mathieu Hilaire:* This work was partly done while the author was supported by the Agence Nationale de la Recherche grant no. (numero de la grant BraVASS).

**Acknowledgements** The author would like to thank.

## 1 Definitions

In this section, we introduce general notations and preliminary definitions.

The model we are interested in is (S)WSTS (and later some particular instances, i.e. Timed/Counter Automata for instance).

### 1.1 Transition systems

► **Definition 1.** A labeled transition system (*LTS* for short) is a tuple  $T = (S, \Lambda, \rightarrow)$  where  $S$  is a set of configurations,  $\Lambda$  is a set of labels, and  $\rightarrow \subseteq S \times \Lambda \times S$  is a ternary relation, denoted as the set of labeled transitions.

We prefer to use infix notation and  $(s, a, s') \in \rightarrow$  will be abbreviated as  $s \xrightarrow{a} s'$  to represent a transition from configuration  $s$  to configuration  $s'$  with label  $a$ .

Labels can be used to represent the reading of an input, but also to represent an action performed during the transition or conditions that must hold in order to allow the use of the transition.

A *path* in a labeled transition system from a *source configuration*  $s_0$  to a *target configuration*  $s_n$  is a sequence  $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \dots \xrightarrow{a_{n-1}} s_n$ . We define the *concatenation*  $\pi_1 \pi_2$  of two paths  $\pi_1$  and  $\pi_2$  when the source configuration of  $\pi_2$  is equal to the target configuration of  $\pi_1$  as expected. The *length* of  $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \dots \xrightarrow{a_{n-1}} s_n$  is defined as  $|\pi| = n$ . We say the path is *labeled* by  $a_0 a_1, \dots, a_{n-1}$ . For all  $w \in \Lambda^*$ , all  $s, s' \in S$ , we will write  $s \xrightarrow{w} s'$  if there exists a path from  $s$  to  $s'$  labeled by  $w$ .

An *infinite path* is an infinite sequence  $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \dots$ . For each infinite (resp. finite) path  $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \dots$  (resp.  $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \dots \xrightarrow{a_{n-1}} s_n$ ) and  $i, j \in \mathbb{N}$  (resp.  $i, j \in [0, n]$ ) with  $i < j$  we denote by  $\pi[i, j]$  the path  $s_i \xrightarrow{a_i} s_{i+1} \xrightarrow{a_{i+1}} \dots \xrightarrow{a_{j-1}} s_j$  and by  $\pi[i]$  the configuration  $s_i$ . As expected, a *prefix* of a finite or infinite path  $\pi$  is a finite path of the

form  $\pi[0, j]$ , and a *suffix* of a finite path  $\pi$  is a path of the form  $\pi[i, n]$ .  
 Given an infinite path  $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \dots$  let  $\text{Inf}(\pi) = \{s \in S \mid \forall i \exists j > i \ s_j = s\}$ .  
 The set of *successors* of a configuration  $s \in S$  is defined as  $\text{SUCC}(s) = \{s' \in S \mid \exists a \in \Lambda \ s \xrightarrow{a} s'\}$ .  
 A configuration without successors is called a *dead end*.  
 The set of *predecessors* of a configuration  $s \in S$  is defined as  $\text{PRED}(s) = \{s' \in S \mid \exists a \in \Lambda \ s' \xrightarrow{a} s\}$ .  
 A labeled transition system  $(S, \Lambda, \rightarrow)$  is *deterministic* if for all configurations  $s_1, s_2, s_3 \in S$  and all  $a \in \Lambda$ ,  $s_1 \xrightarrow{a} s_2$  and  $s_1 \xrightarrow{a} s_3$  implies  $s_2 = s_3$ .

► **Definition 2.** An (unlabeled) transition system is a pair  $T = (S, \rightarrow)$  where  $S$  is a set of configurations and  $\rightarrow \subseteq S \times S$  is a binary relation on the set of configurations, denoted as the set of transitions.

We again prefer to use infix notation and write  $s \rightarrow s'$  to denote a *transition* from configuration  $s$  to configuration  $s'$  (i.e.,  $(s, s') \in \rightarrow$ ).

Note that an unlabeled transition system can be seen as a labeled transition system where the set of labels consists of only one element. Determinism, (infinite) paths, their length, and concatenation in unlabeled transition systems are then defined as expected.

Thinking about whether or not it is pertinent to have LTS and not only TS. LTS can be usefull for TA because of the use of the guards/time as labels but it may be unnecessary.

We write  $\rightarrow^k, \rightarrow^+, \rightarrow^=, \rightarrow^*$  for the  $k$ -step iteration of  $\rightarrow$ , its transitive closure, its reflexive closure, its reflexive and transitive closure). We use similar notation for  $\text{SUCC}$  and  $\text{PRED}$ ...

This makes sense for TS but not so much for LTS ...

A transition system is *finitely branching* if all  $\text{SUCC}(s)$  are finite. We restrict our attention to finitely branching TSs.

Alain: the forward coverability algorithm for infinitely branching TSs.; the backward cov algo may work for essentially finitely branching TSs. Not sure that TS induced by TA are finitely branching. Actually I believe they are not, i.e. for instance for a TA with one clock  $x$ , from a state  $q$  and clock  $x$  set at 0, if there is a transition e.g.  $(q, x \geq 3, \emptyset, q')$  then the set of successors of  $(q, 0)$  is  $\{q'\} \times \{3, 4, 5, \dots\}$ . Need to check where finitely branching appears as an assumption/requirement.

## 1.2 Well-quasi-orderings

A *quasi-ordering* (a qo) is any reflexive and transitive relation  $\leq$ .

We abbreviate  $x \leq y \not\leq x$  by  $x < y$ .

Any qo induces an equivalence relation ( $x \equiv y$  iff  $x \leq y \leq x$ ).

We now recall a few results from the theory of well-orderings (add reference [...]).

► **Definition 3.** A well-quasi-ordering (a wqo) is any quasi-ordering  $\leq$  (over some set  $X$ ) such that, for any infinite sequence  $x_0, x_1, x_2, \dots$  in  $X$ , there exist indexes  $i \leq j$  with  $x_i \leq x_j$ .

Notice that a wqo is well-founded, i.e. it admits no infinite strictly decreasing sequence  $x_0 > x_1 > x_2 > \dots$

► **Lemma 4.** (Erdős and Rado). Assume  $\leq$  is a wqo. Then any infinite sequence contains an infinite increasing subsequence:  $x_{i_0} \leq x_{i_1} \leq x_{i_2} \dots$  (with  $i_0 < i_1 < i_2 \dots$ ).

Given  $\leq$  a quasi-ordering over some set  $X$ , an *upward-closed set* is any set  $I \subseteq X$  such that if  $y \geq x$  and  $x \in I$  then  $y \in I$ . A *downward-closed set* is any set  $I \subseteq X$  such that if  $y \leq x$  and  $x \in I$  then  $y \in I$ . To any  $A \subseteq X$  we associate the *upward-closure of*  $A \uparrow A = \{x \in X \mid \exists a \in A \ y \geq a\}$

and the downward-closure of  $A$   $\downarrow A = \{x \in X \mid \exists a \in A \ y \leq a\}$ . We abbreviate  $\uparrow \{x\}$  (resp.  $\downarrow \{x\}$ ) as  $\uparrow x$  (resp.  $\downarrow x$ ).

A *basis* of an upward-closed set  $I$  is a set  $I_b$  such that  $I = \bigcup_{x \in I_b} \uparrow x$ .

► **Lemma 5.** (Higman [40]) *If  $\leq$  is a wqo then any upward-closed  $I$  has a finite basis.*

**Proof.** The set of minimal elements of  $I$  is a basis because  $\leq$  is well-founded. It only contains a finite number of non-equivalent elements otherwise they would make an infinite sequence contradicting the wqo assumption. ◀

► **Lemma 6.** *If  $\leq$  is a wqo then any infinite increasing sequence  $I_0 \subseteq I_1 \subseteq I_2 \subseteq \dots$  of upward-closed sets eventually stabilizes; i.e. there is a  $k \in \mathbb{N}$  such that  $I_k = I_{k+1} = I_{k+2} = \dots$ .*

**Proof.** Assume we have a counter-example. We extract an infinite subsequence where inclusion is strict:  $I_{n_0} \subsetneq I_{n_1} \subsetneq I_{n_2} \dots$ . Now, for any  $i > 0$ , we can pick some  $x_i \in I_{n_i} \setminus I_{n_{i-1}}$ . The well-quasi-ordering hypothesis means that the infinite sequence of  $x_i$ 's contains an increasing pair  $x_i \leq x_j$  for some  $i < j$ . Because  $x_i$  belongs to an upward-closed set  $I_{n_i}$  we have  $x_j \in I_{n_i}$ , contradicting  $x_j \notin I_{n_{j-1}}$ . ◀

### 1.3 Well-structured transition systems

► **Definition 7.** A (resp. strongly) well-structured transition systems (abbreviated as WSTS (resp. SWSTS)) is a TS  $(S, \rightarrow, \leq)$  equipped with a wqo  $\leq \subseteq S \times S$  between states such that the wqo is (resp. strongly) compatible with the transition relation, i.e., for all  $s_1, t_1, s_2 \in S$  with  $s_1 \leq t_2$  and  $s_1 \rightarrow s_2$ , there exists  $t_2 \in S$  with  $s_2 \leq t_2$  and  $t_1 \rightarrow^* t_2$  (resp.  $t_1 \rightarrow^1 t_2$ ).

Several families of formal models of processes give rise to WSTSs in a natural way, e.g. Petri nets when inclusion between markings is used as the well-ordering.

► **Proposition 8.** *If  $S$  is an WSTS and  $I \subseteq S$  is an upward-closed set of states, then  $\text{PRED}^*(I)$  is upward-closed.*

► **Proposition 9.** *If  $S$  is an SWSTS and  $I \subseteq S$  is an upward-closed set of states, then  $\text{PRED}(I)$  is upward-closed.*

► **Definition 10.** A WSTS has effective pred-basis if there exists an algorithm accepting any state  $s \in S$  and returning  $\text{pb}(s)$ , a finite basis of  $\uparrow \text{PRED}(\uparrow s)$ .

A downward-closed set  $J$  is *decidable* if, given  $s \in S$ , it is decidable whether  $s \in J$ . Since a downward-closed set does not have an “upward-basis” in general, we will demand that membership is decidable.

▷ **Claim 11.** (stability of ideals) Let  $I, J \subseteq S$  be upward-closed. Then the sets  $I \cup J$ , and  $I \cap J$  are upward-closed.

- (i) For every upward-closed set  $I \subseteq S$ , there exists a finite basis  $B$  of  $I$ .
- (ii) Given a finite set  $A \subseteq S$  with  $I = \uparrow A$ , we can compute a finite basis  $B$  of  $I$ .

► **Definition 12.** (index). For an upward-closed set  $I \subseteq S$  and  $k \geq 0$ , let  $I^k = \bigcup_{0 \leq j \leq k} \text{pre}^j(I)$ . The index  $k(I)$  is the smallest  $k_0$  s.t.  $I^k = I^{k_0}$  for all  $k \geq k_0$ .

Let  $I \subseteq S$  be an upward-closed set and  $k \geq 0$  s.t.  $I^k = I^{k+1}$ , then  $I^\ell = I^k$  for all  $\ell \geq k$ , i.e.,  $k(I) \leq k$ . This also implies that  $\text{pre}^*(I) = I^k$ .

123 ► **Lemma 13.** *Given a basis of an upward-closed set  $I \subseteq S$ , and a state  $s$  of a strongly*  
 124 *well-structured transition system, we can decide whether we can reach  $I$  from  $s$ .*

**Proof.** We have to show that we can compute a basis of  $I^{k+1}$  if we are given a basis of  $I^k$ . Then the decidability of the stop condition follows directly. Let  $B$  be a basis of  $I^k$ . We have

$$I^{k+1} = I \cup \text{pre}(I^k) = I \cup \bigcup_{s' \in B} \text{pre}(\uparrow \{s'\}).$$

125 Since  $\text{pre}(\uparrow \{s'\})$  is computable for any  $s' \in S$  by definition, we obtain a finite generating  
 126 set of  $I^{k+1}$ . By Fact 3, we can compute a basis of  $I^{k+1}$ . ◀

## 127 1.4 Defining resilience

### 128 1.4.1 TS resilience

129 Ask the question why use a set of propositions for *SAFE* and *BAD* rather than use subsets  
 130 of the set of configurations ?

131 We ask whether we can reach a state in *SAFE* in a reasonable amount of time whenever  
 132 we reach a state in *BAD*. From this we formulate two resilience problems. First consider  
 133 the case where the recovery time is bound by a given natural number  $k \geq 0$ , i.e., the explicit  
 134 resilience problem for TS.

#### 135 TS $k$ -RESILIENCE PROBLEM

136 **INPUT:** A state  $s$  of a TS  $(S, \rightarrow)$ , two disjoint subset of  $S$  *SAFE* and *BAD*.

137 **QUESTION:**  $\forall s' \in \text{BAD} (s \rightarrow^* s') \implies \exists s'' \in \text{SAFE} s' \rightarrow^{\leq k} s''$  ?

138 We can also ask whether there exists such a bound  $k$ . We call this problem the bounded  
 139 resilience problem for TS.

#### 141 TS BOUNDED RESILIENCE PROBLEM

142 **INPUT:** A state  $s$  of a TS  $(S, \rightarrow)$ , two disjoint subset of  $S$  *SAFE* and *BAD*.

143 **QUESTION:**  $\exists k \geq 0 \forall s' \in \text{BAD} (s \rightarrow^* s') \implies \exists s'' \in \text{SAFE} s' \rightarrow^{\leq k} s''$  ?

### 145 1.4.2 WSTS resilience

146 Properties in well-structured transition systems are often given as upward- or downward  
 147 closed sets [references]. Transferring the abstract resilience problems into this framework, it  
 148 is therefore reasonable to demand that both propositions, *SAFE* and *BAD*, are given by  
 149 upward-closed or downward-closed sets.

150 We assume that the safety property is given by an upward-closed set and the bad condition  
 151 by a decidable downward-closed set.

152 From these considerations, we formulate instances of the abstract resilience problems for  
 153 well-structured transition systems.

154 Again, we first consider the case where the recovery time is bounded by a  $k \in \mathbb{N}$ .

#### 155 WSTS $k$ -RESILIENCE PROBLEM

156 **INPUT:** A state  $s$  of a WSTS  $(S, \rightarrow, \leq)$ , an upward-closed set  $I$  with a given basis, a  
 157 decidable downward-closed set  $J$ .

158 **QUESTION:**  $\forall s' \in J (s \rightarrow^* s') \implies \exists s'' \in I s' \rightarrow^{\leq k} s''$  ?

159 Analogously, we formulate the bounded resilience problem for WSTSs.  
 160

```

(1)       $i \leftarrow 0$ 
(2)      while  $\neg(\uparrow post^*(D_i) \subseteq D_i$  and
 $s \in D_i$  and  $b(D) \cap J \subseteq I^k)$  loop
(3)           $i \leftarrow i + 1$ 
(4)      end loop
(5)      return false

```

■ **Figure 1 Procedure 1:** enumerates inductive invariants to find an inclusion certificate.

## 161 WSTS BOUNDED RESILIENCE PROBLEM

162 **INPUT:** A state  $s$  of a WSTS  $(S, \rightarrow, \leq)$ , an upward-closed set  $I$  with a given basis, a  
163 decidable downward-closed set  $J$ .

164 **QUESTION:**  $\exists k \geq 0 \forall s' \in J (s \rightarrow^* s') \implies \exists s'' \in I s' \rightarrow^{\leq k} s'' ?$   
165

## 166 2 Decidability

167 ► **Theorem 14.** SWSTS  $k$ -RESILIENCE *is decidable*.

168 **Proof.** Assume  $(S, \rightarrow, \leq)$  is a SWSTS with upward compatibility,  $J$  is a decidable downward-  
169 closed subset of  $S$ , and  $I$  is an upward-closed set with a given basis.

170 We define inductively  $I^k = \bigcup_{0 \leq j \leq k} pre^j(I)$ . Note that for all  $k \in \mathbb{N}$ ,  $I^k$  is upward-closed  
171 due to the strongly upward compatibility of  $(S, \rightarrow, \leq)$ .

172 The  $k$ -resilience property can be expressed as the formula  $post^*(s) \cap J \subseteq I^k$ . In order  
173 to decide whether the inclusion holds, we execute two procedures in parallel, one trying to  
174 prove  $post^*(s) \cap J \subseteq I^k$  and one looking for a counter example.

175 In order to certify inclusion in  $I^k$ , we need to work with finite representations. The next  
176 lemma uses that  $I$  and  $J$  are upward- and downward-closed, respectively.

177 ► **Lemma 15.** Let  $A \subseteq S$  be a set,  $J \subseteq S$  downward-closed and  $I \subseteq S$  upward-closed. Then  
178  $A \cap J \subseteq I \leftrightarrow (\uparrow A) \cap J \subseteq I$ .

179 ► **Corollary 16.** For all  $k \in \mathbb{N}$ ,  $post^*(s) \cap J \subseteq I^k \leftrightarrow (\uparrow post^*(s)) \cap J \subseteq I^k$ .

180 Assume  $k$  is fixed for now.

181 Procedure 1 enumerates inductive invariants in some fixed order  $D_1, D_2, \dots$ , i.e.  
182 downward closed subsets  $D_i \subseteq S$  such that  $\uparrow post(D_i) \subseteq D_i$ . Every inductive invariant  $D_i$  is  
183 an “over-approximation” of  $\uparrow post^*(s)$  if it contains  $s$ . (on énumère des sur-approximations  
184 de la cloture par le haut de  $post^*(s)$  par leur bases finies). Each “over-approximation”  $D_i$   
185 is given by its basis  $b(D_i)$ . Notice that, by standard monotonicity,  $\uparrow post^*(s)$  is such an  
186 inductive invariant and may eventually be found.

187 Procedure 1 stops when it finds a basis  $b(D)$  of an invariant  $D$  such that  $b(D) \cap J \subseteq I^k$ .  
188 Since  $b(D)$  is finite and  $J$  is decidable, we can directly compute  $b(D) \cap J$ . We can compute  
189 a basis of  $I^{k+1}$  if we have a basis of  $I^k$ . Due to the Lemma,  $b(D) \cap J \subseteq I^k$  implies  $D \cap J \subseteq I^k$ .  
190 Hence  $b(D) \cap J \subseteq I^k$  implies  $\uparrow post^*(s) \cap J \subseteq D \cap J \subseteq I^k$ . (since  $D$  contains  $\uparrow post^*(s)$ ).

191 The second procedure iteratively computes  $post^{\leq n}(s) \cap J$  until it finds an element not in  
192  $I^k$ .

```

(1)    $D \leftarrow \{s\}$ 
(2)   while  $D \cap J \subseteq I^k$  loop
(3)      $D \leftarrow D \cup \text{post}(D)$ 
(4)   end loop
(5)   return false

```

■ **Figure 2 Procedure 2:** searches for a non-inclusion certificate.

193 We show that these two procedures are correct:

- 194 1.  $k$ -resilience holds if, and only if, Procedure 1 terminates.  
 195 2.  $k$ -resilience do not hold if, and only if, Procedure 2 terminates.

196 **Proof:**

- 197 1. By a simple induction, it can be shown that  $\uparrow \text{post}^*(D) \subseteq D$  for every inductive invariant  
 198  $D$ . If Procedure 1 terminates, then  $\text{post}^*(s) \cap J \subseteq \uparrow \text{post}^*(s) \cap J \subseteq D \cap J \subseteq I^k$  which implies  
 199 that  $k$ -resilience holds.  
 200 It remains to show that Procedure 1 terminates whenever  $k$ -resilience holds. To do so,  
 201 it suffices to prove that  $\uparrow \text{post}^*(s)$  is an inductive invariant. Indeed, this implies that  
 202  $\uparrow \text{post}^*(s)$  is eventually found by Procedure 1 when  $k$ -resilience holds.  
 203 Formally, let us show that  $\uparrow \text{post}(\uparrow \text{post}^*(s)) \subseteq \uparrow \text{post}^*(s)$ . Let  $b \in \uparrow \text{post}(\uparrow \text{post}^*(s))$  there  
 204 exists  $a', a, b$  such that  $s \rightarrow^* a'$ ,  $a' \leq a$ ,  $a \rightarrow b'$ , and  $b' \leq b$ . By downward compatibility  
 205 there exists  $b'' \leq b'$  such that  $a \rightarrow b''$ . Therefore,  $x \rightarrow^* b''$  and  $b' \geq b$ , hence  $b \in \downarrow \text{post}^*(x)$ .  
 206 2. Procedure 2 computes  $\text{post}^{\leq n}(s) \cap J$  until it finds an element not in  $I^k$ .  
 207 If Procedure 2 terminates, then  $k$ -resilience does not hold. It remains to show that  
 208 Procedure 2 terminates whenever  $k$ -resilience does not hold.

209

210 ► **Theorem 17.** SWSTS RESILIENCE is decidable.

211 **Proof.** sketch

212 Iteratively check whether  $k$ -resilience holds. If this is the case, return  $k_{\min} = k$ . Otherwise  
 213 check whether  $I^{k+1} = I^k$ . If so, return  $-1$  (false), otherwise continue. The stop condition is  
 214 decidable and by Fact 4 also sufficient. ◀

## 215 3 Applications section

### 216 3.1 Timed Automata

217 Should be defined in a later 'application section' once we start writing any proof, for now I  
 218 leave it there

219 A *guard* over a finite set of clocks  $\Omega$  is a comparison of the form  $\omega \bowtie c$ , where  $\omega \in \Omega$ ,  $c \in \mathbb{N}$ ,  
 220 and  $\bowtie \in \{<, \leq, =, \geq, >\}$ . We denote by  $\text{GUARDS}(\Omega)$  the *set of guards* over the set of clocks  $\Omega$ .  
 221 The *size* of a guard  $g = \omega \bowtie c$  is defined as  $|g| = \log(c)$ . A *clock valuation* is a function from  $\Omega$   
 222 to  $\mathbb{N}$ ; we write  $\vec{0}$  to denote the clock valuation  $\omega \mapsto 0$  whenever the set  $\Omega$  is clear from the  
 223 context. For each clock valuation  $v$  and each  $t \in \mathbb{N}$  we denote by  $v + t$  the clock valuation  
 224  $\omega \mapsto v(\omega) + t$ . For each guard  $g = \omega \bowtie c$  with  $c \in \mathbb{N}$ , we write  $v \models g$  if  $v(\omega) \bowtie c$ .

A timed automaton is a finite automaton extended with a finite set of clocks  $\Omega$  that all progress at the same rate and that can individually be reset to zero. Moreover, every transition is labeled by a guard over  $\Omega$  and by a set of clocks to be reset.

Formally, a *timed automaton* (TA for short) is a tuple  $\mathcal{A} = (Q, \Omega, R, q_{init}, F)$ , where

- $Q$  is a non-empty finite *set of states*,
- $\Omega$  is a non-empty finite *set of clocks*,
- $R \subseteq Q \times \mathbf{G}(\Omega) \times \mathcal{P}(\Omega) \times Q$  is a finite *set of rules*,
- $q_{init} \in Q$  is an *initial state*, and
- $F \subseteq Q$  is a *set of final states*.

We also refer to  $\mathcal{A}$  as an  $n$ -TA if  $|\Omega| = n$ . The *size* of  $\mathcal{A}$  is defined as

$$|\mathcal{A}| = |Q| + |\Omega| + |R| + \sum_{(q,g,U,q') \in R} |g|.$$

Let  $\text{Consts}(\mathcal{A}) = \{c \in \mathbb{N} \mid \exists (q, g, U, q') \in R, \exists \omega \in \Omega, \bowtie \in \{<, \leq, =, \geq, >\} : g = \omega \bowtie c\}$  denote the set of constants that appear in the guards of the rules of  $\mathcal{A}$ .

By  $\text{Conf}(\mathcal{A}) = Q \times \mathbb{N}^\Omega$  we denote the set of *configurations* of  $\mathcal{A}$ . We prefer however to abbreviate a configuration  $(q, v)$  by  $q(v)$ .

A TA  $\mathcal{A} = (Q, \Omega, R, q_{init}, F)$  induces the labeled transition system  $T_{\mathcal{A}} = (\text{Conf}(\mathcal{A}), \Lambda_{\mathcal{A}}, \rightarrow_{\mathcal{A}})$  where  $\Lambda_{\mathcal{A}} = R \times \mathbb{N}$  and where  $\rightarrow_{\mathcal{A}}$  is defined such that, for all  $(\delta, t) \in R \times \mathbb{N}$  with  $\delta = (q, g, U, q') \in R$ , for all  $q(v), q'(v') \in \text{Conf}(\mathcal{A})$ ,  $q(v) \xrightarrow{\delta, t}_{\mathcal{A}} q'(v')$  if  $v + t \models g$ ,  $v'(u) = 0$  for all  $u \in U$  and  $v'(\omega) = v(\omega) + t$  for all  $\omega \in \Omega \setminus U$ .

A *run* from  $q_0(v_0)$  to  $q_n(v_n)$  in  $\mathcal{A}$  is a path in the transition system  $T_{\mathcal{A}}$ , that is, a sequence  $\pi = q_0(v_0) \xrightarrow{\delta_1, t_1}_{\mathcal{A}} q_1(v_1) \cdots \xrightarrow{\delta_n, t_n}_{\mathcal{A}} q_n(v_n)$ ; it is called *reset-free* if for all  $i \in \{1, \dots, n\}$ ,  $\delta_i = (g_i, \emptyset)$  for some guard  $g_i$ .

We say  $\pi$  is *accepting* if  $q_0(v_0) = q_{init}(\vec{0})$  and  $q_n \in F$ .

It is worth mentioning that there are further modes of time valuations and guards which exist in the literature, we refer to [1] for a recent overview. Notably, we consider in this article only the case of timed automata over discrete time. It is worth mentioning that in the case of timed automata over continuous time (i.e. with clocks having values in  $\mathbb{R}_{\geq 0}$ ), techniques [2, 3] exist for reducing the reachability problem to discrete time in the case of closed (i.e. non-strict) clock constraints ranging over integers.

#### TA $k$ -RESILIENCE PROBLEM

**INPUT:** A state  $q$  of a TA  $(Q, X, \Delta)$ , a set  $SAFE \subseteq Q$ , a set  $BAD \subseteq Q$ .

**QUESTION:**  $\forall q' \in BAD \forall v, v' \in \mathbb{N}^X (q(v) \rightarrow^* q'(v')) \implies \exists q'' \in SAFE \exists v'' \in \mathbb{N}^X q'(v') \rightarrow^{\leq k} q''(v'') ?$

Analogously, we formulate the bounded resilience problem for WSTSs.

#### TA BOUNDED RESILIENCE PROBLEM

**INPUT:** A state  $q$  of a TA  $(Q, X, \Delta)$ , a set  $SAFE \subseteq Q$ , a set  $BAD \subseteq Q$ .

**QUESTION:**  $\exists k \geq 0 \forall q' \in BAD \forall v, v' \in \mathbb{N}^X (q(v) \rightarrow^* q'(v')) \implies \exists q'' \in SAFE \exists v'' \in \mathbb{N}^X q'(v') \rightarrow^{\leq k} q''(v'') ?$

I think there can be a discussion to be had here about how to quantify on the clock valuations

Here one thing that could be interesting to try to formalize is: how to enforce that the time that passes is less than  $k$ , rather than the number of transitions. This is tricky to deal with I find but it should be more doable if for instance we use one counter automata, where the counter effect of the sequence can be quantified more explicitly I suppose ? But here you could also use a kinda special clock  $x$  that is reset when you enter *BAD* and is not reset between a state in *BAD* and a state in *SAFE*, you could check that  $x < k$ .

... I guess if you use 0/1-TA then the problems become closer one to another ? Also of note is that 0/1-TA induces transition systems with bounded branching, so I guess it may be interesting to investigate these first ?

A 0/1 *timed automaton* (0/1-TA for short) is a tuple

$$\mathcal{B} = (Q, X, \Delta_0, \Delta_1, q_{init}, F),$$

where  $\mathcal{B}_i = (Q, X, R_i, q_{init}, F)$  is a TA for all  $i \in \{0, 1\}$ . For simplicity we define its *size* as  $|\mathcal{B}| = |\mathcal{B}_0| + |\mathcal{B}_1|$ . We analogously denote the constants of  $\mathcal{B}$  by  $\text{Consts}(\mathcal{B})$  and its configurations by  $\text{Conf}(\mathcal{B})$ .

A 0/1 timed automaton  $\mathcal{B} = (Q, X, R_0, R_1, q_{init}, F)$  induces the labeled transition system  $T_{\mathcal{B}} = (\text{Conf}(\mathcal{B}), \lambda_{\mathcal{B}}, \rightarrow_{\mathcal{B}})$  where  $\lambda_{\mathcal{B}} = (R_0 \cup R_1) \times \{0, 1\}$  and where  $\rightarrow_{\mathcal{B}}$  is defined such that for all  $q(z), q'(z') \in \text{Conf}(\mathcal{B})$ , for all  $(\delta, i) \in \lambda_{\mathcal{B}}$  with  $\delta = (q, g, U, q') \in R_i$   $q(v) \xrightarrow{\delta, i}_{\mathcal{B}} q'(v')$  if  $v + i \models g$ ,  $v'(u) = 0$  for all  $u \in U$  and  $v'(\omega) = v(\omega) + i$  for all  $\omega \in \Omega \setminus U$ .

As expected, we write  $q(v) \xrightarrow{\delta, i}_{\mathcal{B}} q'(v')$  if  $q(v) \xrightarrow{\delta, i}_{\mathcal{B}} q'(v')$  for some  $i \in \{0, 1\}$ , and some  $\delta \in R_i$ .

### 3.2 One-Counter Automata

Should be defined in a later 'application section' once we start writing any proof, for now I leave it there

#### OCA $k$ -RESILIENCE PROBLEM

**INPUT:** A state  $q$  of a OCA  $(Q, \Delta)$ , a set  $\text{SAFE} \subseteq Q$ , a set  $\text{BAD} \subseteq Q$ .

**QUESTION:**  $\forall q' \in \text{BAD} \forall n, n' \in \mathbb{N} (q(n) \rightarrow^* q'(n')) \implies \exists q'' \in \text{SAFE} \exists n'' \in \mathbb{N} q'(n') \rightarrow^{\leq k} q''(n'') ?$

#### OCA BOUNDED RESILIENCE PROBLEM

**INPUT:** A state  $q$  of a OCA  $(Q, \Delta)$ , a set  $\text{SAFE} \subseteq Q$ , a set  $\text{BAD} \subseteq Q$ .

**QUESTION:**  $\exists k \geq 0 \forall q' \in \text{BAD} \forall n, n' \in \mathbb{N} (q(n) \rightarrow^* q'(n')) \implies \exists q'' \in \text{SAFE} \exists n'' \in \mathbb{N} q'(n') \rightarrow^{\leq k} q''(n'') ?$

### 3.3 Vector Addition System with States

Should be defined in a later 'application section' once we start writing any proof, for now I leave it there

#### References

- 1 Étienne André. What's decidable about parametric timed automata? *Int. J. Softw. Tools Technol. Transf.*, 21(2):203–219, 2019. doi:10.1007/s10009-017-0467-0.
- 2 Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. What good are digital clocks? In Werner Kuich, editor, *Automata, Languages and Programming, 19th International Colloquium*,



- 306 *ICALP92, Vienna, Austria, July 13-17, 1992, Proceedings*, volume 623 of *Lecture Notes in*  
 307 *Computer Science*, pages 545–558. Springer, 1992.
- 308 3 Joël Ouaknine and James Worrell. Universality and language inclusion for open and closed  
 309 timed automata. In Oded Maler and Amir Pnueli, editors, *Hybrid Systems: Computation and*  
 310 *Control, 6th International Workshop, HSCC 2003 Prague, Czech Republic, April 3-5, 2003,*  
 311 *Proceedings*, volume 2623 of *Lecture Notes in Computer Science*, pages 375–388. Springer,  
 312 2003. doi:10.1007/3-540-36580-X\_28.

313 **A** Appendix thing if necessary