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).

Before defining WSTS, need a definition of TS and WQO

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*, Λ 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 $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} s_n$. We define the *concatenation* $\pi_1\pi_2$ of two paths π_1 and π_2 when the source configuration of π_2 is equal to the target configuration of π_1 as expected. The *length* of $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} s_n$ is defined as $|\pi| = n$. We say the path is *labeled* by $a_0 a_1, \ldots a_{n-1}$. 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 $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$. For each infinite (resp. finite) path $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$ (resp. $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} s_n$) and $i, j \in \mathcal{N}$ (resp. $i, j \in [0, n]$) with $i < j$ we denote by $\pi[i, j]$ the path $s_i \xrightarrow{a_i} s_{i+1} \xrightarrow{a_{i+1}} \cdots \xrightarrow{a_{j-1}} s_j$ and by $\pi[i]$

the configuration s_i. As expected, a *prefix* of a finite or infinite path π is a finite path of the form $\pi[0,j]$, and a *suffix* of a finite path π is a path of the form $\pi[i,n]$.
Given an infinite path $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$ let $Inf(\pi) = \{s \in S \mid \forall i\ \exists j > i\ s_j = s\}$.
The set of *successors* of a configuration $s \in S$ is defined as $\textsc{Succ}(s) = \{s' \in S \mid \exists a \in \Lambda\ s \xrightarrow{a} s'\}$.
A configuration without successors is called a *dead end*.
The set of *predecessor* of a configuration $s \in S$ is defined as $\textsc{Pred}(s) = \{s' \in S \mid \exists a \in \Lambda\ s' \xrightarrow{a} s\}$.

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 unecessary.

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 $\textsc{Succ}$ and $\textsc{Pred}$...

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

A transition system is *finitely branching* if all $\textsc{Succ}(s)$ 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. $(q, x \geq 3, \varnothing, q')$ then the set of successors of $(q, 0)$ is $\{q'\} \times \{3, 4, 5, \ldots\}$. 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 $x \leq y \nleq x$ 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 $x_0, x_1, x_2, \ldots$ 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 $x_0 > x_1 > x_2 > \cdots$

Add lemma about infinite increasing subsequences ?

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. $\downarrow x$).

▶ **Lemma 4.** *(Higman [40]) If $\le$ is a wqo; then any upward-closed I has a finite basis.*

Expliquer ce que c'est une base d'abords.

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

Alain: non ceci est vrai seulement si le quasi ordre est un ordre cad antisymétrique. Mais ce n'est pas un pb. Relis mon papier de 2016 sur les well abstracted...

Un lemme je pense c'est important de le noter, peut-être pas comme ça peut être noter différement faudra voir

▶ **Lemma 5.** *If $\le$ is a wqo; any infinite increasing sequence $I_0 \subseteq I_1 \subseteq I_2 \subseteq \cdots$ of upward-closed sets eventually stabilizes; i.e. there is a $k \in N$ such that $I_k = I_{k+1} = I_{k+2} = \cdots$.*

Proof. Assue we have a counter-example ... ◀

Define WSTS
Define SWSTS - may be necessary

1.3 Well-structured transition systems

▶ **Definition 6.** *A* (resp. strongly) well-structured transition systems *(abbreviated as WSTS resp. SWSTS) is a TS $(S, \to, \le)$ equipped with a wqo $\le \subseteq S \times S$ 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 $s_1 \le t_2$ and $s_1 \to s_2$, there exists $t_2 \in S$ with $s_2 \le t_2$ and $t_1 \to^* t_2$ (resp. $t_1 \to^1 t_2$).*

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.

For one-counter automata, in case the only tests are zero tests then I supposed $\le$ is $\le$ for non-zero integers, and I'll have to look-up/think for what to do with the zero element for instance. For TA it seem kind of nontrivial (since they allow $< c$ tests).

Define 'has effective pred-basis'. Maybe it should be included in WSTS definition, maybe it can be another def. I kind of like the idea of 'effective pred-basis' and 'decidable $\le$' being independant from the WSTS definition

▶ **Definition 7.** *A WSTS has* effective pred-basis *if there exists an algorithm accepting any state $s \in S$ and returning $pb(s)$, a finite basis of $\uparrow \textsc{Pred}(\uparrow s)$.*

Define what an Ideal is. Actually an Ideal is just an upward-closed set, so maybe this just adds some confusion. Anti-ideal just downward closed so again just not that helpful a notation. Maybe have a

▶ **Definition 8.** *A* bi-ideal *$I \subseteq S$ is an upward-closed and downward-closed set, i.e $\uparrow I = I = \downarrow I$.*

A downard-closed set J is *decidable* if, given $s \in S$, it is decidable whether $s \in J$.

"Bi-ideals often represent "control states" as in [cf %]. "

Probably one can already 'deduce' from this that ideal I and anti-ideal J for resp. good and bad states, in the case of e.g. timed automata would be given by sets of states

Since a downward-closed set does not have an "upward-basis" in general, we will demand that membership is decidable.

Do we still demand this ?

▷ Claim 9. (stability of ideals) Let $I, J \subseteq S$ be upward-closed. Then the sets $\textsc{Pred}(I)$, $I \cup J$, and $I \cap J$ are upward-closed.

1.4 Defining resilience

1.4.1 TS resilience

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

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

TS k-RESILIENCE PROBLEM

INPUT: A state s of a TS $(S, \to)$, two disjoints subset of S $SAFE$ and BAD.
QUESTION: $\forall s' \in BAD\ (s \to^* s') \implies \exists s'' \in SAFE\ s' \to^{\leq k} s''$?

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

TS BOUNDED RESILIENCE PROBLEM

INPUT: A state s of a TS $(S, \to)$, two disjoints subset of S $SAFE$ and BAD.
QUESTION: $\exists k \geq 0\ \forall s' \in BAD\ (s \to^* s') \implies \exists s'' \in SAFE\ s' \to^{\leq k} s''$?

1.4.2 WSTS resilience

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

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

Seems like a reasonable assumption to me.

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

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

WSTS k-RESILIENCE PROBLEM

INPUT: A state s of a WSTS $(S, \to, \leq)$, an upward-closed set I with a given basis, a decidable downward-closed set J.
QUESTION: $\forall s' \in J\ (s \to^* s') \implies \exists s'' \in I\ s' \to^{\leq k} s''$?

Analogously, we formulate the bounded resilience problem for WSTSs.

WSTS BOUNDED RESILIENCE PROBLEM

INPUT: A state s of a WSTS $(S, \to, \leq)$, an upward-closed set I with a given basis, a decidable downward-closed set J.
QUESTION: $\exists k \geq 0\ \forall s' \in J\ (s \to^* s') \implies \exists s'' \in I\ s' \to^{\leq k} s''$?

In the Özkan paper the input include a basis of $\uparrow post^*(s)$. Not 100% sure it is necessary, think we can try to do without this assumption in the input.

2 Decidability

Proof. sketch

Si on inverse les types de propriétés (downward closed pour $I = SAFE$, upward closed pour $J = BAD$) alors on peut écrire un lemme symétrique du Lemme 4:

▶ **Lemma 10.** *Let $A \subseteq S$ be a set, $J \subseteq S$ upward-closed and $I' \subseteq S$ downward-closed. Then $A \cap J \subseteq I' \leftrightarrow (\downarrow A) \cap J \subseteq I'$.*

Et une fois que l'on a ça, on peut écrire $post^*(s) \cap J \subseteq I' \leftrightarrow (\downarrow post^*(s)) \cap J \subseteq I'$ et travailler avec $\downarrow post^*(s)$ à la place de $\uparrow post^*(s)$.

La question deviens: quel k pour que $\downarrow post^*(s) \cap J \subseteq I^k$. Ou alors, à k fixé, est-ce que $\downarrow post^*(s) \cap J \subseteq I^k$.

Il faut aussi une condition pour s'assurer que I^k soit bien downward-closed.

Où $I^{k+1} = I \cup pre(I^k)$.

On vas imaginer temporairement qu'on arriver à s'assurer I^k downward-closed. (Peut-être on doit faire la supposition de SWSTS strongly downward compatible à la place, pour avoir ça ?)

On reviens sur $\downarrow post^*(s) \cap J \subseteq I^k$.

C'est là peut être qu'on peut s'inspirer de l'algo forward.

Enumerates inductive invariants in some fixed order D_1 , D_2 , . . . , i.e. downward closed subsets $D_i \subseteq X$ such that $\downarrow Post(D_i) \subseteq D_i$. [...] Every inductive invariant D_i is an "over-approximation" of $post^*(x)$ if it contains x.

On a un nombre fini de bornes supérieures pour D_i on peut considèrer qu'elles consistent en une base de D_i ? On énumère les D_i en énumérant leurs bases directement ?

Ensuite, si jamais $Base(D_1) \cap J \subseteq I^k$.

alors

$D_1 \cap J \subseteq I^k$ (par le lemme)

et donc

$\downarrow post * (s) \cap J \subseteq D_1 \cap J \subseteq I^k$

(puisque D_1 est une sur-approximation).

Ce qui voudrais dire que on a bien la propriété de résilience.

Le problème évidement c'est que cette procédure ne termine peut être pas enfin on n'a aucune guarrantie à priori. À ce moment là il faudrait sans doute faire une autre procédure en parallèle, comme dans l'algo forward, et qui elle terminerais si on a *pas* la propriété de résilience.

La propriété de k-résilience peut être résumée par la formule: $post^*(s) \cap J \subseteq I^k$.

Donc la question ce serait est-ce que, à k fixé (k fixé pour l'instant on vas dire) on aurais une procédure qui termine pour montrer que ce n'est pas le cas ?

Ce serait une procédure qui pourrait par exemple calculer $post^m(s) \cap J$ jusqu'à trouver un élément pas dans I^k ?

Càd on calcule les éléments de $post^m(s)$ un à un, pour chacun on vérifie s'il est dans J puis s'il y est on vérifie s'il est dans I^k ?

Il faut aussi une base de J et que I^k soit décidable, mais ça a l'air faisable j'ai l'impression...

Ça me perturbe un peu quand même de devoir inverser toutes les propriétés et avoir SWSTS downward-compatible, je pense ça doit être possible de faire les choses mieux...

◀

3 Applications section

3.1 Timed Automata

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

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

A timed automaton is a finite automaton extended with a finite set of clocks Ω that all progress at the same rate and that can individually be reset to zero. Moreover, every transition is labeled by a guard over Ω 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*,
- Ω is a non-empty finite *set of clocks*,
- $R \subseteq Q \times \mathsf{G}(\Omega) \times \mathcal{P}(\Omega) \times Q$ 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 $\mathsf{Consts}(\mathcal{A}) = \{c \in \mathcal{N} \mid \exists (q, g, U, q') \in R,\ \exists \omega \in \Omega, \bowtie \in \{<, \leq, =, \geq, >\} : g = \omega \bowtie c\}$ denote the set of constants that appear in the guards of the rules of $\mathcal{A}$.

By $\mathsf{Conf}(\mathcal{A}) = Q \times \mathcal{N}^{\Omega}$ 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 $T_{\mathcal{A}} = (\mathsf{Conf}(\mathcal{A}), \Lambda_{\mathcal{A}}, \to_{\mathcal{A}}$) where $\Lambda_{\mathcal{A}} = R \times \mathcal{N}$ and where $\to_{\mathcal{A}}$ is defined such that, for all $(\delta, t) \in R \times \mathcal{N}$ with $\delta = (q, g, U, q') \in R$, for all $q(v), q'(v') \in \mathsf{Conf}(\mathcal{A})$, $q(v) \xrightarrow{\delta,t}_{\mathcal{A}} q'(v')$ 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 $i \in \{1, \ldots, n\}$, $\delta_i = (g_i, \varnothing)$ for some guard g_i.

We say π 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 [?] 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 $\mathsf{R}_{\geq 0}$), techniques [?, ?] exist for reducing the reachability problem to discrete time in the case of closed (i.e. non-strict) clock constraints ranging over integers.

<u>TA k-RESILIENCE PROBLEM</u>

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

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

Analogously, we formulate the bounded resilience problem for WSTSs.

TA BOUNDED RESILIENCE PROBLEM

INPUT: A state q of a TA (Q, X, Δ), a set $SAFE \subseteq Q$, a set $BAD \subseteq Q$.
QUESTION: $\exists k \geq 0 \; \forall q' \in BAD \forall v, v' \in \mathcal{N}^X \; (q(v) \to^* q'(v')) \implies \exists q'' \in SAFE \exists v'' \in \mathcal{N}^X \; q'(v') \to^{\leq k} q''(v'')$?

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 $\mathsf{Consts}(\mathcal{B})$ and its configurations by $\mathsf{Conf}(\mathcal{B})$.

A 0/1 timed automaton $\mathcal{B} = (Q, X, R_0, R_1, q_{init}, F)$ induces the labeled transition system $T_{\mathcal{B}} = (\mathsf{Conf}(\mathcal{B}), \lambda_{\mathcal{B}}, \to_{\mathcal{B}})$ where $\lambda_{\mathcal{B}} = (R_0 \cup R_1) \times \{0, 1\}$ and where $\to_{\mathcal{B}}$ is defined such that for all $q(z), q'(z') \in \mathsf{Conf}(\mathcal{B})$, 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 $v + i \vDash g$, $v'(u) = 0$ 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, Δ), a set $SAFE \subseteq Q$, a set $BAD \subseteq Q$.
QUESTION: $\forall q' \in BAD \forall n, n' \in \mathcal{N} \; (q(n) \to^* q'(n')) \implies \exists q'' \in SAFE \exists n'' \in \mathcal{N} \; q'(n') \to^{\leq k} q''(n'')$?

OCA BOUNDED RESILIENCE PROBLEM

INPUT: A state q of a OCA (Q, Δ), a set $SAFE \subseteq Q$, a set $BAD \subseteq Q$.
QUESTION: $\exists k \geq 0 \; \forall q' \in BAD \forall n, n' \in \mathcal{N} \; (q(n) \to^* q'(n')) \implies \exists q'' \in SAFE \exists n'' \in \mathcal{N} \; q'(n') \to^{\leq k} q''(n'')$?

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

A Appendix thing if necessary