Robust Model Checking of Timed Automata under Clock Drifts

Nima Roohi
Department of Computer
Science, University of Illinois
at Urbana-Champaign, Illinois,
USA
roohi2@illinois.edu

Pavithra Prabhakar
Department of Computer
Science, Kansas State
University, KS, USA
pprabhakar@ksu.edu

Mahesh Viswanathan
Department of Computer
Science, University of Illinois
at Urbana-Champaign, Illinois,
USA
vmahesh@illinois.edu

ABSTRACT

Timed automata have an idealized semantics where clocks are assumed to be perfectly continuous and synchronized, and guards have infinite precision. These assumptions cannot be realized physically. In order to ensure that correct timed automata designs can be implemented on real-time platforms, several authors have suggested that timed automata be studied under robust semantics. A timed automaton $\mathcal{H}$ is said to robustly satisfy a property if there is a positive ϵ and/or a positive δ such that the automaton satisfies the property even when the clocks are allowed to drift by ϵ and/or guards are enlarged by δ . In this paper we show that, 1. checking ω -regular properties when only clocks are perturbed or when both clocks and guards are perturbed, is PSPACE-complete; and 2. one can compute the exact reachable set of a bounded timed automaton when clocks are drifted by infinitesimally small amount, using polynomial space. In particular, we remove the restrictive assumption on the timed automaton that its region graph only contains progress cycles, under which the second result above has been previously established.

1. INTRODUCTION

Timed automata [2] are the standard formal model for real-time systems because they are an elegant and expressive formalism, and, yet are amenable to algorithmic analysis [14, 18]. However, timed automata have an idealistic semantics that makes assumptions that are physically unrealizable. In timed automata, time is measured by perfectly continuous and synchronous clocks that have infinite precision as opposed to finite precision, almost synchronized, digital clocks that are accessed by implementations. Timed automata can also respond instantaneously to events whereas physical realizations of the real-time systems react with some non-zero delay. Finally, timed automata allow modelling control algorithms that exhibit Zeno behaviors, have unrealistic convergence properties [10], or isolated behaviors [12]. These deficiencies have been observed by a number of re-

searchers [1, 3, 10, 12, 16, 17]. In order to ensure that correct timed automata yield correct, implementable designs, the remedy suggested by these papers is to consider a *robust* semantics for timed automata.

Starting from the seminal work of [12] where a topological notion of robustness was proposed, different notions of robustness have been considered [3, 15, 16]. One notion of robustness that has been studied extensively, is the following. For a timed automaton $\mathcal{H}$, using $\mathcal{H}_\delta^\epsilon$ to denote the automaton similar to $\mathcal{H}$ with guards enlarged by δ and clocks allowed to drift by ϵ , robust satisfaction of φ asks if there is some $\epsilon \in \mathbb{R}_+$ and $\delta \in \mathbb{R}_+$ such that $\mathcal{H}_\delta^\epsilon$ satisfies φ . This notion of robustness has been shown to imply implementability in real-time platforms [11, 16].

The algorithmic complexity of checking the robustness of timed automata designs has received much attention. For a timed automaton $\mathcal{H}$, let us denote by $\mathcal{H}^\epsilon$ the semantics where clocks can drift by ϵ (but guards remain unperturbed), and by $\mathcal{H}_\delta$ the one where guards are enlarged by δ (but clocks do not drift). Robust satisfaction under only clock drifts asks if for some $\epsilon \in \mathbb{R}_+$, $\mathcal{H}^\epsilon$ satisfies the property, while robust satisfaction under only enlarged guards asks if for some $\delta \in \mathbb{R}_+$, $\mathcal{H}_\delta$ satisfies the property. For safety properties, and bounded timed automata¹ the problem was first considered in [15]. It is shown that robustness with only clocks drifts with respect to safety properties can be decided in PSPACE. These results were generalized in [11] where safety verification under both clock drifts and enlarged guards is solved in PSPACE. The approach in both [15] and [11] is based on computing the reachable states under infinitesimal guard and clock perturbations (or only guard or only clock perturbations), referred to as $\text{limreach}_\delta^\epsilon(\mathcal{H})$ (or $\text{limreach}_\delta(\mathcal{H})$ or $\text{limreach}^\epsilon(\mathcal{H})$). It is shown that $\mathcal{H}$ is robustly safe (under just clock drifts, or guard perturbations, or both) iff the corresponding limreach set is disjoint from the unsafe states. Moreover, it is observed in [11] that the three limreach sets coincide, that is,

$$\text{limreach}_\delta^\epsilon(\mathcal{H}) = \text{limreach}^\epsilon(\mathcal{H}) = \text{limreach}_\delta(\mathcal{H}) \quad (1)$$

Except the results about disjointness of unsafe and limreach sets, all the above results on safety verification (including the fact that limreach sets coincide) in [11, 15] are established for timed automata that satisfy the *progress cycle assumption* (PCA) which requires that every cycle in the region graph of $\mathcal{H}$ resets every clock of timed automaton at least once.

¹A timed automaton is *bounded* if the invariant of every location is bounded.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

HSCC'17, April 18 - 20, 2017, Pittsburgh, PA, USA

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-1-4503-4590-3/17/04...\$15.00

DOI: <http://dx.doi.org/10.1145/3049797.3049821>

The progress cycle assumption can be restrictive when modeling real-time systems because it does not allow the design to measure time spent in cycles. Therefore, the decidability of robust verification without the progress cycle assumption was investigated in [8]. The results from [11, 15] were generalized in a couple of directions. First the progress cycle assumption was removed. Second, general ω -regular properties were considered as opposed to just safety. In addition, the restriction to bounded timed automata was also removed; this restriction to bounded automata is, however, not limiting because every timed automaton is weakly bisimilar to a bounded timed automaton [5, 7]. However, in some respects, the results in [8] are also less general than those in [11]; the paper [8] only considers robustness with respect to perturbation of guards alone.

In this paper, we continue this line of investigation. In the absence of the progress cycle assumption, we ask what is the complexity of robustly verifying a property when clocks are allowed to drift, as well. Our first observation is that an automaton $\mathcal{H}$ robustly satisfies a property $\Box\Diamond B$ when both clocks and guards are perturbed iff $\mathcal{H}$ robustly satisfies the same property when only guards are enlarged. In addition, $\text{limreach}_\delta^\epsilon(\mathcal{H}) = \text{limreach}_\delta(\mathcal{H})$. Thus, using the algorithm in [8], one can verify designs under both clock drifts and guard perturbation. On the other hand, we show that robustness when only clocks are drifted is not equivalent with the *stronger* notion of robustness when both guards and clocks are perturbed. More precisely, we show that there are timed automata that are robust when only clocks are drifted, but not when both guards and clocks are perturbed (see Example 12). This is in contrast to Equation (1) that holds under the progress cycle assumption. We then present an algorithm to check ω -regular properties when only clocks are drifted. We show that a timed automaton $\mathcal{H}$ robustly satisfies a property $\Box\Diamond B$ when only clocks are drifted iff there is a constant δ_1 (depending only on the size of $\mathcal{H}$) such that the automaton in which only the guard constraints involving *positive* constants are perturbed by δ_1 , satisfies $\Box\Diamond B$. This observation can be exploited to give a PSPACE algorithm. We also show that this problem is PSPACE-hard, thus establishing the optimality of our algorithm.

Next, we consider the problem of computing limreach^ϵ for bounded timed automata in the absence of the progress cycle assumption. While the algorithm to verify ω -regular properties discussed in the previous paragraph also applies to robust safety verification, computing limreach^ϵ is of independent interest just like computing reachable sets is an important task independent of safety verification. Puri's algorithm [15] (generalized in [11]) works by iteratively adding the (topological) closure of regions on progress cycles that have a non-empty intersection with the current limreach set. We show that the (almost) same algorithm correctly computes limreach^ϵ even when the progress cycle assumption does not hold. This algorithm can be shown to use polynomial space. Our contributions, in the context of previous results, is summarized in Table 1. As can be seen from the table, the results in this paper cannot be used to compute $\text{limreach}_\delta^\epsilon$ nor limreach_δ , in the absence of the progress cycle assumption.

2. PRELIMINARIES

The sets of *natural*, *positive natural*, *positive rational*, *real*, *positive real*, and *non-negative real*, numbers are represented by $\mathbb{N}$, $\mathbb{N}_+$, $\mathbb{Q}_+$, $\mathbb{R}$, $\mathbb{R}_+$, and $\mathbb{R}_{\geq 0}$, respectively. For $r \in \mathbb{R}_{\geq 0}$,

Problem / Assumption	With PCA	Without PCA
$\text{limreach}_\delta^\epsilon$ computation	✓ [11]	
limreach_δ computation	✓ [11]	
limreach^ϵ computation	✓ [11, 15]	(✓)
Robust ω -regular	(✓)	(✓)
Robust ω -regular with only Enlarged Guards	✓ [6]	✓ [8]
Robust ω -regular with only Drifted Clocks	(✓)	(✓)

Table 1: Summary of Robust Model Checking Problem Results: Check marks indicate that the corresponding problem is solvable in polynomial space; check marks within parenthesis are established in this paper and those without the parenthesis were established in the reference cited alongside. Recall that PCA refers to the Progress Cycle Assumption.

we define $\lfloor r \rfloor$ to be integer part of r , and $\langle r \rangle$ to be $r - \lfloor r \rfloor$. Similarly, $\lceil r \rceil$ is the smallest integer not smaller than r . For $a, b \in \mathbb{R} \cup \{-\infty, \infty\}$, $[a, b] := \{x \in \mathbb{R} \mid a \leq x \leq b\}$ is the *interval* of real numbers between a and b . The set of *intervals* will be denoted by $\mathcal{I}$. Note that $\mathcal{I}$ is closed under finite intersection. An interval $I = [a, b]$ is said to be *bounded* if neither a nor b are in $\{-\infty, \infty\}$.

Let A and B be two arbitrary sets. Size of A is denoted by $|A|$. The power set of A is represented by 2^A . The Cartesian product of A and B is represented by $A \times B$. For a tuple $p := (a, b)$, elements a and b are represented by $\text{fst}(p)$ and $\text{snd}(p)$, respectively. If $A, B \subseteq \mathbb{R}$ then $A + B := \{a + b \mid a \in A \wedge b \in B\}$. The set of functions from A to B is represented by B^A or $A \rightarrow B$. For a function $f \in A \rightarrow B$, $\text{dom}(f)$ denotes the domain of f . For any two functions $f, g \in A \rightarrow \mathbb{R}$, the function $f - g \in A \rightarrow \mathbb{R}$ maps x to $f(x) - g(x)$. For any $C \subseteq A$, $f[C \leftarrow 0] \in A \rightarrow \mathbb{R}$ maps x to 0 if $x \in C$ and maps x to $f(x)$ if $x \notin C$. Consider $D \subseteq 2^B$. If D is closed under intersection, and $f, g \in D^A$, $f \cap g \in A \rightarrow D$ is the function $(f \cap g)(x) := f(x) \cap g(x)$. For functions $f \in A \rightarrow D$ and $g \in A \rightarrow B$, we say $g \in f$ iff $\forall x \in A. g(x) \in f(x)$.

For a set A , ε denotes the empty sequence, A^n (for $n \in \mathbb{N}$) the set of sequences of length n , A^* the set of finite sequences, A^ω the set of infinite sequences, and $A^\infty = A^* \cup A^\omega$ the set of finite and infinite sequences. We will often abuse notation and refer to both A and A^1 (sequences of length 1) as A . For two sets of sequences X and Y , XY denotes the set of sequences formed by concatenating a sequence from X with a sequence from Y . For a sequence σ , σ_i will denote the i^{th} element in the sequence σ ; the first element of the sequence is assumed to have index 0.

For any two points $p, p' \in \mathbb{R}^A$ and two subsets $P, P' \subseteq \mathbb{R}^A$ we define $d_\infty(p, p') := \max\{|p(a) - p'(a)| \mid a \in A\}$, $d_\infty(p, P') := \inf\{d_\infty(p, p') \mid p' \in P'\}$, and $d_\infty(P, P') := \max\{\sup_{p \in P} d_\infty(p, P'), \sup_{p' \in P'} d_\infty(p', P)\}$.

2.1 Timed Automata

In this section we define timed automata and its semantics under different kinds of perturbation. For an introduction to the theory of timed automata see [2].

Definition 1. A *timed automaton* $\mathcal{H}$ is a tuple $(Q, X, Q^{\text{init}}, I, E)$, where

- Q is a finite non-empty set of (discrete) locations.
- X is a finite non-empty set of clocks.

- $\mathbf{Q}^{\text{init}} \subseteq \mathbf{Q}$ is a set of initial locations.
- $\mathbf{I} \in \mathbf{Q} \rightarrow \mathcal{I}^{\mathbf{X}}$ maps each location q to a rectangular set as the invariant of clocks in q .
- $\mathbf{E}$ is a finite set of edges. Each edge $e \in \mathbf{E}$ itself is a tuple (s, d, l, g, r) in which
 - $s, d \in \mathbf{Q}$ are source and destination locations, respectively.
 - $l \subseteq \mathbf{E}$ is the label of e ,
 - $g \in \mathcal{I}^{\mathbf{X}}$ is a rectangular set, the guard of e .
 - $r \subseteq \mathbf{X}$ is the set of clocks that are reset to 0 after taking e .

We write Se , De , Le , Ge , and Re to denote the source, destination, label, guard and reset associated with the edge e , respectively.

A valuation function $\nu \in \mathbb{R}_{\geq 0}^{\mathbf{X}}$ assigns a non-negative value to each clock of $\mathcal{H}$. We denote the set of all valuations by $\mathbf{V}_{\mathcal{H}}$. Furthermore, let $\mathbf{M} \in \mathbb{R}_{\geq 0}$ be the maximum constant appearing in the specification of guards of edges and invariants of locations in $\mathcal{H}$. We refer to the different elements of $\mathcal{H}$, by adding $\mathcal{H}$ as a subscript. For example, $\mathbf{X}_{\mathcal{H}}$ is the set of clocks of $\mathcal{H}$. But we may omit the subscript whenever it is clear from the context. Also, we denote $\mathbf{I}(q)(x)$ and $\mathbf{G}(e)(x)$ respectively by $\mathbf{I}(q, x)$ and $\mathbf{G}(e, x)$. Finally, a timed automaton is said to be bounded iff all intervals specifying the invariants are bounded.

For a timed automaton $\mathcal{H}$, we consider two types of perturbations. By $\mathcal{H}_{\delta}^{\epsilon}$, we refer to the semantics where guards are enlarged by δ and clocks are allowed to drift by ϵ . On the other hand, $\mathcal{H}_{+\delta}^{\epsilon}$ refers to the semantics where clocks are allowed to drift by ϵ but only positive guards are perturbed by δ . Before defining these semantics, we introduce some notation. For an interval $I = [a, b]$ and $\delta \in \mathbb{R}_{\geq 0}$, $I_{\delta} := [a - \delta, b + \delta]$. On the other hand, $I_{+\delta} := [c, d]$, where

$$c = \begin{cases} a - \delta & \text{if } a > 0 \\ a & \text{otherwise} \end{cases} \quad d = \begin{cases} b + \delta & \text{if } b > 0 \\ b & \text{otherwise} \end{cases}$$

Definition 2. For any timed automaton $\mathcal{H}$ and $\epsilon, \delta \in \mathbb{R}_{\geq 0}$, the semantics of $\mathcal{H}_{\eta}^{\epsilon}$, where $\eta \in \{\delta, +\delta\}$, is defined to be the transition system $(\mathbf{S}, \Sigma, \rightarrow, \mathbf{S}^{\text{init}})$ where

- $\mathbf{S} := \mathbf{Q} \times \mathbf{V}$,
- $\Sigma := \mathbf{E} \cup \mathbb{R}_{\geq 0}$,
- $\mathbf{S}^{\text{init}} := \{(q, 0^{\mathbf{X}}) \mid q \in \mathbf{Q}^{\text{init}} \wedge 0^{\mathbf{X}} \in \mathbf{I}_{\eta}(q)\}$, and
- $\rightarrow := \rightarrow_1 \cup \rightarrow_2$ where
 - $\rightarrow_1$ is the set of continuous transitions and for all $t \in \mathbb{R}_{\geq 0}$ we have $(q, \nu) \xrightarrow{t}_1 (q', \nu')$ iff 1. $q = q'$, 2. $\nu, \nu' \in \mathbf{I}_{\eta}(q)$, and 3. $(\nu' - \nu) \in [t(1 - \epsilon), t(1 + \epsilon)]^{\mathbf{X}}$
 - $\rightarrow_2$ is the set of discrete transitions and for any $e \in \mathbf{E}$ we have $(q, \nu) \xrightarrow{e}_2 (q', \nu')$ iff 1. $q = \text{Se}$, 2. $q' = \text{De}$, 3. $\nu \in \mathbf{I}_{\eta}(q) \cap \mathbf{G}_{\eta}(e)$, 4. $\nu' = \nu[\text{Re} \leftarrow 0]$, and 5. $\nu' \in \mathbf{I}_{\eta}(q')$.

When either ϵ or δ is 0, we will drop it from the superscript/subscript. In particular, for $\epsilon, \delta \in \mathbb{R}_{+}$, $\mathcal{H}^{\epsilon}$ refers to $\mathcal{H}_{\delta}^{\epsilon}$, $\mathcal{H}_{\delta}$ refers to $\mathcal{H}_{\delta}^0$, and $\mathcal{H}_{+\delta}$ is $\mathcal{H}_{+\delta}^0$. Finally, we will abuse notation and use $\mathcal{H}$ to refer to both the timed automaton and its standard semantics $\mathcal{H}_{\delta}^0$.

Consider a timed automaton $\mathcal{H}$, $\epsilon, \delta \in \mathbb{R}_{\geq 0}$, and $\eta \in \{\delta, +\delta\}$. Let $\mathcal{T}$ be $\mathcal{H}_{\eta}^{\epsilon}$. A trajectory of $\mathcal{T}$ is a non-empty sequence of the form $s_0, (t_1, e_1), s_1, (t_2, e_2), s_2, \dots$, where s_i is the state of $\mathcal{T}$ immediately after the i^{th} discrete transition, t_i is the time (after transition $i - 1$) the i^{th} discrete transition is taken, and e_i is the edge of the discrete i^{th} transition. Formally, a trajectory τ of $\mathcal{T}$ starting from state s is a

sequence in $\mathbf{S}((\mathbb{R}_{\geq 0} \times \mathbf{E})^{\mathbf{S}})^{\infty}$ such that $\tau_0 = s$, and for every i (with $2i + 2 \leq \text{length of } \tau$), there is a state s_i such that

$$\tau_{2i} \xrightarrow{\text{fst}(\tau_{2i+1})} \tau_{s_i} \xrightarrow{\text{snd}(\tau_{2i+1})} \tau_{\tau_{2i+2}}.$$

The set of trajectories, finite trajectories, and infinite trajectories of $\mathcal{T}$ is denoted by $\llbracket \mathcal{T} \rrbracket_{\infty}$, $\llbracket \mathcal{T} \rrbracket_{*}$, and $\llbracket \mathcal{T} \rrbracket_{\omega}$, respectively. We will use $\text{first}(\tau)$ to refer to the starting state of trajectory τ (namely, state τ_0), and, if τ is finite, $\text{last}(\tau)$ to the last state in the sequence. Let $\tau \in \llbracket \mathcal{T} \rrbracket_{\infty}$ be a sequence with k elements. Observe that k is either infinite or it is an odd number. The length of a trajectory τ is defined to be the number of discrete steps; it is denoted as $|\tau| = \lfloor k/2 \rfloor$. We define $\text{duration}(\tau) := \sum_{i < |\tau|} \text{fst}(\tau_{2i+1})$, to be the sum of the time taken at each step of the trajectory. A trajectory τ is said to be an execution if $\text{first}(\tau) \in \mathbf{S}^{\text{init}}$, i.e., it starts from an initial state. We denote the set of executions, finite executions, and infinite executions of $\mathcal{T}$ by $\llbracket \mathcal{T} \rrbracket_{\infty}^0$, $\llbracket \mathcal{T} \rrbracket_{*}^0$, and $\llbracket \mathcal{T} \rrbracket_{\omega}^0$, respectively. For any $T \in \mathbb{R}$ and trajectory $\tau \in \llbracket \mathcal{T} \rrbracket_{\infty}$, we say that a trajectory τ is T -time-bounded iff $\text{duration}(\tau) \leq T$. Also, $\text{locs}(\tau)$ is the set of locations visited by τ , $\text{inf}(\tau)$ is the set of edges visited infinitely often by τ , and $\text{trace}(\tau)$ is the sequence of edges visited by τ . For a finite trajectory $\tau \in \llbracket \mathcal{T} \rrbracket_{*}$ and a trajectory $\tau' = s\sigma \in \llbracket \mathcal{T} \rrbracket_{\infty}$ starting from state $s = \text{last}(\tau)$, concatenation of τ and τ' , denoted as $\tau \sim \tau'$, is the trajectory/sequence $\tau\sigma$. Finally, we define $\text{reach}(\mathcal{H}_{\delta}^{\epsilon}) := \{\nu \mid \exists \tau \in \llbracket \mathcal{H}_{\delta}^{\epsilon} \rrbracket_{*}^0. \nu = \text{last}(\tau)\}$, $\text{limreach}^{\epsilon}(\mathcal{H}) := \bigcap_{\epsilon \in \mathbb{R}_{+}} \text{reach}(\mathcal{H}^{\epsilon})$, $\text{limreach}_{\delta}(\mathcal{H}) := \bigcap_{\delta \in \mathbb{R}_{+}} \text{reach}(\mathcal{H}_{\delta})$, $\text{limreach}_{\delta}^{\epsilon}(\mathcal{H}) := \bigcap_{\epsilon \in \mathbb{R}_{+}} \bigcap_{\delta \in \mathbb{R}_{+}} \text{reach}(\mathcal{H}_{\delta}^{\epsilon})$.

An equivalence relation, called *region equivalence* plays a critical role in the reachability analysis of timed automata [2]. It is an equivalence on states, and equivalence classes of this relation are called *regions*. Region equivalence is defined as follows. Consider a timed automaton $\mathcal{H}$ with clocks $\mathbf{X}$, locations $\mathbf{Q}$, and maximum constant $\mathbf{M}$. Let $\mathcal{T}$ be the transition system of $\mathcal{H}$ with states $\mathbf{S}$. Two states $(q_1, \nu_1), (q_2, \nu_2) \in \mathbf{S}$ are region equivalent, $(q_1, \nu_1) \sim (q_2, \nu_2)$, iff

1. Locations q_1 and q_2 are the same, i.e., $q_1 = q_2$,
2. For every $x \in \mathbf{X}$, either $\lfloor \nu_1(x) \rfloor = \lfloor \nu_2(x) \rfloor$ or both $\nu_1(x) > \mathbf{M}$ and $\nu_2(x) > \mathbf{M}$,
3. For every $x \in \mathbf{X}$, if $\nu_1(x) \leq \mathbf{M}$ then $\langle \nu_1(x) \rangle = 0$ iff $\langle \nu_2(x) \rangle = 0$, and
4. For every $x, y \in \mathbf{X}$, if $\nu_1(x) \leq \mathbf{M}$ and $\nu_2(y) \leq \mathbf{M}$ then $\langle \nu_1(x) \rangle \leq \langle \nu_1(y) \rangle$ iff $\langle \nu_2(x) \rangle \leq \langle \nu_2(y) \rangle$.

The number of equivalence classes of $\sim$, or regions, is bounded by $|\mathbf{Q}| \cdot |\mathbf{X}|! \cdot 2^{|\mathbf{X}|} \cdot (2\mathbf{M} + 2)^{|\mathbf{X}|}$; thus, the number of regions is linear in $|\mathbf{Q}|$ and $\mathbf{M}$ and is singly exponential in $|\mathbf{X}|$. For a region r , $[r]$ will be the smallest (topologically) closed set containing r , and for a state s , $[s]$ will be the topological closure of the region containing s ; thus, if $s \in r$ then $[s] = [r]$. Finally, for any two states $s_1 := (q_1, \nu_1)$ and $s_2 := (q_2, \nu_2)$ we define $\mathbf{d}_{\infty}(s_1, s_2)$ to be ∞ if $q_1 \neq q_2$ and $\mathbf{d}_{\infty}(\nu_1, \nu_2)$ otherwise. The importance of regions is captured in the following proposition that says that region equivalence is a bisimulation.

Proposition 3. If τ is a finite trajectory starting at state s and $s' \sim s$ then there is a finite trajectory τ' starting from s' such that $\text{trace}(\tau) = \text{trace}(\tau')$ and $\text{last}(\tau) \sim \text{last}(\tau')$.

Proposition 3 says the quotient of $\mathcal{T}$ with respect to $\sim$ is well defined. This quotient is called the *region graph* defined as $\mathcal{G} := (V, E)$, where V is the set of regions, and for $\sigma \in \mathbf{E} \cup \{\text{time}\}$, $(r_1, \sigma, r_2) \in E$ iff $\exists s_1 \in r_1, s_2 \in r_2 \bullet$ if $\sigma \in \mathbf{E}$, $s_1 \xrightarrow{\sigma} s_2$ and if $\sigma = \text{time}$, then $r_1 \neq r_2$ and there is

$t \in \mathbb{R}_{\geq 0}$ such that $s_1 \xrightarrow{t} s_2$. For any set of regions A , we define $\text{reach}(\mathcal{G}, A)$ to be the set of nodes in $\mathcal{G}$ reachable from A . A path π in the region graph is called *progress cycle* iff the first and last nodes in π are the same and every clock in $\mathbf{X}$ is reset at least once along the edges of π . Finally, a timed automaton $\mathcal{H}$ is said to satisfy the *progress cycle assumption (PCA)* if every cycle in its region graph $\mathcal{G}$ is a progress cycle.

We conclude this section by defining the main verification problems that we consider in this paper. Recall that the classical safety and ω -regular model checking problems can be defined as follows:

Problem 4. Let $\mathcal{H}$ be a timed automaton with only integer constants, $Q \subseteq \mathbb{Q}_{\mathcal{H}}$ be an arbitrary set of locations, and $B \subseteq \mathbf{E}_{\mathcal{H}}$ be an arbitrary subset of edges. $\mathcal{H}$ is said to satisfy the safety property $\Box Q$ (denoted $\mathcal{H} \models \Box Q$) if no control location outside Q is reachable, that is, it is not the case that there is $q \notin Q$ and $\nu \in \mathbf{V}_{\mathcal{H}}$ such that $(q, \nu) \in \text{reach}(\mathcal{H})$. Also, $\mathcal{H}$ is said to satisfy ω -regular property $\Box \Diamond B$ (denoted by $\mathcal{H} \models \Box \Diamond B$), iff $\forall \tau \in \llbracket \mathcal{H} \rrbracket_{\omega}^0 \bullet \inf(\tau) \cap B \neq \emptyset$.

The safety and ω -regular model checking problems are that given timed automaton $\mathcal{H}$, sets $Q \subseteq \mathbb{Q}_{\mathcal{H}}$ and $B \subseteq \mathbf{E}_{\mathcal{H}}$, determine if $\mathcal{H} \models \Box Q$ and $\mathcal{H} \models \Box \Diamond B$, respectively.

Note that the safety and ω -regular model checking problems as stated are a special case of general safety and ω -regular verification. However, the general regular safety and ω -regular property verification problems can be reduced to these special cases by taking a cross product of $\mathcal{H}$ and an appropriate property automaton [4]. In this paper we consider robust versions of these classical problems.

Problem 5. Let $\mathcal{H}$ be a timed automaton with only integer constants, and $B \subseteq \mathbf{E}_{\mathcal{H}}$ be an arbitrary subset of edges. The robust ω -regular model checking problem asks if there are $\epsilon, \delta \in \mathbb{R}_+$ such that $\mathcal{H}_{\delta}^{\epsilon} \models \Box \Diamond B$.

Two variants of the above problem that we will also consider are as follows. The robust ω -regular model checking problem when only guards are enlarged asks if there is $\delta \in \mathbb{R}_+$ such that $\mathcal{H}_{\delta} \models \Box \Diamond B$. The robust ω -regular model checking problem when only clocks drift asks if there is $\epsilon \in \mathbb{R}_+$ such that $\mathcal{H}^{\epsilon} \models \Box \Diamond B$.

Robust versions of the safety verification problem are defined similarly; the formal definition is skipped in the interest of space.

3. ROBUST ω -REGULAR MODEL CHECKING

In this section, we present our results for the robust ω -regular model checking problem when both clocks and guards are perturbed and the robust ω -regular model checking problem when only clocks are perturbed. Recall that the robust ω -regular model checking problem when only guards are enlarged was solved in [8].

3.1 Robust ω -regular model checking when both guards and clocks are perturbed

Our main result in this section is that robust ω -regular model checking problem when both guards and clocks are perturbed is equivalent to the robust ω -regular model checking problem when only guards are perturbed. This is formalized in the following theorem.

Theorem 6. For any timed automaton $\mathcal{H}$ and B a subset of $\mathbf{E}_{\mathcal{H}}$,

$$(\exists \epsilon, \delta \in \mathbb{R}_+ \bullet \mathcal{H}_{\delta}^{\epsilon} \models \Box \Diamond B) \Leftrightarrow (\exists \delta' \in \mathbb{R}_+ \bullet \mathcal{H}_{\delta'} \models \Box \Diamond B)$$

Note that the implication from left to right is trivial, since if $\mathcal{H}_{\delta}^{\epsilon} \models \Box \Diamond B$ then by assigning δ to δ' , we obtain $\mathcal{H}_{\delta'} \models \Box \Diamond B$ as well. The non-trivial part of the proof lies in showing the other direction. It requires the robustness under perturbed guards to be transferred to that under clock drifts and perturbed guards, which is facilitated by the following lemma.

Lemma 7. For any timed automaton $\mathcal{H}$, $\epsilon \in (0, \frac{1}{4})$, $\gamma \in (0, \frac{M}{2})$, and $\tau \in \llbracket \mathcal{H}_{\gamma}^{\epsilon} \rrbracket_{\infty}^0$, there is $\tau' \in \llbracket \mathcal{H}_{2M\epsilon+\gamma} \rrbracket_{\infty}^0$, such that τ and τ' have the exact same sequence of discrete and continuous transitions. Here, M is the largest constant appearing in the constraints of $\mathcal{H}$.

Proof. Fix $\epsilon \in (0, \frac{1}{4})$, $\gamma \in (0, \frac{M}{2})$, and $\tau \in \llbracket \mathcal{H}_{\gamma}^{\epsilon} \rrbracket_{\infty}^0$. Let $\delta := 2\epsilon M + \gamma$. Obviously, the set of edges that are enabled initially in $\mathcal{H}_{\delta}^{\epsilon}$ corresponds to a subset of the edges that are initially enabled in $\mathcal{H}_{\delta}$. We show the same is true at any step in τ . For any variable $x \in \mathbf{X}$ if valuation ν_x satisfies guard g at the time of taking edge e in $\mathcal{H}_{\gamma}^{\epsilon}$, there are two cases:

1. $g := x \triangleright c$ for some integer $c \in \{0, \dots, M\}$ and $\triangleright \in \{>, \geq\}$. Time taken by x to reach its current value since its last reset (or since the beginning of time if x has never been reset) is at least $\frac{c-\gamma}{1+\epsilon}$. We show that this time is big enough for the enlarged guard $x \triangleright c - \delta$ to be satisfied when x evolves without clock drifts.

$$\begin{aligned} (M \triangleright c) & \Rightarrow \\ (2M + \gamma + 2\epsilon M \triangleright c) & \Rightarrow \\ (0 \triangleright -2\epsilon M + c\epsilon - \gamma\epsilon - 2\epsilon^2 M) & \Rightarrow \\ (c - \gamma \triangleright c - \gamma - 2\epsilon M + c\epsilon - \gamma\epsilon - 2\epsilon^2 M) & \Rightarrow \\ (c - \gamma \triangleright (1 + \epsilon)(c - \gamma - 2\epsilon M)) & \Rightarrow \\ \left(\frac{c - \gamma}{1 + \epsilon} \triangleright c - \gamma - 2\epsilon M = c - \delta \right) & \end{aligned}$$

2. $g := x \triangleleft c$ for some integer $c \in \{0, \dots, M\}$ and $\triangleleft \in \{<, \leq\}$. Time taken by x to reach its current value, since its last reset (or since the beginning of time if x has never been reset) is at most $\frac{c+\gamma}{1-\epsilon}$. We show that this time is small enough for the enlarged guard $x \triangleleft c + \delta$ to be satisfied when there are no clock drifts. Again,

$$\begin{aligned} (c \leq M) & \Rightarrow \\ \left(c + \frac{M}{2} + \frac{M}{2} \leq 2M \right) & \Rightarrow \\ (c + 2\epsilon M + \gamma \triangleleft 2M) & \Rightarrow \\ (0 \triangleleft 2\epsilon M - c\epsilon - 2\epsilon^2 M - \epsilon\gamma) & \Rightarrow \\ (c + \gamma \triangleleft c + \gamma + 2\epsilon M - c\epsilon - 2\epsilon^2 M - \epsilon\gamma) & \Rightarrow \\ (c + \gamma \triangleleft (1 - \epsilon)(c + 2\epsilon M + \gamma)) & \Rightarrow \\ \left(\frac{c + \gamma}{1 - \epsilon} \triangleleft c + 2\epsilon M + \gamma = c + \delta \right) & \end{aligned}$$

Note that variables are reset at the exact same times in $\tau \in \llbracket \mathcal{H}_{\gamma}^{\epsilon} \rrbracket_{\infty}^0$ and $\tau \in \llbracket \mathcal{H}_{\delta} \rrbracket_{\infty}^0$. So whenever a variable is reset in $\tau \in \llbracket \mathcal{H}_{\gamma}^{\epsilon} \rrbracket_{\infty}^0$ at time t , the same variable is reset in $\tau \in \llbracket \mathcal{H}_{\delta} \rrbracket_{\infty}^0$ at time t . This makes the values of x in two executions 0 at time t . $\square$

The right to left implication of Theorem 6 follows directly from Lemma 7. If $\mathcal{H}_{\delta'} \models \Box \Diamond B$, then by choosing any $\epsilon \in (0, \frac{1}{4})$ and $\delta \in (0, \frac{M}{2})$ such that $2M\epsilon + \delta < \delta'$, we know any

execution in $\llbracket \mathcal{H}_\delta^\epsilon \rrbracket_\omega^0$ has a corresponding execution in $\llbracket \mathcal{H}_{\delta'} \rrbracket_\omega^0$ with the same sequence of discrete transitions. Lemma 7 also implies $\text{limreach}_\delta^\epsilon(\mathcal{H})$ is subset of $\text{limreach}_\delta(\mathcal{H})$ and therefore they are equal even when the progress cycle assumption does not hold. This is formalized in the Corollary 8.

Corollary 8. *For any bounded timed automaton $\mathcal{H}$, we have $\text{limreach}_\delta^\epsilon(\mathcal{H}) \subseteq \text{limreach}_\delta(\mathcal{H})$.*

Proof. $\text{limreach}_\delta(\mathcal{H})$ is taken to be the intersection of the reach set of $\mathcal{H}_\delta$ for all δ . Since the guards are perturbed in $\mathcal{H}_\delta$, we can (without loss of generality) assume that the intervals are closed in $\mathcal{H}_\delta$ when defining $\text{limreach}_\delta(\mathcal{H})$ [11]. Thus, $\text{reach}(\mathcal{H}_\delta)$ is a closed set, for any $\delta \in \mathbb{R}_+$. Let $s \in \text{limreach}_\delta^\epsilon(\mathcal{H})$ be an arbitrary state. For any $\delta \in \mathbb{R}_+$, we will show $s \in \text{reach}(\mathcal{H}_\delta)$, thus establishing the claim. Since $\text{reach}(\mathcal{H}_\delta)$ is closed, we can establish this by showing that $d_\infty(s, \text{reach}(\mathcal{H}_\delta))$ is arbitrarily small, or more precisely, for any $\kappa \in \mathbb{R}_+$, $d_\infty(s, \text{reach}(\mathcal{H}_\delta)) < \kappa$.

Let us fix $\kappa \in \mathbb{R}_+$. Pick $\epsilon \in (0, \frac{1}{4})$ and $\gamma \in (0, \frac{M}{2})$ such that $2M\epsilon + \gamma < \delta$ and $2M\epsilon < \kappa$. Since $s \in \text{reach}(\mathcal{H}_\gamma^\epsilon)$ we know there is an execution $\tau \in \llbracket \mathcal{H}_\gamma^\epsilon \rrbracket_\omega^0$ such that $\text{last}(\tau) = s$. By Lemma 7 we know there is an execution $\tau' \in \llbracket \mathcal{H}_{2M\epsilon+\gamma} \rrbracket_\omega^0 \subseteq \llbracket \mathcal{H}_\delta \rrbracket_\omega^0$ with the same sequence of discrete and continuous transitions as τ . Since $\mathcal{H}$ is bounded and $\epsilon < \frac{1}{2}$, every variable is reset at least once every $2M$ units of time. Since the clocks are drifted by at most ϵ in $\mathcal{H}_\gamma^\epsilon$ and variables are reset at the exact same times in τ and τ' , $d_\infty(s, \text{last}(\tau')) \leq 2M\epsilon < \kappa$. Therefore, $d_\infty(s, \text{reach}(\mathcal{H}_\delta)) < \kappa$. $\square$

To obtain an algorithm for robust ω -regular model checking problem when both guards and clocks are perturbed, we resort to the results in [8] for robust ω -regular model checking problem when only guards are perturbed. The algorithm in [8] provides a computable value δ_0 , such that the $\mathcal{H}_{\delta_0} \models \Box\Diamond B$ iff there is $\delta \in \mathbb{R}_+$ such that $\mathcal{H}_\delta \models \Box\Diamond B$. We recall this result from [8].

Theorem 9 (Based on Lemma 11 and Theorem 3 in [8]). *Let $\mathcal{H}$ be a timed automaton, W be its number of regions, and B be a subset of $\mathbf{E}_\mathcal{H}$. Then*

1. $\forall \delta \in \mathbb{R}_+, \tau \in \llbracket \mathcal{H}_{\delta_0} \rrbracket_\omega^0, \exists \tau' \in \llbracket \mathcal{H}_\delta \rrbracket_\omega^0$ such that $\inf(\tau) = \inf(\tau')$
2. $\exists \delta \in \mathbb{R}_+ \cdot \mathcal{H}_\delta \models \Box\Diamond B \Leftrightarrow \mathcal{H}_{\delta_0} \models \Box\Diamond B$

where $\delta_0 := \frac{1}{2}(8|X|^2(W+1))^{-1}$ if $\mathcal{H}$ satisfies the progress cycle assumption, otherwise

$$\delta_0 := \frac{1}{2}(5(W+1)|X|^3(2|Q|(|X|!4^{|X|}+4)^2)^{-1}.$$

Proof. The second result follows immediately from the first one. As far as the first result is concerned, the only difference when compared to the observations in [8], is that here $\inf(\cdot)$ returns the set of edges, as opposed to the set of locations, that are visited infinitely often. This is not a problem since the proof in [8] essentially obtains τ' from τ by 1. repeating some subexecutions of τ a finite number of times (duration of continuous transitions may change during this step), and 2. repeating the previous step for a finite number of times. $\square$

In order to solve robust ω -regular model checking when both guards and clocks are perturbed, our algorithm first checks $\mathcal{H}_{\delta_0} \models \Box\Diamond B$ where δ_0 is introduced by Theorem 9. If $\mathcal{H}_{\delta_0} \not\models \Box\Diamond B$ then clearly $\mathcal{H}$ does not robustly satisfy $\Box\Diamond B$ when both guards and clocks are perturbed. On the other hand, if $\mathcal{H}_{\delta_0} \models \Box\Diamond B$ then using Lemma 7 we know $\mathcal{H}_{\delta_1}^\epsilon \models \Box\Diamond B$ for any $\delta_1, \epsilon_1 \in \mathbb{R}_+$ that satisfy $2M\epsilon_1 + \delta_1 < \delta_0$.

Figure 1: $\exists \epsilon \in \mathbb{R}_+ \cdot \mathcal{H}^\epsilon \models \Box\Diamond B$ does not always imply $\exists \delta \in \mathbb{R}_+ \cdot \mathcal{H}_\delta \models \Box\Diamond B$

Theorem 10. *Let $\mathcal{H}$ be a timed automaton and B be a subset of $\mathbf{E}_\mathcal{H}$. Let δ_0 be as defined in Theorem 9. Then, for any, ϵ_1, δ_1 such that $2M\epsilon_1 + \delta_1 \leq \delta_0$, the following holds:*

$$(\exists \epsilon, \delta \in \mathbb{R}_+ \cdot \mathcal{H}_\delta^\epsilon \models \Box\Diamond B) \Leftrightarrow (\mathcal{H}_{\delta_0} \models \Box\Diamond B) \Leftrightarrow (\mathcal{H}_{\delta_1}^{\epsilon_1} \models \Box\Diamond B)$$

Since the robust ω -regular model checking problem when only guards are perturbed is PSPACE-complete, from Theorem 10, we obtain the following:

Corollary 11. *The robust ω -regular model checking problem when both guards and clocks are perturbed is PSPACE-complete.*

While robust ω -regular model checking problem when both guards and clocks are perturbed and when only guards are perturbed are equivalent (Theorem 6), the same is surprisingly not true for the case when only clocks are drifted. This is demonstrated by our next example.

Example 12. *In this example, we show that*

$$(\exists \epsilon \in \mathbb{R}_+ \cdot \mathcal{H}^\epsilon \models \Box\Diamond B) \not\Leftrightarrow (\exists \delta \in \mathbb{R}_+ \cdot \mathcal{H}_\delta \models \Box\Diamond B)$$

Consider the timed automaton $\mathcal{H}$ in Figure 1. It starts at location 0 with $x = y = 0$ and both the variables evolve at rate 1. Note that e_1 , the self loop on location 0, can only be taken when $x = 0$, that is, it is feasible only at time 0. Hence, the edge e_2 from location 0 to location 1 can never be taken, since once time passes both x and y will have non-zero values and the value of x can never be reset.

Taking $B := \{e_1\}$, consider the ω -regular property $\Box\Diamond B$. Note that the only infinite execution τ of the timed automaton is the one which traverses e_1 repeatedly without time elapsing. This satisfies the condition $\Box\Diamond B$. Moreover, even if we allow clock drifts, we will still have only one infinite execution, namely, τ . Hence, $\mathcal{H}^\epsilon \models \Box\Diamond B$ (for any $\epsilon \in [0, 1)$).

However, we argue that for any δ , $\mathcal{H}_\delta \not\models \Box\Diamond B$, since, there will be infinite trajectories that visit location 1 and execute e_3 , the self loop on location 1, repeatedly. To see this, note that a δ enlargement of the guard $x \leq 0$ will lead to $x \leq \delta$. Hence, by repeatedly taking e_1 every δ units of time, the value $x = 0$ and $y \in [1 - \delta, 1 + \delta]$ can be reached. Hence, the edge to location 1 can be taken.

Observe that the same example works if we bound guards and invariants. Furthermore, this example also shows that $\text{limreach}^\epsilon(\mathcal{H}) \neq \text{limreach}_\delta(\mathcal{H})$, a fact we mentioned in the introduction.

3.2 Robust ω -regular model checking when only clocks are perturbed

In Example 12, we showed that even if there exists an $\epsilon \in \mathbb{R}_+$ such that $\mathcal{H}^\epsilon \models \Box\Diamond B$, there may not exist a $\delta \in \mathbb{R}_+$ such that $\mathcal{H}_\delta \models \Box\Diamond B$. However, we show that the implication holds when we consider a weaker notion of guard perturbations.

Theorem 13. For any timed automaton $\mathcal{H}$ and B a subset of $\mathbf{E}$,

$$\left(\exists \epsilon \in \mathbb{R}_+ \cdot \mathcal{H}^\epsilon \models \Box \Diamond B \right) \Leftrightarrow \left(\exists \delta \in \mathbb{R}_+ \cdot \mathcal{H}_{+\delta} \models \Box \Diamond B \right)$$

The crux of the proof of Theorem 13 lies in the following lemma which establishes the connection between the trace language of automata under clock drifts and the trace language of automata when positive guard constants are perturbed.

Lemma 14. For a timed automaton $\mathcal{H}$ with maximum constant M and $\epsilon \in (0, \frac{1}{2})$ the following holds:

$$\text{trace}(\llbracket \mathcal{H}_{+\frac{\epsilon}{2}} \rrbracket_\infty^0) \subseteq \text{trace}(\llbracket \mathcal{H}^\epsilon \rrbracket_\infty^0) \subseteq \text{trace}(\llbracket \mathcal{H}_{+2M\epsilon} \rrbracket_\infty^0)$$

Proof. $\mathcal{H}^\epsilon$ is an initialized rectangular automaton. In [13], Henzinger *et al.* describe a transformation from 1. an initialized rectangular automaton $\mathcal{H}^\epsilon$ into an initialized multirate hybrid automaton $\mathcal{H}'$ that is trace equivalent with $\mathcal{H}^\epsilon$, and 2. $\mathcal{H}'$ to a timed automaton $\mathcal{H}''$ which is bisimilar to $\mathcal{H}'$.

Initialized multirate hybrid automaton $\mathcal{H}'$ is obtained from $\mathcal{H}$ by replacing every variable $x \in X_{\mathcal{H}}$ by two variables x_l and x_u . Initially all variables in $\mathcal{H}'$ are zero. $\mathcal{H}$ and $\mathcal{H}'$ have the same set of (initial) locations and none of them has invariant in $\mathcal{H}'$. For any location $q \in Q_{\mathcal{H}}$ and variable $x \in X_{\mathcal{H}}$, $\text{Flow}_{\mathcal{H}'}(q, x_l) = 1 - \epsilon$ and $\text{Flow}_{\mathcal{H}'}(q, x_u) = 1 + \epsilon$ (recall that $\text{Flow}(q, x)$ is rate at which variable x changes in location q). There is a bijection between edges of $\mathcal{H}$ and $\mathcal{H}'$. For any edge $e \in E_{\mathcal{H}}$ there is an edge $e' \in E_{\mathcal{H}'}$ such that 1. e' has the same source and destination as e , 2. For any $x \in X_{\mathcal{H}}$ we have $x \in R_{\mathcal{H}}e \Leftrightarrow x_l, x_u \in R_{\mathcal{H}'}e'$, 3. For any $x \in X_{\mathcal{H}}$ and constant $c \in \mathbb{R}$ we have $(x \leq c) \in G_{\mathcal{H}}e \Leftrightarrow (x_l \leq c) \in G_{\mathcal{H}'}e'$ and $(x \geq c) \in G_{\mathcal{H}}e \Leftrightarrow (x_u \geq c) \in G_{\mathcal{H}'}e'$.

Timed automaton $\mathcal{H}''$ is obtained from $\mathcal{H}'$ by 1. setting $\text{Flow}_{\mathcal{H}''}(q, x_l) = \text{Flow}_{\mathcal{H}'}(q, x_u) = 1$ for all $q \in Q_{\mathcal{H}}$ and $x \in X_{\mathcal{H}}$. 2. every guard $x_l \leq c$ is replaced by $x_l \leq \frac{c}{1-\epsilon}$, and 3. every guard $x_u \geq c$ is replaced by $x_u \geq \frac{c}{1+\epsilon}$. Note that x_l and x_u are both reset at the exact same times to zero, they have the same initial value and reset to the same value. Therefore, one can merge these two variables into one and combine their guards together. This is possible since $\mathcal{H}$ is a timed automaton.

If $c = 0$ then $\frac{c}{1-\epsilon} = \frac{c}{1+\epsilon} = 0$. This corresponds to not changing zero guards. Next we show for $c \geq 1$ the following are true: 1. $c + \epsilon \leq \frac{c}{1-\epsilon} \leq c + 2M\epsilon$, and 2. $c - M\epsilon \leq \frac{c}{1+\epsilon} \leq c - \frac{\epsilon}{2}$.

$$\begin{aligned} \left(\frac{c}{1-\epsilon} \leq c + 2M\epsilon \right) &\Leftrightarrow (c \leq c - c\epsilon + 2M\epsilon - 2M\epsilon^2) \Leftrightarrow \\ &\quad (c + 2M\epsilon \leq 2M) \\ \left(c + \epsilon \leq \frac{c}{1-\epsilon} \right) &\Leftrightarrow (c - c\epsilon + \epsilon - \epsilon^2 \leq c) \Leftrightarrow (1 - \epsilon \leq c) \\ \left(\frac{c}{1+\epsilon} \leq c - \frac{\epsilon}{2} \right) &\Leftrightarrow \left(c \leq c + c\epsilon - \frac{\epsilon}{2} - \frac{\epsilon^2}{2} \right) \Leftrightarrow \left(\frac{1}{2} + \frac{\epsilon}{2} \leq c \right) \\ \left(c - M\epsilon \leq \frac{c}{1+\epsilon} \right) &\Leftrightarrow (c + c\epsilon - M\epsilon - M\epsilon^2 \leq c) \Leftrightarrow \\ &\quad (c - M\epsilon \leq M) \end{aligned}$$

This means that if we enlarge guards with $2M\epsilon$, any time a guard is enabled in $\mathcal{H}''$ the same guard is enabled in $\mathcal{H}_{+2M\epsilon}$. Similarly, if we enlarge guards with $\frac{\epsilon}{2}$, any time a guard is enabled in $\mathcal{H}_{+\frac{\epsilon}{2}}$ the same guard is enabled in $\mathcal{H}''$. The proof is complete once we remember that $\mathcal{H}^\epsilon$ and $\mathcal{H}''$ are trace equivalent. $\square$

Next, we present an algorithm for robust ω -regular model checking problem when only positive guards are perturbed.

The main technical observation is that there exists a computable δ_1 such that for all positive guard perturbations $\delta \in (0, \delta_1)$, the set of edges that are visited infinitely often along the executions of $\mathcal{H}_{+\delta_1}$ are contained in that of $\mathcal{H}_{+\delta}$. Therefore, if $\mathcal{H}_{+\delta} \models \Box \Diamond B$ for some δ , then $\mathcal{H}_{+\delta_1} \models \Box \Diamond B$ (the other implication is trivial). These observations are captured in Lemma 15 and Theorem 16. We first need a few definitions.

For any timed automaton $\mathcal{H}$ and any two edges $e_1, e_2 \in \mathbf{E}$, let $\text{succ}(e_1, e_2)$ be a predicate that returns true iff 1. $\text{De}_1 = \text{Se}_2$, and 2. $\forall x \in \text{Re}_1 \cdot (x = 0) \wedge \text{Ge}_2 \not\models \perp$. The second condition means if x is reset by e_1 then predicate $x = 0$ is consistent with the guard of e_2 . Intuitively, $\text{succ}(e_1, e_2)$ is true iff e_1 and e_2 can be merged. When this is the case, we define $e := \text{merge}(e_1, e_2)$ to be the edge $(\text{Se}_1, \text{De}_2, \text{Ge}_1 \wedge g, \text{Re}_1 \cup \text{Re}_2, \text{Le}_1 \cup \text{Le}_2 \cup \{e_1, e_2\})$ where for any guard $(x \bowtie c) \in \text{Ge}_2$ if $x \notin \text{Re}_1$ then $(x \bowtie c) \in g$, otherwise, g does not constrain x . It is easy to see that if $s_1 \xrightarrow{e_1} s_2 \xrightarrow{0} s_3 \xrightarrow{e_2} s_4$, for some $s_1, s_2, s_3, s_4 \in S_{\mathcal{H}}$ then $s_1 \xrightarrow{e} s_4$. Let E^s be the smallest set that contains E and for any two edges $e_1, e_2 \in E^s$ if $\text{succ}(e_1, e_2)$ then $\text{merge}(e_1, e_2) \in E^s$. Let $\mathcal{H}^s$ be same as $\mathcal{H}$ except its set of edges is replaced by E^s and q is an initial location in $\mathcal{H}^s$ iff $q \in Q_{\mathcal{H}}^{\text{init}}$ or $(q', 0^x) \xrightarrow{e} (q, 0^x)$ for some $q' \in Q_{\mathcal{H}}^{\text{init}}$ and $e \in E^s$. Intuitively, q is an initial location in $\mathcal{H}^s$ iff q is an initial location of $\mathcal{H}$ or it can be reached using an execution of 0 duration. Note that $|E^s| < \infty$ therefore $\mathcal{H}^s$ is a well defined timed automaton. Finally, by construction, for any two states $s, s' \in S_{\mathcal{H}} = S_{\mathcal{H}^s}$ and $e \in E^s$ if $s \xrightarrow{e} s'$ and e is a merge of $e_1, \dots, e_n \in E_{\mathcal{H}}$ in the given order (we don't need merge to be associative) then $s_0 \xrightarrow{e_1} s'_1 \xrightarrow{0} s_1 \xrightarrow{e_2} \dots \xrightarrow{0} s'_n \xrightarrow{e_n} s_n$ for some $s_0, \dots, s_n, s'_1, \dots, s'_n \in S_{\mathcal{H}}$ such that $s_0 = s$ and $s_n = s'$. Furthermore, $\text{Le} = \{e_1, \dots, e_n\}$.

Lemma 15. Let $\mathcal{H}$ be a timed automaton and $\delta_1 := \frac{\delta_0}{24}$, where δ_0 is defined as in Theorem 9. For any $\delta \in (0, \delta_1)$ and execution $\tau \in \llbracket \mathcal{H}_{+\delta_1} \rrbracket_\omega^0$ there is execution $\tau' \in \llbracket \mathcal{H}_{+\delta} \rrbracket_\omega^0$ such that $\inf(\tau) = \inf(\tau')$.

Proof. If $\text{duration}(\tau) = 0$ the theorem is trivially true. Therefore, for the rest of the proof assume $\text{duration}(\tau) > 0$.

1. Suppose τ has infinitely many non-zero time transitions. We prove this part for $\delta_1 := \delta_0$ (which is stronger than $\delta_1 := \frac{\delta_0}{24}$). After any non-zero time transition, value of no variable is zero. Therefore, none of the guards of the form $x \leq 0$ are satisfied. Construct an execution $u \in \llbracket \mathcal{H}_{+\delta_1}^s \rrbracket_\omega^0$ from τ by removing all zero time transitions and merging edges that their in-between-time-transitions have been removed. Also, if u does not start with non-zero time transition, remove e_0 the first discrete transition of it. After this we know u starts with a non-zero time transition and $x \leq 0$ is never used in its guards. Theorem 9 only uses edges of u and guards of u are the same in $\mathcal{H}_{+\delta_1}^s$ and $\mathcal{H}_{\delta_1}^s$. Therefore, by Theorem 9 there is an execution $u' \in \llbracket \mathcal{H}_{+\delta}^s \rrbracket_\omega^0$ such that $\inf(u') = \inf(u)$. If we replace any edge e in u that belongs to $E^s \setminus E$ with some sequence of edges from Le and do the same for e_0 (if it was initially removed), we get an execution $\tau'' \in \llbracket \mathcal{H}_{+\delta_1} \rrbracket_\omega^0$. If we do the same to u' , we get an execution $\tau' \in \llbracket \mathcal{H}_{+\delta} \rrbracket_\omega^0$. Furthermore, we know $\inf(\tau') = \inf(\tau'')$. This part is complete once we notice $\inf(\tau) = \inf(\tau'')$ since edges in E^s keep track of set of edges that they visit internally.

2. Next suppose $\tau := \tau^1 \sim \tau^2$ for some $\tau^1 \in [\mathcal{H}_{+\delta_1}]_*^0$ and $\tau^2 \in [\mathcal{H}_{+\delta_1}]_\omega$ such that $\text{duration}(\tau^2) = 0$. Let $Y \subseteq \mathbf{X}$ be the possibly empty set of variables such that for any $x \in Y$ the guard $x \leq 0$ occurs infinitely often in τ . Let $\tau := \tau^3 \sim \tau^4$ for some $\tau^3 \in [\mathcal{H}_{+\delta_1}]_*^0$ and $\tau^4 \in [\mathcal{H}_{+\delta_1}]_\omega$ such that $\text{duration}(\tau^4) = 0$, all variables in Y are always zero in τ^4 , and $x \leq 0$ appears in guards of τ^4 only for $x \in Y$. For any variable $x \in \mathbf{X}$, let $\tau_x^4 \in [\mathcal{H}_{+\delta_1}]_\infty$ be the longest prefix of τ^4 that either never resets x or resets x only on its last step. Also, let g_x be the conjunction of all guards on x in τ_x^4 that are not $x \leq 0$. Furthermore, let $g := \bigwedge_{x \in \mathbf{X}} g_x$, and let g_δ be g enlarged by δ . Note that g_{δ_1} is satisfied by $\text{last}(\tau^3)$. Let q be the last location visited in τ^3 . Add a new location q' to $\mathcal{Q}_\mathcal{H}$ and two new edges e and e' to $\mathbf{E}_\mathcal{H}$. e is from q to q' with guard $g \wedge \bigwedge_{x \in Y} x \leq 0$ that resets no variable, and e' is a self-loop on q' with guard g that also resets no variable. Call the new timed automaton $\mathcal{H}'$. Let $u := u^1 \sim u^2$ for some $u^1 \in [\mathcal{H}'_{+2\delta_1}]_*^0$ and $u^2 \in [\mathcal{H}'_{+2\delta_1}]_\omega$ such that $u^1 := \text{first}(\tau^3) \xrightarrow{\tau^3} \text{last}(\tau^3) \xrightarrow{e} s$ and $u^2 := s_n \xrightarrow{t_n} s'_n \xrightarrow{e'} s_{n+1}$ for some $s, s_0, s'_0, s_1, s'_1, s_2, s'_2, \dots \in \mathbf{S}_{\mathcal{H}'}$ where $s_0 = s$ and $t_n := \frac{\delta_1}{2^{n+1}}$ for all $n \in \mathbb{N}$. Since $\text{duration}(u^2) = \delta_1$ we know $g_{2\delta_1}$ is always satisfied in u^2 and therefore u is a valid execution. Let δ'_0 be the maximum enlargement Theorem 9 allows for $\mathcal{H}'$. Since $\mathcal{H}'$ has only one location more than $\mathcal{H}$ and number of edges does not matter in Theorem 9, we know $2\delta_1 \leq \delta'_0$. So we can use the previous case and find $u' \in [\mathcal{H}'_{+\delta}]_\omega$ such that $\inf(u) = \inf(u') = \{e'\}$. By construction of $\mathcal{H}'$, we know u' can be written as $u'^1 \sim u'^2$ for some $u'^1 \in [\mathcal{H}'_{+\delta}]_*^0$ and $u'^2 \in [\mathcal{H}'_{+\delta}]_\omega$ such that the last edge in u'^1 is e , e is used only once in u'^1 , and the only edge in u'^2 is e' . Inside u' , right before and after taking e , g_δ is satisfied and all variables in Y are zero. Let $\tau' := \tau'^1 \sim \tau'^2$, where τ'^1 is obtained from u'^1 by removing e and the time transition after that, and τ'^2 is obtained from τ'^4 by using the same trace and setting all time transitions equal to zero. We show that $\tau' \in [\mathcal{H}_{+\delta}]_\omega^0$. We know at the end of τ'^1 and everywhere in τ'^2 all variables in Y are zero which is the same in τ^4 . For any variable $x \notin Y$ we know value of x satisfies g_δ at the end of τ'^1 and this value does not change in τ'^2 until x gets reset to 0. During this time it always satisfies g_δ and after it gets reset to 0 it remains 0 both in τ'^2 and τ^4 .

□

Theorem 16. For any timed automaton $\mathcal{H}$ and B a subset of $\mathbf{E}$,

$$(\exists \delta \in \mathbb{R}_+ \cdot \mathcal{H}_{+\delta} \models \Box \Diamond B) \Leftrightarrow (\mathcal{H}_{+\delta_1} \models \Box \Diamond B)$$

where δ_1 is as defined in Lemma 15.

The above observations together give us the following result.

Theorem 17. Let $\mathcal{H}$ be a timed automaton with maximum constant M and let B be a subset of $\mathbf{E}$. Let δ_1 be as defined in Lemma 15, and let $\epsilon_1 := \frac{\delta_1}{2M}$. Then:

$$(\exists \epsilon \in \mathbb{R}_+ \cdot \mathcal{H}^\epsilon \models \Box \Diamond B) \Leftrightarrow (\mathcal{H}_{+\delta_1} \models \Box \Diamond B) \Leftrightarrow (\mathcal{H}^{\epsilon_1} \models \Box \Diamond B)$$

Figure 2: Overview of the results in Section 3. δ_0 is defined in Theorem 9, $\delta_1 := \frac{\delta_0}{24}$, $\epsilon_1 := \frac{\delta_1}{2M}$, and M is the maximum constant appeared in $\mathcal{H}$. Arrows are implications. Implications of dashed arrows are obvious. All problems are in PSPACE-complete.

Proof. Using Theorem 13 and Lemma 15 we only need to show $\exists \epsilon \in \mathbb{R}_+ \cdot \mathcal{H}^\epsilon \models \Box \Diamond B$ implies $\mathcal{H}^{\epsilon_1} \models \Box \Diamond B$. If $\epsilon \geq \epsilon_1$ then we are done. Otherwise, wlog. we can assume $\epsilon < \frac{1}{2}$ and using Lemma 14 we know for any $\tau \in [\mathcal{H}^{\epsilon_1}]_\omega^0$ if τ visits any element of B infinitely often then there is $\tau' \in [\mathcal{H}_{+2M\epsilon_1}]_\omega^0$ that visits the same set infinitely often. Next, using $2M\epsilon_1 \leq \delta_1$ and Theorem 13, there is $\tau'' \in [\mathcal{H}_{+\frac{\epsilon}{2}}]_\omega^0$ that visits the same set of edges infinitely often. Finally, using Lemma 14 we know there is $\tau''' \in [\mathcal{H}^\epsilon]_\omega^0$ that visits the same element in B infinitely often. □

Theorem 17 gives a simple algorithm to solve robust ω -regular model checking problem when only clocks are drifted (compute δ_1 and check that $\mathcal{H}_{+\delta_1} \models \Box \Diamond B$). Since $\mathcal{H}_{+\delta_1}$ is a timed automaton, δ_1 is only exponentially small, and ω -regular model checking problem for timed automaton can be solved in PSPACE, we have a simple upper bound on the complexity. We show that in fact this complexity bound is tight.

Corollary 18. The robust ω -regular model checking problem when only clocks are drifted is PSPACE-complete.

Proof. PSPACE-completeness when guards are enlarged by δ_0 has been established in [8]. Bouyer *et al.* proved for any poly-space bounded Turing machine $\mathcal{A}$ there is a timed automaton $\mathcal{H}$ and property $\Box \Diamond B$ such that $\mathcal{A}$ accepts the zero input iff $\mathcal{H}_{\delta_0} \models \Box \Diamond B$ and iff $\mathcal{H} \models \Box \Diamond B$. With the help of Lemma 7 and using $\epsilon_0 := \frac{\delta_0}{2M}$ we have $(\mathcal{H} \models \Box \Diamond B) \Rightarrow (\mathcal{H}_{\delta_0} \models \Box \Diamond B) \Rightarrow (\mathcal{H}^{\epsilon_0} \models \Box \Diamond B) \Rightarrow (\mathcal{H} \models \Box \Diamond B)$. Therefore, $\mathcal{A}$ accepts the zero input iff $\mathcal{H}^{\epsilon_0} \models \Box \Diamond B$. □

4. COMPUTING LIMIT REACHABLE SETS UNDER CLOCK DRIFTS

In this section, we will prove that $\text{limreach}^\epsilon(\mathcal{H})$ is computable in polynomial space for a bounded timed automaton $\mathcal{H}$. Recall that, a timed automaton $\mathcal{H}$ is bounded if

the invariants of $\mathcal{H}$ are bounded by M , the maximum constant appearing in the constraints of $\mathcal{H}$. Restricting our attention to bounded timed automata is not limiting. It is well known [5, 7], that for any timed automaton $\mathcal{H}$, one can construct a bounded timed automaton $\text{bnd}(\mathcal{H})$ that is weakly bisimilar to $\mathcal{H}$. In fact, the construction also works when $\mathcal{H}$ is perturbed. Thus, given that we show how to compute $\text{limreach}^\epsilon(\mathcal{H})$ for a bounded automaton $\mathcal{H}$, we can compute the set of regions containing states that belong to $\text{limreach}^\epsilon(\mathcal{H}')$ for any (not necessarily bounded) timed automaton $\mathcal{H}'$.

The algorithm to compute $\text{limreach}^\epsilon(\mathcal{H})$ is very similar to the one for timed automata under the progress cycle assumption, and is shown as Algorithm 1. Lines 1 to 3 compute the set $\mathcal{C}$ of all progress cycles whose length are bounded by $W + 2W|\mathbf{X}|$, where W is the number of regions of $\mathcal{H}$. The rest of the algorithm is similar to the approach in [11, 15]; the algorithm first computes the reachable regions assuming there is no perturbation (Line 4). It then iteratively adds closure of initial regions of progress cycles that are not in the current set of reachable regions but have non-empty intersection with it, and recomputes the reachable regions, until a fixpoint is reached. At that point the set J^* is the set of limit reachable states of $\mathcal{H}$ (Line 9).

Algorithm 1: Computing the limit reachable set of a bounded timed automaton

Input : Bounded timed automaton $\mathcal{H}$
Output : $\text{limreach}^\epsilon(\mathcal{H})$

- 1 $\mathcal{G} \leftarrow$ the region graph of $\mathcal{H}$
- 2 $W \leftarrow$ number of nodes in $\mathcal{G}$
- 3 $\mathcal{C} \leftarrow$ the set of progress cycles in $\mathcal{G}$ of length $\leq W + 2W|\mathbf{X}|$
- 4 $J^* \leftarrow \bigcup_{s \in \text{Sinit}} \text{reach}(\mathcal{G}, [s])$
- 5 **while** $\exists p \in \mathcal{C} \bullet [p_0] \not\subseteq J^* \wedge J^* \cap [p_0] \neq \emptyset$ **do**
- 6 $J^* \leftarrow J^* \cup [p_0]$
- 7 $J^* \leftarrow \text{reach}(\mathcal{G}, J^*)$
- 8 **end**
- 9 **return** J^*

The main observation in this section is that J^* is indeed $\text{limreach}^\epsilon(\mathcal{H})$, which is stated next. Later in the section we argue that Algorithm 1 can in fact, be implemented in such a way that it uses only polynomial space.

Theorem 19. *For a timed automaton $\mathcal{H}$, let J^* be the set returned on Line 9 in Algorithm 1. We have:*

$$\text{limreach}^\epsilon(\mathcal{H}) = J^*$$

The proof of $J^* \subseteq \text{limreach}^\epsilon(\mathcal{H})$ follows from the observations in [11]². To show that $\text{limreach}^\epsilon(\mathcal{H}) \subseteq J^*$, from Algorithm 1, we observe that J^* is a (topologically) closed set. If we show that $d_\infty(\text{limreach}^\epsilon(\mathcal{H}), J^*) = 0$, then the result follows immediately. This is the crux of the proof. We state this observation in a slightly stronger form below as Lemma 23. Before presenting this lemma and its proof, we

²The algorithm in [11] only considers simple cycles, i.e., cycles whose length is bounded by W . However, the proof of $J^* \subseteq \text{limreach}^\epsilon(\mathcal{H})$ in [11] does not rely on the assumption, and so even though our algorithm may consider non-simple cycles in its analysis, this does not result in any additional reachable states.

recall 3 observations from [9, 11] that we will use. The first observation (from [11]) is that if the distance between the closure of two regions is small, then they have a non-empty intersection.

Lemma 20 (Lemma 16 in [11]). *For any timed automaton $\mathcal{H}$ and for any two regions r_1 and r_2 , if $d_\infty([r_1], [r_2]) < \frac{1}{|\mathbf{X}|}$ then $[r_1] \cap [r_2] \neq \emptyset$.*

Next, we recall a result from [11] that shows that for any distance bound α and number of steps k , all trajectories of certain perturbed timed automata $\mathcal{H}_\delta^\epsilon$ are close to some trajectory of $\mathcal{H}$.

Theorem 21 (Theorem 44 in [11]). *Let $\mathcal{H}$ be a bounded timed automaton. For any distance $\alpha \in (0, 1)$ and number of steps $k \in \mathbb{N}$, there are two numbers $D(\alpha, k), E(\alpha, k) \in \mathbb{R}_+$ such that for any $\epsilon \in [0, E(\alpha, k)]$, $\delta \in [0, D(\alpha, k)]$, and a trajectory $\tau' \in [\mathcal{H}_\delta^\epsilon]_*$ such that $|\tau'| \leq k$ there is a trajectory $\tau \in [\mathcal{H}]_*$ with the following properties:*

1. $\text{first}(\tau) \in [\text{first}(\tau')]$
2. $\text{trace}(\tau) = \text{trace}(\tau')$,
3. τ is close to τ' in the following sense:
 $\forall i \in \{0, \dots, |\tau|\} \bullet d_\infty(\tau_{2i}, \tau'_{2i}) < \alpha$.

Finally, we recall a result about monotonic rectangular hybrid automata from [9] that we apply to clock drifted timed automata. The result states that corresponding to any trajectory of duration T between two states in $\text{S}_{\mathcal{H}^\epsilon}$, there is a trajectory of the same duration between the same states such that number of steps is bounded by a function of T and $\mathcal{H}$. We state this observation for $\mathcal{H}^\epsilon$ in a slightly different form than is proved in [9].

Theorem 22 (Based on Theorem 2 in [9]). *Let $\mathcal{H}$ be a timed automaton with M as the maximum constant appearing in its constraints, $\epsilon \in [0, 1]$ be a perturbation, and $T \in \mathbb{R}_{\geq 0}$ be a time bound. Then for any $\tau \in [\mathcal{H}^\epsilon]_*$ there is $\tau' \in [\mathcal{H}^\epsilon]_*$ with the following properties:*

1. $\text{first}(\tau') = \text{first}(\tau)$,
2. $\text{last}(\tau') = \text{last}(\tau)$,
3. $\text{locs}(\tau') = \text{locs}(\tau)$,
4. $\text{duration}(\tau') = \text{duration}(\tau)$, and
5. $|\tau'| \leq F(\mathcal{H}, T) = 24(2\lceil T \rceil + 1) \times |\mathbf{X}|^2 \times |\mathbf{Q}|^2 \times (2\text{cmax} + 3)^{2|\mathbf{X}|}$, where $\text{cmax} := \max\{2, M\}$.

Proof. The result follows from the observation in [9] about monotonic rectangular hybrid automata; $\mathcal{H}^\epsilon$ is such an automaton, where the maximum rate of any clock (called rmax in [9]) is bounded by 2. Note that items 3 and 4 are not explicitly mentioned in Theorem 2 in [9], but they follow from the proof in [9]. $\square$

We have now all the results from previous papers that we need to establish the proof of Lemma 23. It is useful to contrast the statements of Lemma 23 and Theorem 21. First, Lemma 23 applies to all trajectories and not just those with a bounded number of steps. Second, in Lemma 23 closeness is measured with respect to J^* (as opposed to trajectories of $\mathcal{H}$).

Lemma 23. *Let $\mathcal{H}$ be a bounded timed automaton with clocks $\mathbf{X}$. For any distance $\alpha \in (0, \frac{1}{2|\mathbf{X}|})$ there is $E \in \mathbb{R}_+$ such that for any $\epsilon \in [0, E]$ and for any $\tau' \in [\mathcal{H}^\epsilon]_*$ such that $\text{first}(\tau') \in J^*$ we have $d_\infty(\text{last}(\tau'), J^*) < \alpha$; here J^* refers to the set returned on Line 9 in Algorithm 1.*

Proof. The proof of this theorem is similar to the proof of Theorem 45 in [11] with significant departures. Let M be the maximum constant appearing on constraints of $\mathcal{H}$. We know M is a bound on clock values in $\mathcal{H}$. Let W be the number of regions in the region graph of $\mathcal{H}$ (as in line 2). Take $k = F(\mathcal{H}, 6MW + 2M)$, where F is the function from Theorem 22. We will prove that the desired $E = \min\{\frac{1}{3}, E(\alpha, k)\}$, where $E(\alpha, k)$ is from Theorem 21. Fix E to be this value. Notice that for this choice of E , the rate of every variable in $\mathcal{H}^\epsilon$ is $\geq 1 - \epsilon > \frac{1}{2}$. This is the reason we choose E to be at most $\frac{1}{3}$.

The proof is by induction on $|\tau'|$. For the base case, consider τ' such that $|\tau'| \leq k$. From Theorem 21 and the definition of E , there is a trajectory $\tau \in \llbracket \mathcal{H} \rrbracket_*$ that is α -close to τ' . Since J^* is a closed set, $\text{first}(\tau) \in [\text{first}(\tau')] \subseteq J^*$. Finally, observe that $\text{reach}(\mathcal{G}, J^*) \subseteq J^*$ ($\mathcal{G}$ is the region graph of $\mathcal{H}$) and so $\text{last}(\tau) \in J^*$. From Theorem 21, we have $d_\infty(\text{last}(\tau), \text{last}(\tau')) < \alpha$ and so $d_\infty(\text{last}(\tau'), J^*) < \alpha$.

For the inductive step, if $|\tau'| > F(\mathcal{H}, \text{duration}(\tau'))$ then Theorem 22 ensures that there is a trajectory $\tau'' \in \llbracket \mathcal{H}^\epsilon \rrbracket_*$ that starts and ends in the same state and has fewer steps. The lemma then follows by the induction hypothesis applied to τ'' .

Let us, therefore, assume that $|\tau'| \leq F(\mathcal{H}, \text{duration}(\tau'))$ and $\text{duration}(\tau') > 6MW + 2M$. Observe that all clocks in $\mathcal{H}^\epsilon$ are bounded by M , and the rate of every clock is $\geq 1 - \epsilon > \frac{1}{2}$. So no time transition has duration $2M$ or longer. Thus, without loss of generality, we can write τ' as $\tau' = \tau_{\text{first}} \sim \tau_{\text{last}}$, where $\text{duration}(\tau_{\text{last}})$ is between $6MW$ and $6MW + 2M$. From Theorem 22, there is $\tau'' \in \llbracket \mathcal{H}^\epsilon \rrbracket_*$ such that 1. $|\tau''| \leq k = F(\mathcal{H}', 6MW + 2M)$, 2. $\text{duration}(\tau'') = \text{duration}(\tau_{\text{last}})$, 3. $\text{first}(\tau'') = \text{first}(\tau_{\text{last}})$, and 4. $\text{last}(\tau'') = \text{last}(\tau_{\text{last}}) = \text{last}(\tau')$. By Theorem 21 we know there is $\tau \in \llbracket \mathcal{H} \rrbracket_*$ that stays α -close to τ'' every step of the way. Let ν_i and ν'_i be respectively states of τ and τ'' at step i . Applying the induction hypothesis to τ'' , for any $i < |\tau''|$, we have $d_\infty(\nu'_i, J^*) \leq \alpha$. Thus, by triangle inequality, for any $i < |\tau''|$,

$$d_\infty(\nu_i, J^*) \leq d_\infty(\nu_i, \nu'_i) + d_\infty(\nu'_i, J^*) \leq 2\alpha < \frac{1}{|X|}$$

Thus, using Lemma 20, we have $[\nu_i] \cap J^* \neq \emptyset$ for any $i < |\tau''|$.

The previous paragraph establishes the closeness of every state ν'_i , except possibly the last; we need to show that $\text{last}(\tau_{\text{last}}) = \text{last}(\tau'')$ is close. By construction, we know that $d_\infty(\text{last}(\tau''), \text{last}(\tau)) \leq \alpha$. Thus, proving $\text{last}(\tau) \in J^*$ will complete the induction step.

Next, because $\mathcal{H}^\epsilon$ is bounded and the rate of every variable is $> \frac{1}{2}$, every variable is reset within $2M$ time since its last reset. Thus, in any trajectory of duration $\geq 2M$, every variable is reset. Based on these observations, we can partition τ'' as $\tau'' = \tau''_1 \sim \tau''_2 \sim \dots \sim \tau''_m$, where for every i , 1. $\text{duration}(\tau''_i) \leq 2M$, 2. every variable is reset in subtrajectory τ''_i , and 3. $m \geq 6MW/2M = 3W \geq W + 2$. Since $\text{trace}(\tau) = \text{trace}(\tau'')$, we can partition τ similarly into $\tau = \tau_1 \sim \dots \sim \tau_m$, where every variable is reset in each subtrajectory τ_i . The proof can now be completed using two technical lemmas that are proved next. We will show

- There is a $j < |\tau|$ such that there is a subtrajectory of τ starting from ν_j that is a progress cycle (Lemma 24).
- $[\nu_j]$ is, in fact, part of a progress cycle of length $\leq W + 2W|X|$ (Lemma 25).

Assuming the above hold, we can conclude that $\nu_j \in J^*$

because of line 6 of Algorithm 1. Also, since τ is a trajectory of $\mathcal{H}$, we can conclude that $\nu_i \in J^*$ for all $i \geq j$ (Line 7 of Algorithm 1), completing the proof of the induction step. $\square$

Lemma 24. *Let $\mathcal{H}$ be a timed automaton with W regions in its region graph. Let τ be any trajectory of $\mathcal{H}$ such that*

$$\tau = \tau_1 \sim \tau_2 \sim \dots \sim \tau_m$$

where $m \geq W + 2$ and every variable of $\mathcal{H}$ is reset in each τ_i (for every i). Then there is a subtrajectory of τ that is a progress cycle.

Proof. As $m \geq W + 2$ and the number of regions is bounded by W , by pigeon-hole principle, there must be s, t and region r such that 1. $t \geq s + 2$, and 2. r is visited in both τ_s and τ_t . These observations ensure that there is a subtrajectory of $\tau_s \sim \dots \sim \tau_t$ that forms a cycle, and since there is an index u between s and t , it is also a progress cycle since all the variables will be reset in τ_u . $\square$

Lemma 25. *Let $\mathcal{H}$ be any timed automaton with $\mathcal{G}$ as its region graph and W as the number of nodes in $\mathcal{G}$. For any cycle π in $\mathcal{G}$ there is a cycle π' in $\mathcal{G}$ such that*

1. $|\pi'| \leq W + 2W|X|$,
2. $\pi_0 = \pi'_0$ (they start from the same node), and
3. π and π' reset the same set of clocks.

We conclude this section by observing that $\text{limreach}^\epsilon(\mathcal{H})$ can be computed in polynomial space.

Theorem 26. *For any bounded timed automaton $\mathcal{H}$, $\text{limreach}^\epsilon(\mathcal{H})$ can be computed by an algorithm that uses space that is bounded by a polynomial function of $|\mathcal{H}|$.*

Proof. From Theorem 19, we know that $J^* = \text{limreach}^\epsilon(\mathcal{H})$. If we show that Algorithm 1 uses only polynomial space, then we are done. Unfortunately, as presented, Algorithm 1 uses exponential space and runs in exponential time. But small changes to the presentation of Algorithm 1 establish the complexity bounds stated in the theorem.

First observe that J^* is a union of regions of $\mathcal{H}$. Therefore, if we show that, given a region r of $\mathcal{H}$, the problem of determining if $r \in J^*$ can be solved in polynomial space then we will establish the complexity bounds. Based on lines 3 through 7, we conclude that $r \in J^*$ iff there is a path $\pi := \pi_0 \pi_1 \pi_2 \dots \pi_{2m}$, for some $m \leq W$ such that

1. π_0 is an initial region of $\mathcal{H}$,
2. π_{2i+1} is reachable from π_{2i} , for $0 \leq i \leq m$,
3. $[\pi_{2i}] \cap \pi_{2i-1} \neq \emptyset$, for $1 \leq i \leq m$, and
4. for $1 \leq i \leq m$, π_{2i} is on a progress cycle.

By progressively guessing regions on a path/cycle, each of the above conditions can be checked in (nondeterministic) polynomial space, giving us the necessary bounds. $\square$

5. EXPERIMENTAL RESULTS

We applied the above algorithmic ideas to verify the robustness of two protocols. Models of both protocols are available in UPPAAL. The first one we considered is Fischer's mutual exclusion protocol for n processes. Any mutual exclusion protocol must satisfy at least three properties: 1. no two processes enter a critical section at the same time, 2. there are no deadlocks in the system, and 3. any request to access a critical section will eventually be granted. The first property is a safety property and the next two are liveness

properties. Each of these properties can be verified to hold for the unperturbed protocol using UPPAAL for $n \leq 6$ in less than 2 seconds. Using UPPAAL, we could also show that the system satisfies each of these properties when there are 6 processes and guards are enlarged by $\delta := 0.01$ ³; the verification time for this was also less than 2 seconds. Therefore, using Lemma 7 and knowing the maximum constant in this model is 3, the model is also robust with $\epsilon := \frac{0.01}{12}$ and $\delta := \frac{0.01}{2}$.

The next example we verified, is 2Doors. It involves two doors and two users that interact using the following rules: 1. a room has two doors which cannot be opened at the same time, 2. a door starts to open if its button is pushed, 3. it takes six seconds for a door to open, and thereafter it says open for at least four seconds, but no more than eight seconds, 4. it takes six seconds for a door to close and it stays closed for at least five seconds. We checked the following properties: 1. Mutex: The two doors are never open at the same time. 2. Either of doors can be opened. 3. Liveness: Whenever a button is pushed, the corresponding door will eventually open. 4. The system is deadlock-free. Using UPPAAL, in less than 2 seconds we could show that the system satisfies each of these properties when guards are enlarged by $\delta := 0.0001$. Therefore, using Lemma 7 and knowing the maximum constant in this model is 8, the model is also robust with $\epsilon := \frac{0.0001}{32}$ and $\delta := \frac{0.0001}{2}$.

Note that properties we checked in these two examples are not necessarily expressible using the special type of (robust) ω -regular model checking problems we have considered in this paper. However, Lemma 7 establishes language inclusion between executions of $\mathcal{H}$ under different types of perturbation, regardless of the property one might want to check.

6. CONCLUSION

In this paper we have investigated the robust model checking problem for timed automata without the progress cycle assumption, when only clocks are allowed to drift and when both guards and clocks are perturbed. We first showed that robust model checking of ω -regular properties with only enlarged guards is equivalent to the case when guards and clock rates are perturbed, and obtained a PSPACE-complete algorithm for the later case. We then gave another PSPACE-complete algorithm for model checking ω -regular properties when only clocks are allowed to drift. Finally, we gave a polynomial space algorithm to compute $\text{limreach}^\epsilon(\mathcal{H})$. However, our results do not yield an algorithm to compute $\text{limreach}_\delta^\epsilon(\mathcal{H})$ ($= \text{limreach}_\delta(\mathcal{H})$). This is a possible direction for future exploration.

7. ACKNOWLEDGEMENT

We gratefully acknowledge the support of the following grants — Nima Roohi was partially supported by NSF CNS 1329991 and AFOSR FA9950-15-1-0059; Pavithra Prabhakar was partially supported by NSF CAREER Award 1552668; and Mahesh Viswanathan was partially supported by NSF CCF 1422798, NSF CNS 1329991 and AFOSR FA9950-15-1-0059.

³ The model has a strict inequality $x > k$. We first replace that with $x \geq k + 1$ and positively verified that all properties are still satisfied.

8. REFERENCES

- [1] K. Altisen and S. Tripakis. Implementation of timed automata: An issue of semantics or modeling? In *Proceedings of FORMATS*, pages 273–288, 2005.
- [2] R. Alur and D. L. Dill. A theory of timed automata. *TCS*, 126:183–235, 1994.
- [3] R. Alur, S. L. Torre, and P. Madhusudan. Perturbed timed automata. In *Proceedings of HSCC*, pages 70–85, 2005.
- [4] C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. 2008.
- [5] G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *Proceedings of HSCC*, pages 147–161, 2001.
- [6] P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of linear-time properties in timed automata. In *Proceedings of LATIN*, pages 238–249, 2006.
- [7] P. Bouyer and F. Chevalier. On conciseness of extensions of timed automata. *J. Autom. Lang. Comb.*, 10(4):393–405, April 2005.
- [8] P. Bouyer, N. Markey, and O. Sankur. Robust model-checking of timed automata via pumping in channel machines. In *Proceedings of FORMATS*, pages 97–112, 2011.
- [9] T. Brihaye, L. Doyen, G. Geeraerts, J. Ouaknine, J. F. Raskin, and J. Worrell. Time-bounded reachability for monotonic hybrid automata: Complexity and fixed points. In *Proceedings of ATVA*, volume 8172 of *LNCIS*, pages 55–70, 2013.
- [10] F. Cassez, T. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *Proceedings of HSCC*, pages 134–148, 2002.
- [11] M. de Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robust safety of timed automata. *Formal Methods in System Design*, 33(1):45–84, 2008.
- [12] V. Gupta, T. Henzinger, and R. Jagadeesan. Robust timed automata. In *Proceedings of the International Workshop on Hybrid and real-time Systems*, pages 331–345, 1997.
- [13] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What’s decidable about hybrid automata? In *Journal of Computer and System Sciences*, pages 373–382, 1995.
- [14] K. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1:134–152, 1997.
- [15] A. Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.
- [16] M. D. Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: From timed models to timed implementations. *Formal Aspects of Computing*, 17(3):319–341, 2005.
- [17] M. D. Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. In *Proceedings of FORMATS*, pages 118–133, 2004.
- [18] S. Yovine. KRONOS: A verification tool for real-time systems. *International Journal on Software Tools for Technology Transfer*, 1:123–133, 1997.