

COARSE ABSTRACTIONS MAKE ZENO BEHAVIOURS DIFFICULT TO DETECT*

FRÉDÉRIC HERBRETEAU^a AND B. SRIVATHSAN^b

^a Univ. Bordeaux, CNRS, LaBRI, UMR 5800, F-33400 Talence, France
e-mail address: fh@labri.fr

^b Software Modeling and Verification group, RWTH Aachen University, Germany
e-mail address: sri@cs.rwth-aachen.de

ABSTRACT. An infinite run of a timed automaton is Zeno if it spans only a finite amount of time. Such runs are considered unfeasible and hence it is important to detect them, or dually, find runs that are non-Zeno. Over the years important improvements have been obtained in checking reachability properties for timed automata. We show that some of these very efficient optimizations make testing for Zeno runs costly. In particular we show NP-completeness for the LU-extrapolation of Behrmann et al. We analyze the source of this complexity in detail and give general conditions on extrapolation operators that guarantee a (low) polynomial complexity of Zenoness checking. We propose a slight weakening of the LU-extrapolation that satisfies these conditions.

INTRODUCTION

Timed automata [1] are finite automata augmented with a finite number of clocks. The values of the clocks increase synchronously along with time in the states of the automaton and these values can be compared to a constant and reset to zero while crossing a transition. This model has been successfully used for verification of real-time systems thanks to a number of tools [4, 7, 21].

Since timed automata model reactive systems that continuously interact with the environment, it is interesting to consider questions related to their infinite executions. An execution is said to be *Zeno* if an infinite number of events happen in a finite time interval. Such executions are clearly unfeasible. During verification, the aim is to detect if there exists a *non-Zeno* execution that violates a certain property. On the other hand while

2012 ACM CCS: [Software and its engineering]: Software creation and management—Software verification and validation—Formal software verification; [Theory of computation]: Computational complexity and cryptography—Complexity classes; [Theory of computation]: Formal languages and automata theory—Automata extensions.

Key words and phrases: Timed automata, Zeno runs, Abstractions, Verification.

* Extended abstract appeared at CONCUR2011.

^a This work has been supported by ANR project DOTS ANR-06-SETI-003.

^b The second author B. Srivathsan was at LaBRI, Univ. Bordeaux, when this work was first submitted.

implementing timed automata, it is required to check the presence of pathological Zeno executions. This brings the motivation to analyze an automaton for the presence of such executions.

The analysis of timed automata faces the challenge of handling its uncountably many configurations. To tackle this problem, one considers a finite graph called the *abstract zone graph* (also known as simulation graph) of the automaton. This finite graph captures the semantics of the automaton. In this paper, we consider the problems of deciding if an automaton has a non-Zeno execution, dually a Zeno execution, given its abstract zone graph as input.

An abstract zone graph is obtained by over-approximating each zone of the so-called *zone graph* with an abstraction function. The zone graph in principle could be infinite and an abstraction function is necessary for reducing it to a finite graph. The coarser the abstraction, the smaller the abstract zone graph, and hence the quicker the analysis of the automaton. This has motivated a lot of research towards finding coarser abstraction functions [3]. The classic maximum-bound abstraction uses as a parameter the maximal constant a clock gets compared to in a transition. A coarser abstraction called the LU-extrapolation was introduced in Behrmann et al. [3] for checking state reachability in timed automata. This is the coarsest among all the implemented approximations and is at present efficiently used in tools like UPPAAL [4].

It was shown in [19, 20] that even infinite executions of the automaton directly correspond to infinite paths in the abstract zone graph when one uses the maximum-bound approximation. In addition, it was proved that the existence of a non-Zeno infinite execution could be determined by adding an extra clock to the automaton to keep track of time and analyzing the abstract zone graph of this transformed automaton [18, 20]. A similar correspondence was established in the case of the LU-extrapolation by Li [16]. These results answer our question about deciding non-Zeno infinite executions of the automaton from its abstract zone graph. However, it was shown in [14] that adding a clock has an exponential worst case complexity. A new polynomial construction was proposed for the case of the classic maximum-bound approximation. But, the case of the LU-extrapolation was not addressed.

In this paper, we prove that the non-Zenoness question turns out to be NP-complete for the LU-extrapolation, that is, given the abstract zone graph over the LU-extrapolation, deciding if the automaton has a non-Zeno execution is NP-complete. We study the source of this complexity in detail and give conditions on abstraction operators to ensure a polynomial complexity. To this regard, we extend the polynomial construction given in [14] to an arbitrary abstraction function and analyze when it stays polynomial. It then follows that a slight weakening of the LU-extrapolation makes the construction polynomial. In the second part of the paper, we repeat the same for the dual question: given an automaton's abstract zone graph, decide if it has Zeno executions. Yet again, we notice NP-completeness for the LU-extrapolation. We introduce an algorithm for checking Zenoness over an abstract zone graph with conditions on the abstraction operator to ensure a polynomial complexity. We provide a different weakening of LU-extrapolation that gives a polynomial solution to the Zenoness question. Finally, we also prove that deciding if a given automaton has a non-Zeno run (resp. Zeno run) is PSPACE-complete when the input is restricted to the automaton only.

Note that the reachability problem for timed automata is PSPACE-complete [1, 8] and the standard algorithms make use of the abstract zone graph to solve the reachability

problem. Therefore one could expect an object as complex as the abstract zone graph to solve the Zeno-related questions too. This makes the complexity analysis of the Zeno-related questions, given both the automaton and abstract zone graph as input, all the more relevant.

Related work. As mentioned above, the LU-extrapolation was proposed in [3] and shown how it could be efficiently used in UPPAAL for the purpose of reachability. The correctness of the classic maximum-bound abstraction was shown in [5]. Extensions of these results to infinite executions occur in [20, 16]. Detection of non-Zeno runs was already addressed in [1]. Their approach works on the region graph, but for correctness reasons, it cannot be used on (abstract) zone graphs. The trick involving adding an extra clock for non-Zenoness is discussed in [18, 20, 2, 14]. The problem of checking existence of Zeno runs was formulated as early as in [18]. A bulk of the literature for this problem also directs to [10, 6, 17]. All of these solutions provide a sufficient-only condition for the absence of Zeno runs. This is different from our proposed solution which gives a complete solution (necessary and sufficient conditions) by analyzing the abstract zone graph of the automaton.

Organization of the paper. We start with the formal definitions of timed automata, abstract zone graphs, the Zenoness and non-Zenoness problems in Section 1. Subsequently, we prove the NP-hardness of the non-Zenoness problem for the LU-extrapolation in Section 2. We then recall in Section 3 the construction proposed for non-Zenoness in [14] and extend it to a general abstraction operator giving conditions for polynomial complexity. Section 4 is dedicated to the dual Zenoness problem. In Section 5 we discuss some interesting observations arising out of the entire complexity analysis. We prove in Section 6 that finding if an automaton has a (non-)Zeno run turns out to PSPACE-complete when the input is restricted to the automaton only. This gives a complete characterization of the complexity of finding (non-)Zeno runs in timed automata. We conclude the paper with some perspectives in Section 7.

A shorter version of this paper appeared at the 22nd International Conference on Concurrency Theory in the year 2011 [12]. The current version includes the missing proofs, a new discussion (Section 5) about two observations arising out of the complexity analysis, and the new result about the PSPACE-completeness of the Zeno-related problems when the only input is the automaton (Section 6).

1. ZENO-RELATED PROBLEMS FOR TIMED AUTOMATA

1.1. Timed automata. Let $\mathbb{R}_{\geq 0}$ denote the set of non-negative real numbers. Let X be a set of variables, named *clocks* hereafter. A *valuation* is a function $v : X \mapsto \mathbb{R}_{\geq 0}$ that maps every clock in X to a non-negative real value. We denote the set of all valuations by $\mathbb{R}_{\geq 0}^X$, and $\mathbf{0}$ the valuation that maps every clock in X to 0. For $\delta \in \mathbb{R}_{\geq 0}$, we denote $v + \delta$ the valuation mapping each $x \in X$ to the value $v(x) + \delta$. For a subset R of X , let $[R]v$ be the valuation that sets x to 0 if $x \in R$ and assigns $v(x)$ otherwise. A *clock constraint* is a conjunction of constraints $x \# c$ for $x \in X$, $\# \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{N}$. We denote $\Phi(X)$ the set of clock constraints over clock variables X . For a valuation v and a constraint ϕ we write $v \models \phi$ when v satisfies ϕ , that is, when ϕ holds after replacing every x by $v(x)$.

A *Timed Automaton (TA)* [1] \mathcal{A} is a finite automaton extended with clocks that enable or disable transitions. Formally, \mathcal{A} is a tuple (Q, q_0, X, T) where Q is a finite set of states, $q_0 \in Q$ is the initial state, X is a finite set of clocks and $T \subseteq Q \times \Phi(X) \times 2^X \times Q$ is a finite set of transitions. For each transition $(q, g, R, q') \in T$, g is a clock constraint, also called a *guard* that defines the valuations of the clocks that are allowed to cross the transition, and R is a set of clocks that are *reset* on the transition.

The semantics of a timed automaton \mathcal{A} is a transition system of its configurations. A *configuration* of \mathcal{A} is a pair $(q, v) \in Q \times \mathbb{R}_{\geq 0}^X$, with $(q_0, \mathbf{0})$ being the *initial configuration*. We have two kinds of transitions:

delay: $(q, v) \rightarrow^\delta (q, v + \delta)$ for some $\delta \in \mathbb{R}_{\geq 0}$;

action: $(q, v) \rightarrow^t (q', v')$ for some transition $t = (q, g, R, q') \in T$ such that $v \models g$ and $v' = [R]v$.

A *run* of \mathcal{A} is a (finite or infinite) sequence of transitions starting from the initial configuration $(q_0, \mathbf{0})$. Without loss of generality, we can assume that the first transition is a delay transition and that delay and action transitions alternate. We write $(q, v) \xrightarrow{\delta, t} (q', v')$ if there is a delay transition $(q, v) \rightarrow^\delta (q, v + \delta)$ followed by an action transition $(q, v + \delta) \rightarrow^t (q', v')$. So a run of \mathcal{A} can be written as:

$$(q_0, v_0) \xrightarrow{\delta_0, t_0} (q_1, v_1) \xrightarrow{\delta_1, t_1} (q_2, v_2) \cdots (q_i, v_i) \cdots$$

where (q_0, v_0) represents the initial configuration $(q_0, \mathbf{0})$.

Definition 1.1 (Zeno/non-Zeno runs). An infinite run $(q_0, v_0) \xrightarrow{\delta_0, t_0} \dots (q_i, v_i) \xrightarrow{\delta_i, t_i} \dots$ is *Zeno* if time does not diverge, that is, $\sum_{i \geq 0} \delta_i \leq c$ for some $c \in \mathbb{R}_{\geq 0}$. Otherwise it is *non-Zeno*.

Theorem 1.2. *The problem of deciding if a timed automaton \mathcal{A} has a non-Zeno run (resp. Zeno run) is PSPACE-complete if \mathcal{A} is the only input.*

A proof of Theorem 1.2 is given in Section 6 (page 27) that relies on results in Sections 3 and 4.2.

As can be seen, the number of configurations (q, v) could be uncountable. We now define an abstract semantics for timed automata. The abstract semantics is usually used for the verification of timed automata.

1.2. Symbolic semantics. We begin with the definition of special *sets of valuations* called zones. A *zone* is a set of valuations defined by a conjunction of two kinds of clock constraints: for $x_i, x_j \in X$

$$\begin{aligned} x_i &\sim c \\ x_i - x_j &\sim c \end{aligned}$$

where $\sim \in \{\leq, <, =, >, \geq\}$ and $c \in \mathbb{Z}$. An example of a zone over two clocks x_1 and x_2 is illustrated in Figure 1. The shaded area is the zone represented by the conjunction of the six constraints shown in the figure.

Zones can be efficiently represented by Difference Bound Matrices (DBMs) [9]. A DBM representation of a zone Z is a $|X| + 1$ square matrix $(Z_{ij})_{i, j \in [0; |X|]}$ where each entry $Z_{ij} = (c_{ij}, \preccurlyeq_{ij})$ represents the constraint $x_i - x_j \preccurlyeq_{ij} c_{ij}$ for $c_{ij} \in \mathbb{Z}$ and $\preccurlyeq_{ij} \in \{<, \leq\}$ or

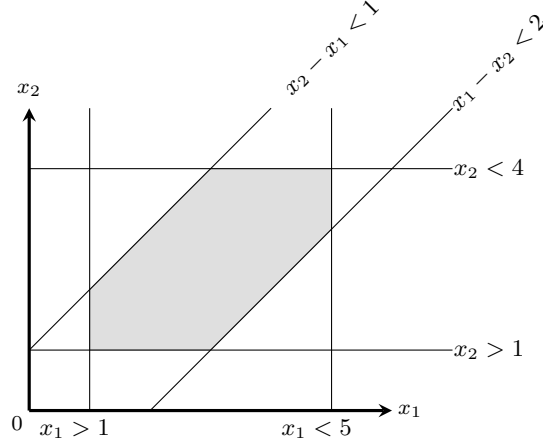


Figure 1: An example of a zone.

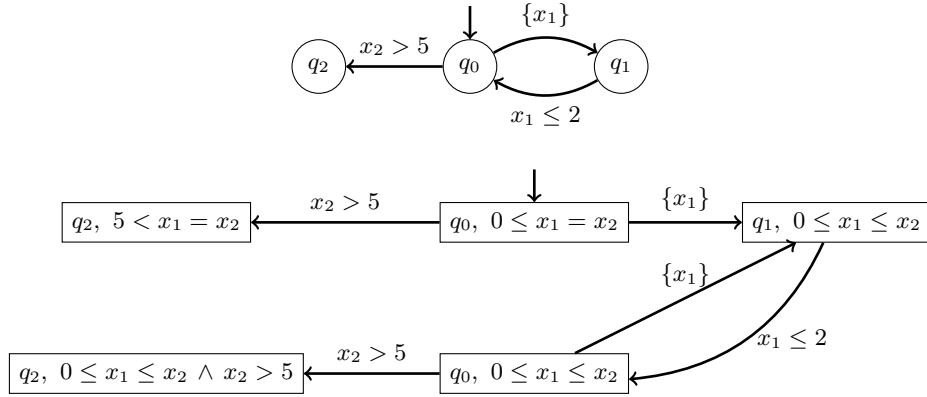


Figure 2: A timed automaton (top) and its zone graph (bottom).

$(c_{ij}, \preccurlyeq_{ij}) = (\infty, <)$. A special variable x_0 encodes the value 0. Hence, in a DBM $x_i > 4$ is encoded as $x_0 - x_i < -4$.

The *symbolic semantics* (or *zone graph*) of an automaton \mathcal{A} is the transition system $ZG(\mathcal{A})$ given by the tuple (S, s_0, \Rightarrow) , where S is the set of nodes, s_0 is the initial node and \Rightarrow is the transition relation. Each node in S is a pair (q, Z) consisting of a state q of the automaton and a zone Z . The initial node s_0 is (q_0, Z_0) where $Z_0 = \{\mathbf{0} + \delta \mid \delta \in \mathbb{R}_{\geq 0}\}$. For every $t = (q, g, R, q') \in T$, there exists a transition \Rightarrow^t from a node (q, Z) as follows:

$$(q, Z) \Rightarrow^t (q', Z') \quad \text{where } Z' = \{v' \mid \exists v \in Z, \exists \delta \in \mathbb{R}_{\geq 0} : (q, v) \rightarrow^t \rightarrow^\delta (q', v')\}$$

In the above definition, $\rightarrow^t \rightarrow^\delta$ denotes the discrete transition t followed by a delay transition of δ time units. It can be shown that if Z is a zone, then Z' is a zone. Moreover, a DBM representation of Z' can be computed from the DBM representation of Z (see for instance [5]). Figure 2 shows an example of an automaton and its zone graph.

Several definitions of $ZG(\mathcal{A})$ have been considered in the literature. They differ on the definition of \Rightarrow . People have considered graphs with both action \Rightarrow^t and delay \Rightarrow^δ

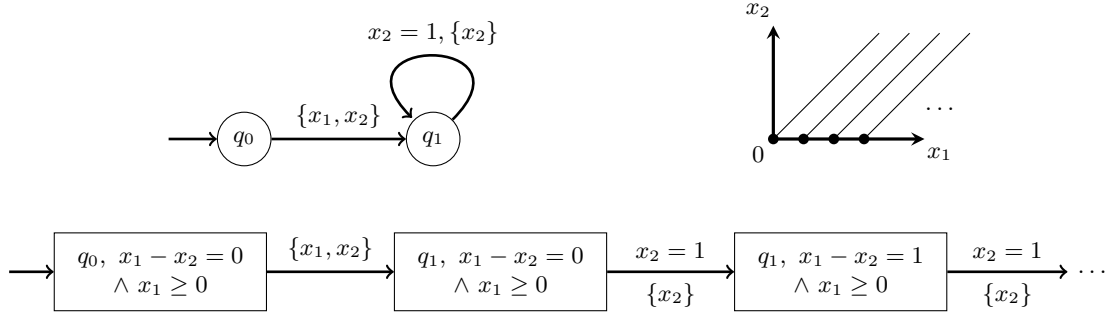


Figure 3: Automaton \mathcal{A}_{inf} (top left), the graph of zones obtained at q_1 (top right) and a part of the infinite zone graph $ZG(\mathcal{A}_{inf})$ (bottom).

transitions or, with only combined transitions \Rightarrow^t , but corresponding to the reverse consecution $\rightarrow^\delta \rightarrow^t$ (delay-then-action). Our results do not depend on a specific choice, but have a simpler presentation using the chosen symbolic semantics.

Although the zone graph $ZG(\mathcal{A})$ deals with sets of valuations instead of valuations themselves, the zone graph could still be infinite. Consider the automaton \mathcal{A}_{inf} shown in Figure 3, with two clocks x_1 and x_2 . The initial node is given by $(q_0, x_1 = x_2 \wedge x_1 \geq 0)$. The transition to q_1 gives the node $(q_1, x_1 = x_2 \wedge x_1 \geq 0)$. The only transition from q_1 taken from this node gives the node $(q_1, x_1 - x_2 = 1 \wedge x_1 \geq 0)$, which is a new node. This node has its own successors and the process continues. Finally at q_1 we have the following zones in the zone graph $ZG(\mathcal{A}_{inf})$:

$$(x_1 - x_2 = k \wedge x_1 \geq 0) \text{ for all } k \in \mathbb{N}$$

This is pictorially shown in Figure 3. It is however sufficient to consider a finite abstraction of the zone graph to capture all the behaviors of a timed automaton. Several abstractions have been introduced to obtain a finite graph from $ZG(\mathcal{A})$.

1.3. Abstract symbolic semantics. A *finite abstraction* is a function $\mathfrak{a} : 2^{\mathbb{R}_{\geq 0}^{|\mathcal{X}|}} \rightarrow 2^{\mathbb{R}_{\geq 0}^{|\mathcal{X}|}}$ such that for every zone Z : $\mathfrak{a}(Z)$ is a zone, $Z \subseteq \mathfrak{a}(Z)$, $\mathfrak{a}(\mathfrak{a}(Z)) = \mathfrak{a}(Z)$ and \mathfrak{a} has a finite range. An abstraction operator defines an abstract semantics.

$$(q, Z) \Rightarrow_{\mathfrak{a}}^t (q', \mathfrak{a}(Z'))$$

when $\mathfrak{a}(Z) = Z$ and $(q, Z) \Rightarrow^t (q', Z')$ in $ZG(\mathcal{A})$.

The *abstract symbolic semantics* (or the *abstract zone graph*) of \mathcal{A} is the transition system $ZG^{\mathfrak{a}}(\mathcal{A})$ induced by $\Rightarrow_{\mathfrak{a}}$ starting from the node $(q_0, \mathfrak{a}(Z_0))$, where (q_0, Z_0) is the initial node of $ZG(\mathcal{A})$.

A *path* π in $ZG^{\mathfrak{a}}(\mathcal{A})$ is a (finite or infinite) sequence of transitions

$$(q_0, Z'_0) \Rightarrow_{\mathfrak{a}}^{t_0} (q_1, Z'_1) \Rightarrow_{\mathfrak{a}}^{t_1} \dots (q_i, Z'_i) \Rightarrow_{\mathfrak{a}}^{t_i} \dots$$

We say that a run $\rho: (q_0, v_0) \xrightarrow{\delta_0, t_0} \dots (q_i, v_i) \xrightarrow{\delta_i, t_i} \dots$ of \mathcal{A} is an *instance* of the path π of $ZG^{\mathfrak{a}}(\mathcal{A})$ as described above, if ρ and π agree on the sequence of transitions t_0, t_1, \dots , and if for every $i \geq 0$, (q_i, v_i) and (q_i, Z'_i) coincide on q_i , and $v_i \in Z'_i$. By definition of Z'_i , this implies $v_i + \delta_i \in Z'_i$.

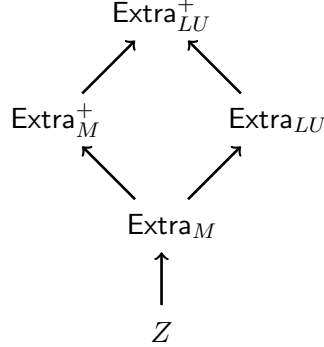


Figure 4: Comparison of the finite abstractions [3].

An abstraction \mathbf{a} is *sound* if every path of $ZG^{\mathbf{a}}(\mathcal{A})$ can be instantiated as a run of \mathcal{A} . Conversely, \mathbf{a} is *complete* when every run of \mathcal{A} is an instance of some path in $ZG^{\mathbf{a}}(\mathcal{A})$. If an abstraction \mathbf{b} satisfies $\mathbf{b}(Z) \subseteq \mathbf{a}(Z)$ for every Z , it is easy to see that the abstract zone graph $ZG^{\mathbf{b}}(\mathcal{A})$ is bigger than $ZG^{\mathbf{a}}(\mathcal{A})$.

1.4. Bounds and finite abstractions. A standard way to obtain finite abstractions is to consider as a parameter, a bound function $M : X \mapsto \mathbb{N} \cup \{-\infty\}$ that associates to each clock x , the maximum integer c appearing in a guard involving x . Abstractions Extra_M [5] and Extra_M^+ [3] are well-known finite abstractions that depend on such a bound function M .

It has been observed that considering separately the guards that lower-bound clocks and guards that upper-bound clocks leads to much coarser abstractions and hence to much smaller abstract zone graphs. This has given rise to abstractions Extra_{LU} and Extra_{LU}^+ [3] which are currently used in implementations. We recall the definitions of Extra_{LU} and Extra_{LU}^+ below.

Let $L : X \mapsto \mathbb{N} \cup \{-\infty\}$ and $U : X \mapsto \mathbb{N} \cup \{-\infty\}$ be two maps that associate to each clock in \mathcal{A} its maximal lower bound and its maximal upper bound respectively: that is, for every $x \in X$, $L(x)$ is the maximal integer c such that $x > c$ or $x \geq c$ appears in some guard of \mathcal{A} . We let $L(x) = -\infty$ if no such c exists. Similarly, we define $U(x)$ with respect to clock constraints like $x \leq c$ and $x < c$. We define $\text{Extra}_{LU}(Z) = Z^{LU}$ and $\text{Extra}_{LU}^+(Z) = Z^{LU+}$ as:

$$Z_{ij}^{LU} = \begin{cases} (\infty, <) & \text{if } c_{ij} > L(x_i) \\ (-U(x_j), <) & \text{if } -c_{ij} > U(x_j) \\ Z_{ij} & \text{otherwise} \end{cases} \quad \Bigg| \quad Z_{ij}^{LU+} = \begin{cases} (\infty, <) & \text{if } c_{ij} > L(x_i) \\ (\infty, <) & \text{if } -c_{0i} > L(x_i) \\ (\infty, <) & \text{if } -c_{0j} > U(x_j), i \neq 0 \\ (-U(x_j), <) & \text{if } -c_{0j} > U(x_j), i = 0 \\ Z_{ij} & \text{otherwise} \end{cases}$$

In the above, we set $L(x_0) = U(x_0) = 0$ for the special clock x_0 . The abstraction Extra_M (resp. Extra_M^+) is obtained from Extra_{LU} (resp. Extra_{LU}^+) by replacing every occurrence of L and U by M which maps every clock x to $\max(L(x), U(x))$. These abstractions compare in the following way (cf. Figure 4).

Theorem 1.3 ([3]). *For every zone Z , we have: $Z \subseteq \text{Extra}_M(Z) \subseteq \text{Extra}_M^+(Z)$; $Z \subseteq \text{Extra}_{LU}(Z) \subseteq \text{Extra}_{LU}^+(Z)$ and $\text{Extra}_M^+(Z) \subseteq \text{Extra}_{LU}^+(Z)$.*

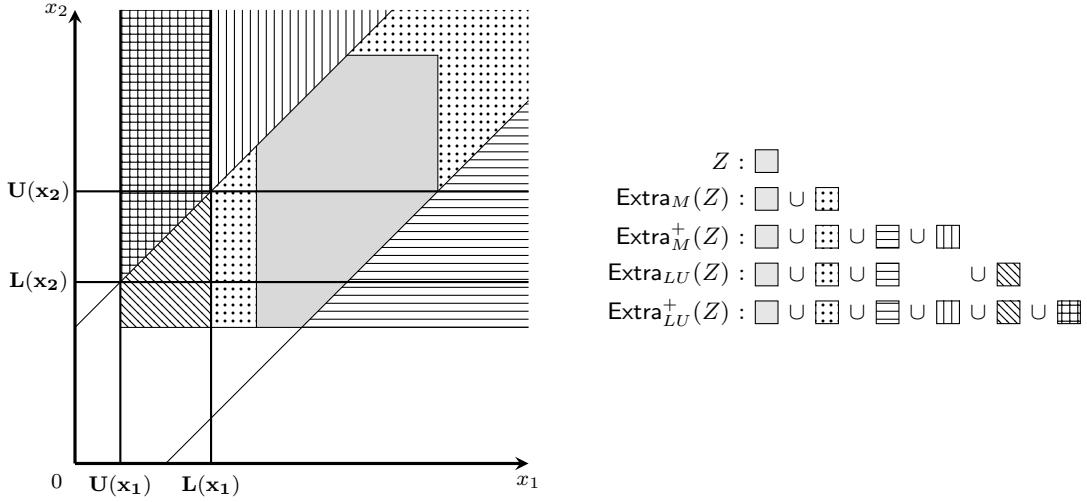


Figure 5: An illustration of the abstraction hierarchy shown in Figure 4.

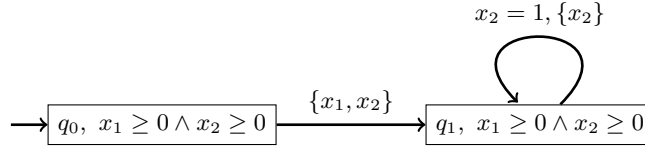


Figure 6: $ZG^{\mathbf{a}}(\mathcal{A}_{inf})$ for the automaton \mathcal{A}_{inf} shown in Figure 3. We get the same abstract zone graph $ZG^{\mathbf{a}}(\mathcal{A}_{inf})$ for \mathbf{a} being either Extra_M , Extra_M^+ , Extra_{LU} or Extra_{LU}^+ .

Figure 5 shows a zone and depicts the action of the different abstractions on it. In the rest of the paper, we say M -extrapolations for Extra_M and Extra_M^+ ; and LU -extrapolations for Extra_{LU} and Extra_{LU}^+ .

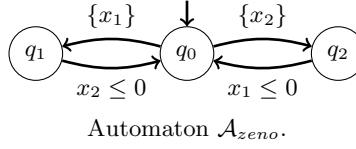
Let us look at the timed automaton \mathcal{A}_{inf} of Figure 3. For this automaton, the maximum bounds function M sets $M(x_1) = -\infty$ and $M(x_2) = 1$. Define:

$$\text{for } k \in \mathbb{N}, \quad Z_k \equiv (x_1 - x_2 = k) \wedge (x_1 \geq 0)$$

By the definition of Extra_M , every zone Z_k has $\text{Extra}_M(Z_k)$ given by the constraints $(x_1 \geq 0 \wedge x_2 \geq 0)$. Therefore, the zone graph $ZG^{\mathbf{a}}(\mathcal{A}_{inf})$ has two nodes for \mathbf{a} being any of the abstractions defined above.

1.5. Zenoness and non-Zenoness problems. A classical verification problem for timed automata is to answer state reachability queries. For this purpose, we consider only the runs of \mathcal{A} and paths in $ZG^{\mathbf{a}}(\mathcal{A})$ that are *finite* sequences of transitions. A reachability query asks if there exists a run of \mathcal{A} leading to a given state. Such problems can be solved using $ZG^{\mathbf{a}}(\mathcal{A})$ when \mathbf{a} is sound and complete. This is true for the M -extrapolations and LU -extrapolations.

Theorem 1.4 ([5, 3]). *Extra_M , Extra_M^+ , Extra_{LU} and Extra_{LU}^+ are sound and complete for finite sequences of transitions.*



$$(q_0, 0 = x_1 = x_2) \longrightarrow (q_1, 0 = x_1 \leq x_2) \longrightarrow (q_0, 0 = x_1 = x_2) \longrightarrow (q_2, 0 = x_2 \leq x_1) \text{ -----} \longrightarrow$$

A path in the abstract zone graph of \mathcal{A}_{zeno} with abstraction Extra_M .

$$(q_0, \top) \longrightarrow (q_1, \top) \longrightarrow (q_0, \top) \longrightarrow (q_2, \top) \text{ -----} \longrightarrow$$

A path in the abstract zone graph of \mathcal{A}_{zeno} with abstraction Extra_{LU} .

Figure 7: Zenoness/non-Zenoness from abstract paths.

Liveness properties require the existence of an infinite run satisfying a given property. For instance, does \mathcal{A} visit state q infinitely often? Soundness and completeness of \mathbf{a} with respect to infinite runs allow to solve such problems from $ZG^{\mathbf{a}}(\mathcal{A})$. It has also been proved that the extrapolations mentioned above are also sound and complete for infinite paths/runs.

Theorem 1.5 ([19, 16]). Extra_M , Extra_M^+ , Extra_{LU} and Extra_{LU}^+ are sound and complete for infinite sequences of transitions.

Thanks to Theorem 1.5, we know that every infinite path π in $ZG^{\mathbf{a}}(\mathcal{A})$ can be instantiated to a run of \mathcal{A} . However, soundness is not sufficient to know if π can be instantiated to a *non-Zeno* run. Additionally, it is also interesting to know when this path can be instantiated to a *Zeno* run. In the sequel, we consider the following questions, given an automaton \mathcal{A} and an abstract zone graph $ZG^{\mathbf{a}}(\mathcal{A})$.

INPUT	\mathcal{A} and $ZG^{\mathbf{a}}(\mathcal{A})$
NON-ZENONESS PROBLEM ($\text{NZP}^{\mathbf{a}}$)	Does \mathcal{A} have a non-Zeno run?
ZENONESS PROBLEM ($\text{ZP}^{\mathbf{a}}$)	Does \mathcal{A} have a Zeno run?

Observe that solving $\text{ZP}^{\mathbf{a}}$ does not solve $\text{NZP}^{\mathbf{a}}$ and vice-versa: one is not the negation of the other. Note that the coarser the abstraction, the lesser is the information maintained about the structure of a zone. Let us motivate by an example.

Figure 7 shows an automaton \mathcal{A}_{zeno} which has all runs Zeno. As we can see, the coarser the abstraction used, the lesser is the information in the simulation graph that one could tap to detect non-Zenoness or Zenoness.

In this paper, we focus on the complexity of deciding $\text{ZP}^{\mathbf{a}}$ and $\text{NZP}^{\mathbf{a}}$ for different abstractions \mathbf{a} . We denote NZP^M and ZP^M when the M -extrapolations are considered. We similarly define NZP^{LU} and ZP^{LU} for the LU -extrapolations.

The non-Zenoness problem is known to be solvable in polynomial time when abstraction Extra_M is considered [14]. This is not true for abstraction Extra_{LU} : in Section 2 we show that NZP^{LU} is NP-hard. As the LU -extrapolations are coarser, one might expect the non-Zenoness question to be tougher to infer. But, it is surprising that the difficulty rises to

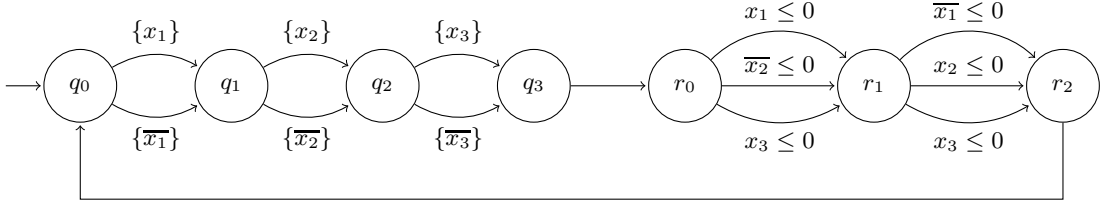


Figure 8: \mathcal{A}_ϕ^{NZ} for $\phi = (p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2 \vee p_3)$

the extent of leading to NP-hardness as opposed to a low polynomial complexity for M -extrapolations. The same asymmetry appears in the Zenoness problem as well, which is shown in Section 4.

In addition to these complexity results, in Section 3, we generalize the construction for non-Zenoness given in [14] to an arbitrary finite abstraction operator and describe the class of abstractions for which NZP^a stays polynomial. The M -extrapolations satisfy this criteria. We show that a small weakening of the LU -extrapolations that preserves an ordering property between clocks also satisfies this criterion. In Section 4, we give an algorithm for ZP^a and describe the class of abstractions that give a polynomial complexity. Yet again, the M -extrapolations satisfy these criteria. We will see that a weakening of the LU -extrapolations that maintains some lower-bound information also satisfies this criterion.

2. NON-ZENONESS IS NP-HARD FOR LU -EXTRAPOLATIONS

We give a reduction from the 3SAT problem: given a 3CNF formula ϕ , we build an automaton \mathcal{A}_ϕ^{NZ} that has a non-Zeno run iff ϕ is satisfiable. The size of the automaton will be linear in the size of ϕ . We will then show that the abstract zone graph $ZG^{LU}(\mathcal{A}_\phi^{NZ})$ is isomorphic to \mathcal{A}_ϕ^{NZ} , thus completing the polynomial reduction from 3SAT to NZP^{LU} .

Automaton \mathcal{A}_ϕ^{NZ} . Let $P = \{p_1, \dots, p_k\}$ be a set of propositional variables and let $\phi = C_1 \wedge \dots \wedge C_n$ be a 3CNF formula with n clauses. We define the timed automaton \mathcal{A}_ϕ^{NZ} as follows. Its set of clocks X equals $\{x_1, \bar{x}_1, \dots, x_k, \bar{x}_k\}$. For a literal λ , let $cl(\lambda)$ denote the clock x_i when $\lambda = p_i$ and the clock \bar{x}_i when $\lambda = \neg p_i$. The set of states of \mathcal{A}_ϕ^{NZ} is $\{q_0, \dots, q_k, r_0, \dots, r_n\}$ with q_0 being the initial state. The transitions are as follows:

- for each proposition p_i we have transitions $q_{i-1} \xrightarrow{\{x_i\}} q_i$ and $q_{i-1} \xrightarrow{\{\bar{x}_i\}} q_i$,
- for each clause $C_m = \lambda_1^m \vee \lambda_2^m \vee \lambda_3^m$, $m = 1, \dots, n$, there are three transitions $r_{m-1} \xrightarrow{cl(\lambda) \leq 0} r_m$ for $\lambda \in \{\lambda_1^m, \lambda_2^m, \lambda_3^m\}$,
- transitions $q_k \rightarrow r_0$ and $r_n \rightarrow q_0$ with no guards and resets.

Figure 8 shows the automaton for the formula $(p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2 \vee p_3)$. The part from q_0 to q_3 encodes an assignment with the following convention: a reset of x_i represents $p_i \mapsto \text{true}$ and a reset of \bar{x}_i means $p_i \mapsto \text{false}$. Then, from r_0 to r_2 we check if the formula is satisfied by this guessed assignment.

The above formula is satisfied by every assignment that maps p_3 to *true*. Any path that encodes this assignment using the convention mentioned above should pick the transition $q_2 \xrightarrow{\{x_3\}} q_3$. Then, it has the possibility to follow transitions $r_0 \xrightarrow{x_3 \leq 0} r_1$ and $r_1 \xrightarrow{x_3 \leq 0} r_2$.

On any cycle containing these three transition, time can elapse (for instance in state q_0) since x_3 is reset before being checked for zero. Therefore, this assignment that makes the formula true corresponds to a non-Zeno run of \mathcal{A}_ϕ^{NZ} .

Conversely, consider the assignment $p_1 \mapsto false$, $p_2 \mapsto true$ and $p_3 \mapsto false$ that does not satisfy the formula. Take a cycle that resets \bar{x}_1 , x_2 and \bar{x}_3 according to the encoding of assignments. Then none of the clocks that are checked for zero on the transitions from r_0 to r_1 has been reset. Notice that these transitions come from the first clause in the formula that evaluates to *false* according to the assignment. To take a transition from r_0 , one of x_1 , \bar{x}_2 and x_3 must be zero and hence time cannot elapse in the path corresponding to this assignment.

Lemma 2.1 below states that if the formula is satisfiable, there exists a sequence of resets that allows time elapse in every loop. Conversely, if the formula is unsatisfiable, in every iteration of the loop, there is a zero-check that prevents time from elapsing.

Lemma 2.1. *A 3CNF formula ϕ is satisfiable iff \mathcal{A}_ϕ^{NZ} has a non-Zeno run.*

Proof. Let ϕ be a conjunction of n clauses C_1, \dots, C_n . Assume that ϕ is satisfiable. Then, there exists a variable assignment $\chi : P \mapsto \{true, false\}$ that evaluates ϕ to true. This entails that in every clause C_m there is a literal λ_m that evaluates to true with χ .

We will now build a non-Zeno run ρ of \mathcal{A}_ϕ^{NZ} using this variable assignment χ . Clearly, it should have the following sequence of states repeated infinitely often:

$$q_0 \rightarrow \dots q_k \rightarrow r_0 \rightarrow r_1 \rightarrow \dots r_n$$

Additionally, ρ satisfies the following conditions:

- from each configuration (q_{i-1}, v) for $i \in [1; k]$, ρ takes the transition $q_{i-1} \xrightarrow{\{x_i\}} q_i$ when $\chi(p_i) = true$ and the transition $q_{i-1} \xrightarrow{\{\bar{x}_i\}} q_i$ otherwise;
- from each configuration (r_{m-1}, v) for $m \in [1; n]$, ρ takes a transition $r_{m-1} \xrightarrow{cl(\lambda_m) \leq 0} r_m$ where λ_m is the literal evaluating to *true* with respect to χ in C_m ;
- and ρ lets 1 time unit elapse from each configuration with state r_n and moves to the state q_0 ; in all other states, there is no time elapse.

Note that as r_n occurs infinitely often, the run ρ is non-Zeno. It remains to prove that ρ is indeed a valid run of \mathcal{A}_ϕ^{NZ} . For this, we need to prove that all zero-checked transitions can be crossed regardless of the unit time elapse. Consider the part of ρ between two successive configurations with state r_n .

$$\dots (r_n, v) \xrightarrow{1} \dots \xrightarrow{\{cl(\lambda_m)\}} \dots (r_{m-1}, v'') \xrightarrow{cl(\lambda_m) \leq 0} (r_m, v'') \dots (r_n, v') \xrightarrow{1} \dots$$

By definition of ρ , λ_m is a literal that evaluates to *true* according to χ . Hence, clock $cl(\lambda_m)$ is reset in the corresponding $q_{j-1} \rightarrow q_j$ transition, before being checked for zero. As $cl(\lambda_m)$ is reset and since ρ does not elapse time in states other than r_n , we have $v''(cl(\lambda_j^m)) = 0$. This permits the transition from r_{m-1} to r_m for all $m \in [1; n]$ and shows that the run ρ exists.

For the other direction, consider a non-Zeno run ρ of \mathcal{A}_ϕ^{NZ} . Since ρ is non-Zeno, time elapses on infinitely many transitions in the run. Every infinite run of \mathcal{A}_ϕ^{NZ} visits a configuration with state r_n infinitely often. Consider two consecutive occurrences of r_n in ρ .

such that time elapses on some transition in the segment in between:

$$\dots(r_n, v) \rightarrow \dots(q_k, v') \rightarrow \dots(r_{m-1}, v'') \xrightarrow{cl(\lambda_m) \leq 0} (r_m, v'') \dots \rightarrow (r_n, v'') \dots$$

By construction, for each $i \in [1; k]$ either x_i or \bar{x}_i is reset on the segment from (r_n, v) to (q_k, v') . Let χ be the variable assignment that associates *true* to p_i when x_i is reset, and *false* otherwise, that is when \bar{x}_i is reset. We prove that χ satisfies ϕ .

Consider the transition $(r_{m-1}, v'') \xrightarrow{cl(\lambda_m) \leq 0} (r_m, v'')$. For the transition to be enabled, we need to have $v''(cl(\lambda_m)) = 0$. Let $(q_{j-1}, v_{j-1}) \rightarrow (q_j, v_j)$ be the transition that resets either $cl(\lambda_m)$ or $cl(\bar{\lambda}_m)$. Notice that time cannot elapse between (q_j, v_j) and (r_{m-1}, v'') . So the time elapse should have occurred between (r_n, v) to (q_{j-1}, v_{j-1}) . Thus it should be clock $cl(\lambda_m)$ that is reset in the transition $(q_{j-1}, v_{j-1}) \rightarrow (q_j, v_j)$. From the above definition of χ , we have λ_m evaluating to *true* with χ and hence C_m evaluates to *true* with χ too. This holds for all the clauses. This shows that ϕ is satisfiable with χ being the satisfying assignment. \square

The NP-hardness of NZP^{LU} then follows due to the small size of $ZG^{LU}(\mathcal{A}_\phi^{NZ})$.

Theorem 2.2. *The abstract zone graph $ZG^{LU}(\mathcal{A}_\phi^{NZ})$ is isomorphic to \mathcal{A}_ϕ^{NZ} . The non-Zenoness problem is NP-hard for abstractions Extra_{LU} and Extra_{LU}^+ .*

Proof. We first prove that $ZG^{LU}(\mathcal{A}_\phi^{NZ})$ is isomorphic to \mathcal{A}_ϕ^{NZ} . For every clock x , $L(x) = -\infty$, hence Extra_{LU} abstracts all the constraints $x_i - x_j \preccurlyeq_{ij} c_{ij}$ to $x_i - x_j < \infty$ except those of the form $x_0 - x_i \preccurlyeq_{0i} c_{0i}$ that are kept unchanged. Due to the guards in \mathcal{A}_ϕ^{NZ} , for every reachable zone Z in $ZG(\mathcal{A}_\phi^{NZ})$ we have $x_0 - x_i \leq 0$ (i.e. $x_i \geq 0$). Therefore $\text{Extra}_{LU}(Z)$ is the zone defined by $\bigwedge_{x \in X} x \geq 0$ which is $\mathbb{R}_{\geq 0}^X$. For each state of \mathcal{A}_ϕ^{NZ} , the zone $\mathbb{R}_{\geq 0}^X$ is the only reachable zone in $ZG^{LU}(\mathcal{A}_\phi^{NZ})$, hence showing the isomorphism.

NP-hardness then follows from Lemma 2.1. The result transfers to Extra_{LU}^+ thanks to Theorem 1.3. \square

Notice that the type of zero checks in \mathcal{A}_ϕ^{NZ} is crucial to Theorem 2.2. Replacing zero-checks of the form $x \leq 0$ by $x = 0$ does not modify the semantics of \mathcal{A}_ϕ^{NZ} . However, this yields $L(x) = 0$ for every clock x . Hence, the constraints of the form $x_i - x_j \leq 0$ are not abstracted: Extra_{LU} then preserves the ordering among the clocks. Each sequence of clock resets leading from q_0 to q_k yields a distinct ordering on the clocks. Thus, there are exponentially many LU-abstracted zones with state q_k . As a consequence, the polynomial reduction from 3SAT is lost. We indeed provide in Section 3 below an algorithm for detecting non-Zeno runs from $ZG^{LU}(\mathcal{A})$ that runs in polynomial time when $L(x) \geq 0$ for all clocks x . On the other hand, notice that changing $x = 0$ to $x \leq 0$ reduces the size of the abstract zone graph, in some cases, by an exponential amount. We will see in Section 5 how this has led to an improvement in the reachability analysis for timed automata.

3. FINDING NON-ZENO RUNS

Recall the non-Zenoness problem (NZP^a):

Given an automaton \mathcal{A} and its abstract zone graph $ZG^a(\mathcal{A})$, decide if \mathcal{A} has a non-Zeno run.

A standard solution to this problem involves adding one auxiliary clock to \mathcal{A} to detect non-Zenoness [19]. This solution was shown to cause an exponential blowup in [14]. In the same paper, a polynomial method has been proposed in the case of the Extra_M abstraction. We briefly recall this construction below.

An infinite run of the timed automaton could be Zeno due to two factors:

- *blocking clocks*: these are clocks bounded from above (i.e. $x \leq c$ for some $c > 0$) infinitely often in the run, but are reset only finitely many times,
- *zero checks*: these are guards of the form $x \leq 0$ or $x = 0$ that occur infinitely often in a manner that prevents time elapse in the run.

To solve NZP^a , the task is to find if there exists an infinite run in $ZG^a(\mathcal{A})$ that neither has blocking clocks nor zero-checks that prevent time-elapse. The method in [14] tackles these two problems as follows. Blocking clocks are handled by first detecting a maximal strongly connected component (SCC) of the zone graph and repeatedly discarding the transitions that bound some blocking clock until a non-trivial SCC with no such clocks is obtained. This algorithm runs in time polynomial for every abstraction. For zero checks, a *guessing zone graph* construction has been introduced to detect nodes where time can elapse.

3.1. Guessing zone graph $GZG^a(\mathcal{A})$. The necessary and sufficient condition for time elapse in a node inspite of zero-checks is to have every reachable zero-check from that node preceded by a corresponding reset (cf. Figure 9).

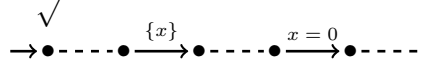


Figure 9: Time can elapse in the node $\sqrt{}$

Therefore, the aim is to check if there exists a node (q, Z) in $ZG^a(\mathcal{A})$ such that there is a path from (q, Z) back to itself in which every zero-check is preceded by a corresponding reset. This would instantiate to an infinite run of \mathcal{A} that can elapse time despite the zero-checks.

This is what the guessing zone graph construction achieves. The nodes of the guessing zone graph are triples (q, Z, Y) where $Y \subseteq X$ is a set of clocks. The sets Y are called the *guess sets*. Whenever a clock is reset, it is added to the guess set of the resulting node. A transition with a zero-check can be crossed only if the clock that is checked for zero is already present in the guess set, that is, if it was reset somewhere in the past. The guess set Y in a node (q, Z, Y) therefore gives the set of clocks that can potentially be checked for zero before being reset in a path starting from (q, Z, Y) . In particular, clocks that are not in Y cannot be checked for zero in the future before being reset. Hence, on a path from a node with an empty guess set, all the zero checks are preceded by the corresponding reset, and time can elapse in that node.

For a valuation v , we write $v \models (X - Y > 0)$ for the constraint saying that all the variables in $X - Y$ are greater than 0 in v , that is: $v \models \left(\bigwedge_{x \in (X - Y)} v(x) > 0 \right)$. For every transition $t = (q, g, R, q')$ of \mathcal{A} , $GZG^a(\mathcal{A})$ has a transition $(q, Z, Y) \Rightarrow_a^t (q', Z', Y')$ only if:

- there is a transition $(q, Z) \Rightarrow_a^t (q', Z')$ in $ZG^a(\mathcal{A})$;
- there is a valuation $v \in Z$ such that $v \models (X - Y > 0)$ and $v \models g$;
- and $Y' = Y \cup R$.

Observe that if the guess set is empty in a node, then the following transition can be taken by a valuation that has all clocks greater than zero. This shows that if there is a path from a node (q, Z, \emptyset) , zero-checks do not hinder time-elapse in this node. When a clock is reset, this is remembered in Y' . This in turn allows the clock to be checked for zero from (q', Z', Y') .

The guessing zone graph also contains special transitions:

- $(q, Z, Y) \Rightarrow_{\mathbf{a}}^{\tau} (q, Z, Y')$ with $Y' = \emptyset$ or $Y' = Y$.

Hence, from any node (q, Z, Y) , by taking a τ transition that leads to (q, Z, \emptyset) , one can non-deterministically check if there is a path from that node where every zero-check is preceded by a corresponding reset.

Figure 10 depicts a timed automaton \mathcal{A}_1 along with its zone graph $ZG^{\mathbf{a}}(\mathcal{A}_1)$ and the reachable part of its guessing zone graph $GZG^{\mathbf{a}}(\mathcal{A}_1)$ where τ -loops have been omitted. The loop that checks x for zero is disabled from node $(1, x = z, \emptyset)$ since x does not belong to the guess set. This indicates that it is not possible to let time elapse and then take this transition. Time can elapse in every node with an empty guess set (nodes with \emptyset as a third component) since, by construction, every zero check must be preceded by the corresponding reset. In particular, the cycle $(2, x - z \geq 1, \emptyset) \Rightarrow_{\mathbf{a}} (3, x - z \geq 1, \{z\}) \Rightarrow_{\mathbf{a}} (2, x - z \geq 1, \{z\}) \Rightarrow_{\mathbf{a}}^{\tau} (2, x - z \geq 1, \emptyset)$ is the suffix of a non-Zeno run.

It has been shown in [14] that the number of guess sets for every node (q, Z) reachable in $ZG^{\mathbf{a}}(\mathcal{A})$ is bound by $|X|+1$ when the abstraction \mathbf{a} is Extra_M . The case of other abstractions was not considered. The same construction does not give polynomial complexity even for Extra_M^+ . We first optimize this construction by considering an arbitrary abstraction.

3.2. Reduced guessing zone graph $rGZG^{\mathbf{a}}(\mathcal{A})$. The reduced guessing zone graph is a slight modification that restricts the guess sets to a subset of the set of clocks. A clock that is never checked for zero need not be remembered in sets Y . We restrict Y sets to only contain clocks that can indeed be checked for zero and we show that this is sound and complete for non-Zenoness.

We say that a clock x is *relevant* if there exists a guard $x \leq 0$ or $x = 0$ in the automaton. We denote the set of relevant clocks by $\text{Rl}(\mathcal{A})$. For a zone Z , let $\mathcal{C}_0(Z)$ denote the set of clocks x such that there exists a valuation $v \in Z$ with $v(x) = 0$. The clocks that can be checked for zero before being reset in a path from (q, Z) , lie in $\text{Rl}(\mathcal{A}) \cap \mathcal{C}_0(Z)$.

Definition 3.1 (Reduced guessing zone graph). Let \mathcal{A} be a timed automaton with clocks X . The *reduced guessing zone graph* $rGZG^{\mathbf{a}}(\mathcal{A})$ has nodes of the form (q, Z, Y) where (q, Z) is a node in $ZG^{\mathbf{a}}(\mathcal{A})$ and $Y \subseteq \text{Rl}(\mathcal{A}) \cap \mathcal{C}_0(Z)$. The initial node is $(q_0, Z_0, \text{Rl}(\mathcal{A}))$, with (q_0, Z_0) the initial node of $ZG^{\mathbf{a}}(\mathcal{A})$. The transitions are as follows:

- For $t = (q, g, R, q')$, there is a transition $(q, Z, Y) \Rightarrow_{\mathbf{a}}^t (q', Z', Y')$ with:

$$Y' = (Y \cup R) \cap \text{Rl}(\mathcal{A}) \cap \mathcal{C}_0(Z')$$

if there is $(q, Z) \Rightarrow_{\mathbf{a}}^t (q', Z')$ in $ZG^{\mathbf{a}}(\mathcal{A})$ and some valuation $v \in Z$ such that $v \models (\text{Rl}(\mathcal{A}) - Y) > 0$ and $v \models g$.

- A new auxiliary letter τ is introduced that adds transitions $(q, Z, Y) \Rightarrow_{\mathbf{a}}^{\tau} (q, Z, Y')$ for $Y' = \emptyset$ or $Y' = Y$.

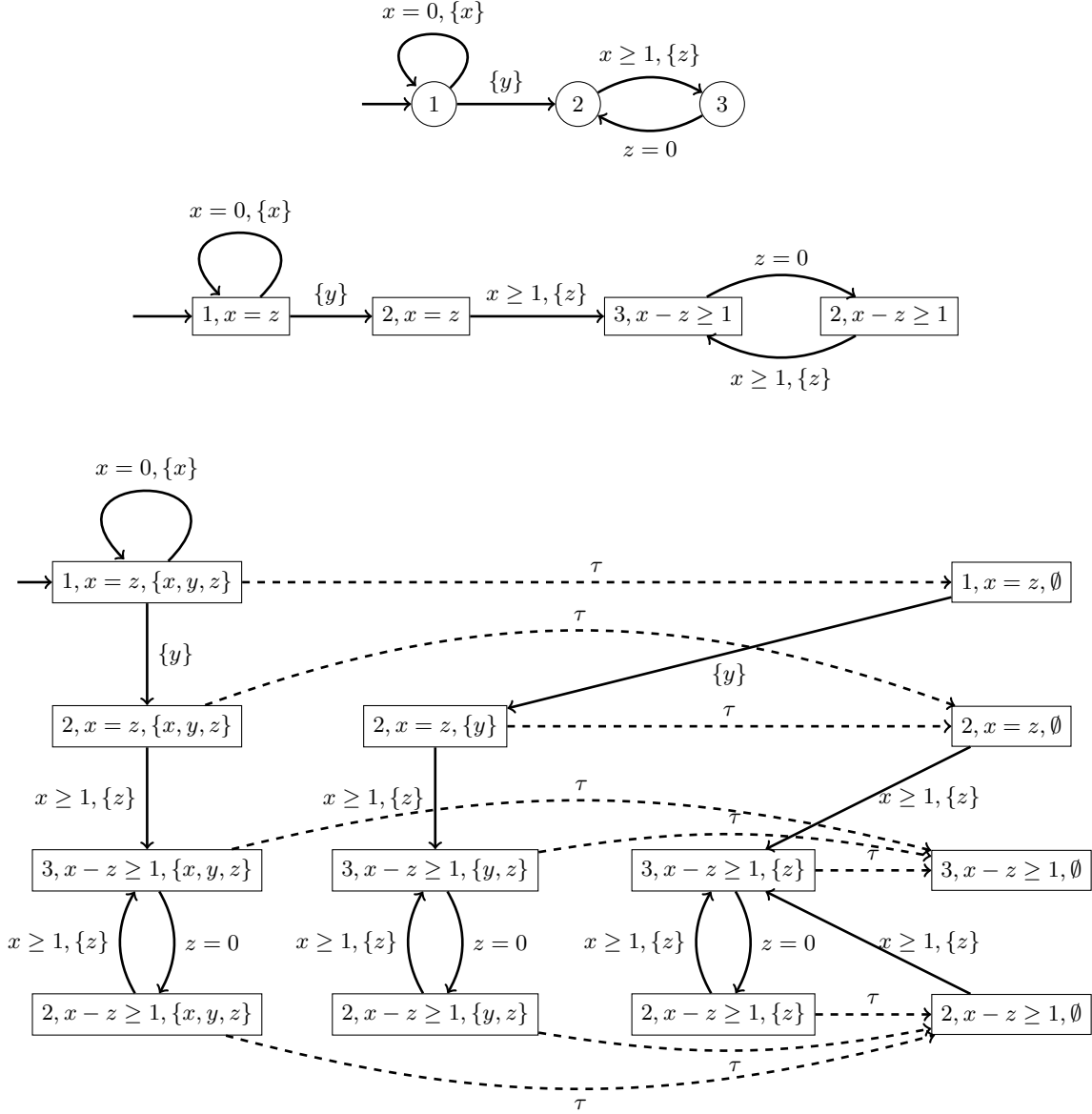


Figure 10: A timed automaton \mathcal{A}_1 (top), its zone graph $ZG^a(\mathcal{A}_1)$ (middle) and the reachable part of the guessing zone graph $GZG^a(\mathcal{A}_1)$ (bottom) with τ self-loops omitted for clarity.

Observe that similar to the case of the guessing zone graph, we require $v \models (\text{Rl}(\mathcal{A}) - Y) > 0$ and $v \models g$ for some $v \in Z$, a transition that checks $x \leq 0$ (or $x = 0$) is allowed from a node (q, Z, Y) only if $x \in Y$. Thus, from a node (q, Z, \emptyset) every reachable zero-check $x = 0$ should be preceded by a transition that resets x , and hence adds it to the guess set. Such a node is called *clear*. The presence of such nodes would ensure a time-elapse even in the presence of zero-checks.

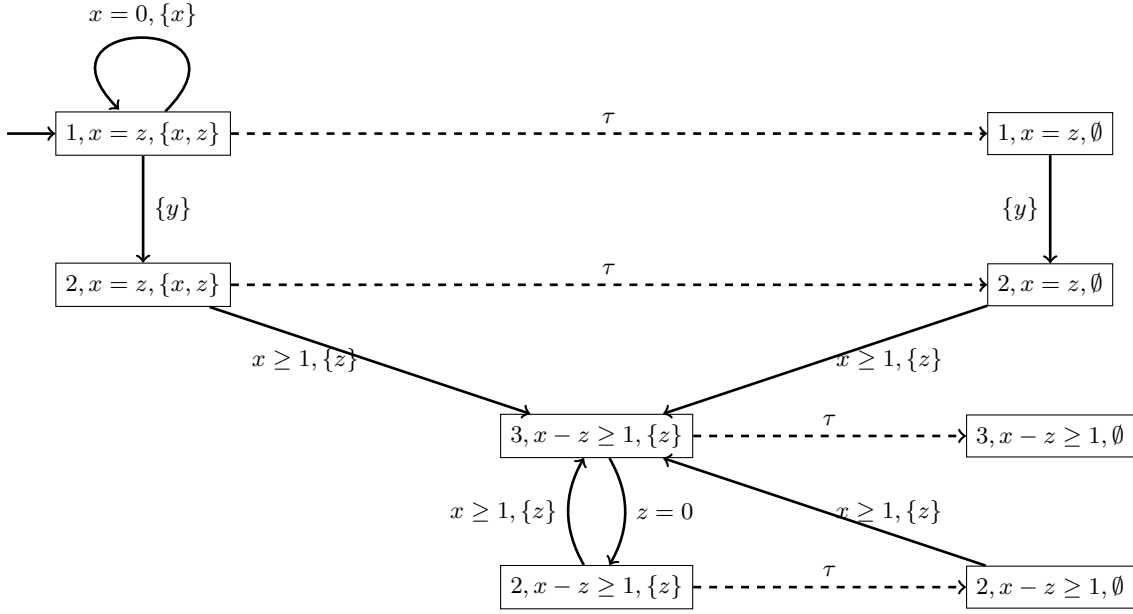


Figure 11: The reachable part of the reduced guessing zone graph $rGZG^\alpha(\mathcal{A}_1)$ of automaton \mathcal{A}_1 in Figure 10 (with τ self-loops omitted for clarity).

Figure 11 shows the reachable part of the reduced guessing zone graph $rGZG^\alpha(\mathcal{A}_1)$ for the automaton \mathcal{A}_1 in Figure 10. Notice that since $y \notin \text{Rl}(\mathcal{A}_1)$, the clock y is not added to the sets Y when it is reset. Observe also that clock x cannot belong to the guess sets in the nodes $(3, x - z \geq 1)$ and $(2, x - z \geq 1)$ as $x > 0$. The resulting reduced guessing zone graph $rGZG^\alpha(\mathcal{A}_1)$ is a lot smaller than the guessing zone graph $GZG^\alpha(\mathcal{A}_1)$ in Figure 10. It still contains all the information needed to detect non-Zeno runs.

Before we prove our result about the reduced guessing zone graph, we define some notions.

Definition 3.2. A node (q, Z, Y) of $rGZG^\alpha(\mathcal{A})$ is called *clear* if the third component is empty: $Y = \emptyset$. A variable x is *bounded* in a transition of $rGZG^\alpha$ if the guard of the transition implies $x \leq c$ for some constant c . A path of $rGZG^\alpha$ is said to be *blocked* if there is a variable that is bounded infinitely often and reset only finitely often by the transitions on the path. Otherwise the path is called *unblocked*.

An unblocked path says that there are no blocking clocks to bound time and clear nodes suggest that inspite of zero-checks that might possibly occur in the future, time can still elapse. We get the following theorem.

Proposition 3.3. *A timed automaton \mathcal{A} has a non-Zeno run iff there exists an unblocked path in $rGZG^\alpha(\mathcal{A})$ visiting a clear node infinitely often.*

The proof of Proposition 3.3 is in the same lines as for the guessing zone graph in [14]. It follows from the following two lemmas.

Lemma 3.4. *If \mathcal{A} has a non-Zeno run, then in $rGZG^\alpha(\mathcal{A})$ there is an unblocked path visiting a clear node infinitely often.*

Proof. Let ρ be a non-Zeno run of \mathcal{A} :

$$(q_0, v_0) \xrightarrow{\delta_0, t_0} (q_1, v_1) \xrightarrow{\delta_1, t_1} \dots$$

Since \mathfrak{a} is complete, ρ is an instantiation of a path π in $ZG^{\mathfrak{a}}(\mathcal{A})$:

$$(q_0, Z_0) \Rightarrow_{\mathfrak{a}}^{t_0} (q_1, Z_1) \Rightarrow_{\mathfrak{a}}^{t_1} \dots$$

Let σ be the following sequence of transitions:

$$(q_0, Z_0, Y_0) \Rightarrow_{\mathfrak{a}}^r (q_0, Z_0, Y'_0) \Rightarrow_{\mathfrak{a}}^{t_0} (q_1, Z_1, Y_1) \Rightarrow_{\mathfrak{a}}^r (q_1, Z_1, Y'_1) \Rightarrow_{\mathfrak{a}}^{t_1} \dots$$

where $Y_0 = \text{Rl}(\mathcal{A})$, Y_i is determined by the transition relation in $rGZG^{\mathfrak{a}}(\mathcal{A})$, and $Y'_i = Y_i$ unless $\delta_i > 0$ when we put $Y'_i = \emptyset$.

Since ρ is non-Zeno, there are infinitely many i such that $\delta_i > 0$, hence σ contains infinitely many clear nodes with $Y'_i = \emptyset$. From the non-Zenoness of ρ , we also get that σ is unblocked.

Now, it remains to show that σ is indeed a path in $rGZG^{\mathfrak{a}}(\mathcal{A})$. For this we need to see that every transition $(q_i, Z_i, Y'_i) \Rightarrow_{\mathfrak{a}}^{t_i} (q_{i+1}, Z_{i+1}, Y_{i+1})$ is realizable from a valuation $v \in Z_i$ such that both $v \models (\text{Rl}(\mathcal{A}) - Y'_i) > 0$ and $v \models g_i$ where g_i is the guard of t_i . We prove this by an induction on the run. As by the definition of ρ , $v_i + \delta_i \models g_i$ for all $i \geq 0$, we only need to prove that $v_i + \delta_i \models (\text{Rl}(\mathcal{A}) - Y'_i) > 0$. This is clearly true for valuation $v_0 + \delta_0 \in Z_0$.

Assume that $v_i + \delta_i \models (\text{Rl}(\mathcal{A}) - Y'_i) > 0$. We now prove that $v_{i+1} + \delta_{i+1} \models (\text{Rl}(\mathcal{A}) - Y'_{i+1}) > 0$. Firstly, observe that $Y'_{i+1} = (Y'_i \cup R_i) \cap \mathcal{C}_0(Z_{i+1}) \cap \text{Rl}(\mathcal{A})$. Therefore a clock $x \in \text{Rl}(\mathcal{A}) - Y'_{i+1}$ either belongs to $\text{Rl}(\mathcal{A}) - Y'_i$ in which case it is greater than 0 by induction hypothesis, or otherwise we have $x \in Y'_i$ but $x \notin \mathcal{C}_0(Z_{i+1})$. By the definition of $\mathcal{C}_0(Z_{i+1})$, all valuations $v \in Z_{i+1}$ satisfy $v(x) > 0$ and so in particular, $v_{i+1}(x) > 0$. This leads to $v_{i+1} \models (\text{Rl}(\mathcal{A}) - Y_{i+1}) > 0$ which easily extends to $v_{i+1} + \delta_{i+1} \models (\text{Rl}(\mathcal{A}) - Y'_{i+1}) > 0$. \square

Lemma 3.5. *Suppose $rGZG^{\mathfrak{a}}(\mathcal{A})$ has an unblocked path visiting a clear node infinitely often then \mathcal{A} has a non-Zeno run.*

Proof. The proof follows the same lines as the proof of Lemma 6 in [14] with the additional information that for all clocks x that do not belong to $\text{Rl}(\mathcal{A})$, we have $g \wedge (x > 0)$ consistent for every guard g .

Let $\pi : (q_0, Z_0, Y_0) \Rightarrow_{\mathfrak{a}}^{t_0} \dots$ be the unblocked path of $rGZG^{\mathfrak{a}}(\mathcal{A})$ that visits a clear node infinitely often. Since \mathfrak{a} is sound, take an instantiation $\rho : (q_0, v_0) \xrightarrow{\delta_0, t_0} \dots$ of \mathcal{A} . If ρ is non-Zeno, we are done.

Assume that ρ is Zeno. It has a suffix where less than 1/2 time unit elapses. Let X^r denote the set of clocks that are reset infinitely often on ρ . We can thus find an index m such that $v_n(x) < 1/2$ for all $x \in X^r$ and for all $n \geq m$. Take indices $i, j \geq m$ such that $Y_i = Y_j = \emptyset$ and all clocks in X^r are reset between i and j . We look at the sequence $(q_i, v_i) \xrightarrow{\delta_i, t_i} \dots (q_j, v_j)$ and claim that every sequence of the form

$$(q_i, v'_i) \xrightarrow{\delta_i, t_i} (q_{i+1}, v'_{i+1}) \xrightarrow{\delta_{i+1}, t_{i+1}} \dots (q_j, v'_j)$$

is a part of a run of \mathcal{A} provided there is $\zeta \in \mathbb{R}_{\geq 0}$ such that the following three conditions hold for all $k = i, \dots, j$:

- (1) $\nu'_k(x) = \nu_k(x) + \zeta + 1/2$ for all $x \notin X^r$,
- (2) $\nu'_k(x) = \nu_k(x) + 1/2$ if $x \in X^r$ and x has not been reset between i and k .
- (3) $\nu'_k(x) = \nu_k(x)$ otherwise, i.e., when $x \in X^r$ and x has been reset between i and k .

It is easy to see that the run obtained by replacing every such $i - j$ interval of ρ by the above sequence gives a non-Zeno run, since a $1/2$ time unit has been elapsed infinitely often.

We now show that the above is indeed a valid run of \mathcal{A} . For this we need to first show that $v'_k + \delta_k$ satisfies the guard in t_k . Let g be the guard.

For $x \notin X^r$, from the assumption that ρ is unblocked, we know that g could only be of the form $x > c$ or $x \geq c$. So $v'_k(x)$ clearly satisfies g . If $x \in X^r$ and is reset between i and k , $v'_k(x) = v_k(x)$ and so we are done. Consider the case when $x \in X^r$ and is not reset between i and k . Observe that $x \notin Y_k$. This is because $Y_i = \emptyset$, and then only variables that are reset are added to Y . Since x is not reset between i and k , it cannot be in Y_k . By definition of transitions in $rGZG^a(\mathcal{A})$, if $x \in \text{Rl}(\mathcal{A})$ this means that $g \wedge (x > 0)$ is consistent. But for $x \notin \text{Rl}(\mathcal{A})$ by definition, $g \wedge (x > 0)$ is consistent. We have that $0 \leq (v_k + \delta_k)(x) < 1/2$ and $1/2 \leq (v'_k + \delta_k)(x) < 1$. So $v'_k + \delta_k$ satisfies all the constraints in g concerning x as $v_k + \delta_k$ does.

It can also be seen that the valuation obtained from v'_k by resetting the clocks in transition t_k is the valuation v'_{k+1} . \square

3.3. Polynomial algorithms for NZP^a. Since we have a node in $rGZG^a(\mathcal{A})$ for every (q, Z) in $ZG^a(\mathcal{A})$ and every subset Y of $\text{Rl}(\mathcal{A})$, it can in principle be exponentially bigger than $ZG^a(\mathcal{A})$. Below, we see that depending on abstraction \mathbf{a} , not all subsets Y need to be considered. Let us first define the notion of a zone ordering clocks.

Definition 3.6. Let X' be a subset of X . We say that a zone Z *orders the clocks in X'* if for all clocks $x, y \in X'$, Z implies that at least one of $x \leq y$ or $y \leq x$ hold, that is either all valuations $v \in Z$ satisfy $v(x) \leq v(y)$ or all valuations $v \in Z$ satisfy $v(y) \leq v(x)$.

Definition 3.7 (Weakly order-preserving abstractions). An abstraction \mathbf{a} *weakly preserves orders* if for all clocks $x, y \in \text{Rl}(\mathcal{A}) \cap \mathcal{C}_0(Z)$, $Z \models x \leq y$ iff $\mathbf{a}(Z) \models x \leq y$.

It has been observed in [14] that all the zones that are reachable in the unabstracted zone graph $ZG(\mathcal{A})$ order the entire set of clocks X . Assume that \mathbf{a} weakly preserves orders, then for every reachable node (q, Z, Y) in $rGZG^a(\mathcal{A})$, the zone Z orders the clocks in $\text{Rl}(\mathcal{A}) \cap \mathcal{C}_0(Z)$. We now show that Y is downward closed with respect to this order given by Z : for clocks $x, y \in \text{Rl}(\mathcal{A}) \cap \mathcal{C}_0(Z)$, if $Z \models x \leq y$ and $y \in Y$, then $x \in Y$. This entails that there are at most $|\text{Rl}(\mathcal{A})| + 1$ downward closed sets to consider, thus giving a polynomial complexity.

Proposition 3.8. *Let \mathcal{A} be a timed automaton. If \mathbf{a} weakly preserves orders, then the reachable part of $rGZG^a(\mathcal{A})$ is $\mathcal{O}(|\text{Rl}(\mathcal{A})|)$ bigger than the reachable part of $ZG^a(\mathcal{A})$.*

Proof. We prove by induction on the transitions in $rGZG^a(\mathcal{A})$ that for every reachable node (q, Z, Y) the set Y is downward closed with respect to the order on the clocks in $\text{Rl}(\mathcal{A}) \cap \mathcal{C}_0(Z)$ implied by Z . This is true for the initial node $(q_0, Z_0, \text{Rl}(\mathcal{A}))$.

Now, assume that this is true for (q, Z, Y) . Take a transition $(q, Z, Y) \xRightarrow{\mathbf{a}}^t (q', Z', Y')$ with $t = (q, g, R, q')$. By definition, $Y' = (Y \cup R) \cap \text{Rl}(\mathcal{A}) \cap \mathcal{C}_0(Z')$. Suppose $Z' \models x \leq y$ for some $x, y \in \text{Rl}(\mathcal{A}) \cap \mathcal{C}_0(Z')$ and suppose $y \in Y'$. This could mean $y \in Y$ or $y \in R$. If $y \in R$, then x is also in R since $Z' \models x \leq y$. If $y \notin R$ then we get $y \in Y$ and $Z \models x \leq y$. By hypothesis that Y is downward closed, $x \in Y$. In both cases $x \in Y'$. \square

The following lemma shows that the M -extrapolations weakly preserve orders. Hence, $rGZG^M(\mathcal{A})$ yields a polynomial algorithm for NZP^M . Thanks to the reduction of the guessing zone graph to the relevant clocks, this algorithm is more efficient than the algorithm in [14] even while using the same abstraction.

Theorem 3.9. *The abstractions Extra_M and Extra_M^+ weakly preserve orders. The non-Zenoness problem is solved in polynomial time for Extra_M and Extra_M^+ .*

Proof. It has been proved in [14] that Extra_M weakly preserves orders. We now prove this for Extra_M^+ . Firstly note that for a clock x in $\text{Rl}(\mathcal{A})$ we have $M(x) \geq 0$. Moreover if $x \in \mathcal{C}_0(Z)$ we have that Z is consistent with $x \leq 0$. Hence, for a clock $x \in \text{Rl}(\mathcal{A}) \cap \mathcal{C}_0(Z)$, Z is consistent with $x \leq M(x)$. Therefore, by definition, $\text{Extra}_M^+(Z)$ restricted to clocks in $\text{Rl}(\mathcal{A}) \cap \mathcal{C}_0(Z)$ is identical to $\text{Extra}_M(Z)$ restricted to the same set of clocks. Since Extra_M weakly preserves orders, we get that Extra_M^+ weakly preserves orders too.

The algorithm in Proposition 3.3 is thus polynomial for Extra_M and Extra_M^+ by Proposition 3.8. \square

However, the polynomial complexity is not preserved by the coarser LU -extrapolations.

Theorem 3.10. *The abstractions Extra_{LU} and Extra_{LU}^+ do not weakly preserve orders. The non-Zenoness problem is NP-complete for Extra_{LU} and Extra_{LU}^+ .*

Proof. From the definition of Extra_{LU} we have that all constraints of the form $x_i - x_j \preceq_{ij} c_{ij}$ are abstracted to $x_i - x_j < \infty$ when $L(x_i) = -\infty$. Thus, information about the relative ordering between x_i and x_j is lost. This shows that Extra_{LU} does not weakly preserve orders when $L(x) = -\infty$ for relevant clocks. This also holds for Extra_{LU}^+ by Theorem 1.3.

NP hardness is proven in Theorem 2.2. It remains to discuss NP membership. Let N be the number of nodes in $ZG^{LU}(\mathcal{A})$. Let us non-deterministically choose a node (q, Z) . We assume that (q, Z) is reachable as this can be checked in polynomial time on $ZG^{LU}(\mathcal{A})$.

We augment (q, Z) with an empty guess set of clocks. From the node (q, Z, \emptyset) , we non-deterministically simulate a path π of the (non-reduced) guessing zone graph [14] obtained from Definition 3.1 with $\text{Rl}(\mathcal{A}) = X$ and $\mathcal{C}_0(Z) = X$ for every zone Z . We avoid taking τ transitions on this path. This ensures that the guess sets accumulate all the resets on π . During the simulation, we also keep track of a separate set U containing all the clocks that are bounded from above on a transition in π .

We write \Rightarrow_a^* to denote the transitive closure of \Rightarrow_a . If during the simulation one reaches a node (q, Z, Y) such that $U \subseteq Y$, then we have a cycle $(q, Z, \emptyset) \Rightarrow_a^* (q, Z, Y) \Rightarrow_a^\tau (q, Z, \emptyset)$ that is unblocked and that visits a clear node infinitely often. Also, since (q, Z) is reachable in $ZG^{LU}(\mathcal{A})$, (q, Z, X) is reachable in the guessing zone graph. Then (q, Z, \emptyset) is reachable from (q, Z, X) with a τ transition. From [14] and from the fact that Extra_{LU} and Extra_{LU}^+ are sound and complete [3] we get a non-Zeno run of \mathcal{A} .

Notice that it is sufficient to simulate $N \times (|X| + 1)$ transitions since we can avoid visiting a node (q', Z', Y') twice in π . \square

3.4. Modified LU -extrapolations for polynomial NZP^a . The LU -extrapolations do not weakly preserve orders in zones due to relevant clocks with $L(x) = -\infty$ and $U(x) \geq 0$. We show that this is the only reason for NP-hardness. We slightly modify Extra_{LU} to get an abstraction Extra_{LU}^- that is coarser than Extra_M , but it still weakly preserves orders.

Definition 3.11 (Weak L bounds). Let \mathcal{A} be a timed automaton. Given the bounds $L(x)$ and $U(x)$ for every clock $x \in X$, the *weak lower bound* \bar{L} is given by: $\bar{L}(x) = 0$ if $x \in \text{RI}(\mathcal{A})$, $L(x) = -\infty$ and $U(x) \geq 0$, and $\bar{L}(x) = L(x)$ otherwise.

We denote $\text{Extra}_{\bar{L}U}$ the Extra_{LU} abstraction obtained by choosing \bar{L} instead of L . Notice that $\text{Extra}_{\bar{L}U}$ and Extra_{LU} coincide when zero-checks are written $x = 0$ instead of $x \leq 0$ in the automaton. By definition of Extra_{LU} and Proposition 3.8, we get the following.

Theorem 3.12. *The abstraction $\text{Extra}_{\bar{L}U}$ weakly preserves orders. The non-Zenoness problem is solved in polynomial time for $\text{Extra}_{\bar{L}U}$.*

$\text{Extra}_{\bar{L}U}$ coincides with Extra_{LU} for a wide class of automata. For instance, when the automaton does not have a zero-check, $\text{Extra}_{\bar{L}U}$ is exactly Extra_{LU} , and the existence of a non-Zeno run can be decided in polynomial time. For some automata however, the zone graph obtained with $\text{Extra}_{\bar{L}U}$ is exponentially bigger than the zone graph obtained with Extra_{LU} . This is for instance the case for the automaton \mathcal{A}_ϕ^{NZ} used to prove NP-hardness of NZP^{LU} in Section 2. Similar to $\text{Extra}_{\bar{L}U}$ we can define $\text{Extra}_{\bar{L}U}^\pm$ which again weakly preserves orders and yield a polynomial algorithm to solve the non-Zenoness problem.

4. THE ZENONESS PROBLEM

In this section we consider the Zenoness problem (ZP^a):

Given an automaton \mathcal{A} and its abstract zone graph $ZG^a(\mathcal{A})$, decide if \mathcal{A} has a Zeno run.

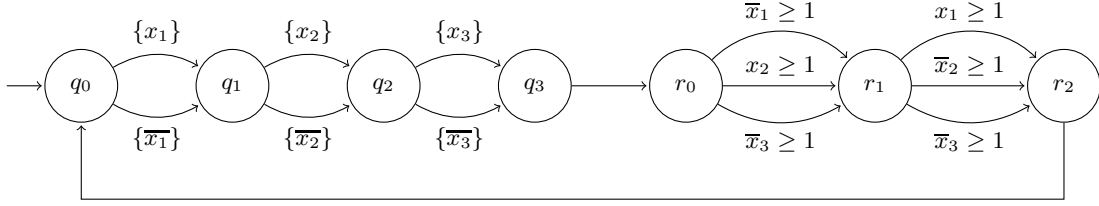
As in the case of non-Zenoness, this problem turns out to be NP-complete when the abstraction operator \mathbf{a} is Extra_{LU} . We subsequently give the hardness proof by providing a reduction from 3SAT.

4.1. Reducing 3SAT to ZP^a with abstraction Extra_{LU} . Let $P = \{p_1, \dots, p_k\}$ be a set of propositional variables. Let $\phi = C_1 \wedge \dots \wedge C_n$ be a 3CNF formula with n clauses. Each clause C_m , $m = 1, 2, \dots, n$ is a disjunction of three literals λ_1^m, λ_2^m and λ_3^m . We construct in polynomial time an automaton \mathcal{A}_ϕ^Z and its zone graph $ZG^{LU}(\mathcal{A}_\phi^Z)$ such that \mathcal{A}_ϕ^Z has a Zeno run iff ϕ is satisfiable, thus proving the NP-hardness.

The automaton \mathcal{A}_ϕ^Z has clocks $\{x_1, \bar{x}_1, \dots, x_k, \bar{x}_k\}$ with x_i and \bar{x}_i corresponding to the literals p_i and $\neg p_i$ respectively. We denote the clock associated to a literal λ by $cl(\lambda)$. The set of states of \mathcal{A}_ϕ^Z is given by $\{q_0, q_1, \dots, q_k\} \cup \{r_0, r_1, r_2, \dots, r_n\}$ with q_0 being the initial state. The transitions are as follows:

- transitions $q_{i-1} \xrightarrow{\{x_i\}} q_i$ and $q_{i-1} \xrightarrow{\{\bar{x}_i\}} q_i$ for $i = 1, 2, \dots, k$,
- a transition $q_k \rightarrow r_0$ with no guards and resets,
- for each clause C_m there are three transitions $r_{m-1} \xrightarrow{cl(\neg\lambda) \geq 1} r_m$ for each literal $\lambda \in \{\lambda_1^m, \lambda_2^m, \lambda_3^m\}$,
- a transition $r_n \rightarrow q_0$ with no guards and resets. This transition creates a cycle in \mathcal{A}_ϕ^Z .

As an example, Figure 12 shows the automaton for the formula $(p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2 \vee p_3)$. Observe that the transitions $r_{m-1} \xrightarrow{cl(\neg\lambda) \geq 1} r_m$ check if the clock corresponding to the *negation* of λ is greater than 1. That is, $cl(\neg\lambda) = cl(\lambda)$.

Figure 12: \mathcal{A}_ϕ^Z for $\phi = (p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_2 \vee p_3)$

Clearly, \mathcal{A}_ϕ^Z can be constructed from ϕ in $\mathcal{O}(|\phi|)$ time. We now show that ϕ is satisfiable iff \mathcal{A}_ϕ^Z has a Zeno run.

Lemma 4.1. *A 3CNF formula ϕ is satisfiable iff \mathcal{A}_ϕ^Z has a Zeno run.*

Proof. For the left-to-right direction, suppose that ϕ is satisfiable. Then there exists a variable assignment $\chi : P \mapsto \{true, false\}$ that evaluates ϕ to true. We now build the Zeno run of \mathcal{A}_ϕ^Z using χ .

Pick an infinite run ρ of \mathcal{A}_ϕ^Z . Clearly, it should have the following sequence of states repeated infinitely often:

$$q_0 \rightarrow \dots q_k \rightarrow r_0 \rightarrow r_1 \rightarrow \dots r_n \quad (4.1)$$

We choose the transitions for ρ that allow time elapse only by a finite amount. If $\chi(p_i) = true$, then we put $q_{i-1} \xrightarrow{\{x_i\}} q_i$ wherever $q_{i-1} \rightarrow q_i$ occurs in ρ . Otherwise $\chi(p_i) = false$ and we put $q_{i-1} \xrightarrow{\{\bar{x}_i\}} q_i$. We now need to choose the transitions $r_{m-1} \rightarrow r_m$ for $m = 1, \dots, n$. Since χ is a satisfying assignment, every clause C_m has a literal λ that evaluates to true with χ . We choose the corresponding transition $r_{m-1} \xrightarrow{cl(-\lambda) \geq 1} r_m$. Observe that if λ evaluates to true, it implies that $cl(\lambda)$ was reset in one of the $q_i \rightarrow q_{i+1}$ transitions but not $cl(-\lambda)$.

Therefore, the above construction yields a sequence of transitions with the property that all clocks that are reset are never checked for greater than 1. This sequence can be taken by elapsing 1 time unit in the very first state, and then subsequently elapsing no time at all, thus giving a Zeno run in \mathcal{A}_ϕ^Z .

We now prove the right-to-left direction. Let ρ be an infinite Zeno run of \mathcal{A}_ϕ^Z . An infinite run should repeat the sequence of states given in (4.1). Since ρ is Zeno, it has a suffix ρ^s such that for every clock x that is reset in ρ^s , $x \geq 1$ never occurs in the transitions of ρ^s . This is because if every suffix of ρ contains a clock that is both reset and checked for greater than 1, this would mean that there is a time elapse of one time unit occurring infinitely often, contradicting the hypothesis that ρ is Zeno.

Consider a segment $S = q_0 \rightarrow \dots q_n \rightarrow r_0 \rightarrow r_1 \rightarrow \dots r_k$ in ρ^s . We construct a satisfying assignment $\chi : P \mapsto \{true, false\}$ for ϕ from S .

- if S contains $q_{i-1} \xrightarrow{\{x_i\}} q_i$ then set $\chi(p_i) = true$
- otherwise, it implies that S contains $q_{i-1} \xrightarrow{\{\bar{x}_i\}} q_i$ in which case we set $\chi(p_i) = false$.

This shows that for a literal λ , if $cl(\lambda)$ is reset in S , then $\chi(\lambda) = true$. From the property of ρ^s that no clock that is reset is checked in a guard, for every transition $r_{m-1} \xrightarrow{cl(-\lambda) \geq 1} r_m$

in S , it is clock $cl(\lambda)$ that is reset and hence $\chi(\lambda) = true$. By construction of \mathcal{A}_ϕ^Z , λ is a literal in C_m . Therefore, we get a literal that is true in every clause evaluating ϕ to true. \square

It remains to show that $ZG^{LU}(\mathcal{A}_\phi^Z)$ can also be calculated in polynomial time from \mathcal{A}_ϕ^Z . We indeed note that the size of the $ZG^{LU}(\mathcal{A}_\phi^Z)$ is the same as that of the automaton. That will conclude the proof that a polynomial algorithm for ZP^{LU} yields a polynomial algorithm for the 3SAT problem.

Proposition 4.2. *The zone graph $ZG^{LU}(\mathcal{A}_\phi^Z)$ is isomorphic to \mathcal{A}_ϕ^Z . The Zenoness problem is NP-hard for Extra_{LU} and Extra_{LU}^+ .*

Proof. By looking at the guards in the transitions, we get that for each clock x , $L(x) = 1$ and $U(x) = -\infty$. The initial node of the zone graph $ZG^{LU}(\mathcal{A}_\phi^Z)$ is $(q_0, \text{Extra}_{LU}(Z_0))$ where Z_0 is the set of valuations given by $(x_1 \geq 0) \wedge (x_1 = \overline{x_1} = \dots = x_k = \overline{x_k})$. By definition, since for each clock x , $U(x) = -\infty$, we have $\text{Extra}_{LU}(Z_0) = \mathbb{R}_{\geq 0}^X$, the non-negative half-space.

On taking a transition with a guard $x \geq 1$ from $\mathbb{R}_{\geq 0}^X$, we come to a zone $\mathbb{R}_{\geq 0}^X \wedge x \geq 1$. However, since $U(x) = -\infty$, $\text{Extra}_{LU}(\mathbb{R}_{\geq 0}^X \wedge x \geq 1)$ gives back $\mathbb{R}_{\geq 0}^X$. Same for transitions that reset a clock. It follows that $ZG^{LU}(\mathcal{A}_\phi^Z)$ is isomorphic to \mathcal{A}_ϕ^Z . This extends to Extra_{LU}^+ by Theorem 1.3. Then NP-hard immediately follows from Lemma 4.1. \square

In the next section, we provide an algorithm for the zenoness problem ZP^a and give conditions on abstraction \mathbf{a} for the solution to be polynomial.

4.2. Finding Zeno paths. We say that a transition is *lifting* if it has a guard that implies $x \geq 1$ for some clock x . The idea is to find if there exists a run of an automaton \mathcal{A} in which every clock x that is reset infinitely often is lifted only finitely many times, ensuring that the run is Zeno. This amounts to checking if there exists a cycle in $ZG(\mathcal{A})$ where every clock that is reset is not lifted. Observe that when $(q, Z) \xrightarrow{x \geq c} (q', Z')$ is a transition of $ZG(\mathcal{A})$, then Z' remembers that x has been lifted to a value bigger than c , that is to say Z' entails $x \geq c$. Therefore, if a node (q, Z) is part of a cycle of our required form, then in particular, all the clocks that are greater than 1 in Z should not be reset in the cycle.

Based on the above intuition, our solution begins with computing the zone graph on-the-fly. At some node (q, Z) the algorithm non-deterministically guesses that this node is part of a cycle that yields a zeno run. This node transits to what we call the *slow mode*. In this mode, a reset of x in a transition is allowed from (q', Z') only if Z' is consistent with $x < 1$: there is at least one valuation $v \in Z'$ that has $v(x) < 1$.

Before we define our construction formally, recall that we would be working with the abstract zone graph $ZG^a(\mathcal{A})$ and not $ZG(\mathcal{A})$. Therefore for our solution to work, the abstraction operator \mathbf{a} should remember the fact that a clock has a value greater than 1. For an automaton \mathcal{A} over the set of clocks X , let $\text{Lf}(\mathcal{A})$ denote the set of clocks that appear in a lifting transition of \mathcal{A} .

Definition 4.3 (Lift-safe abstractions). An abstraction \mathbf{a} is called *lift-safe* if for every zone Z and for every clock $x \in \text{Lf}(\mathcal{A})$, $Z \models x \geq 1$ iff $\mathbf{a}(Z) \models x \geq 1$.

We are now in a position to define our *slow zone graph* construction to decide if an automaton has a Zeno run.

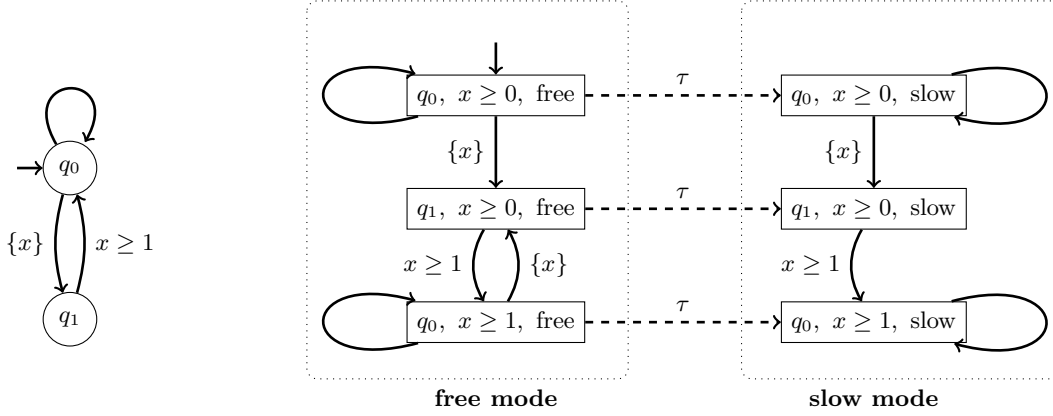


Figure 13: A timed automaton (left) and corresponding slow zone graph (right).

Definition 4.4 (Slow zone graph). Let \mathcal{A} be a timed automaton over the set of clocks X . Let \mathfrak{a} be a lift-safe abstraction. The *slow zone graph* $\mathcal{SZG}^{\mathfrak{a}}(\mathcal{A})$ has nodes of the form (q, Z, l) where $l = \{\text{free}, \text{slow}\}$. The initial node is (q_0, Z_0, free) where (q_0, Z_0) is the initial node of $ZG^{\mathfrak{a}}(\mathcal{A})$. For every transition $(q, Z) \Rightarrow_{\mathfrak{a}}^t (q', Z')$ in $ZG^{\mathfrak{a}}(\mathcal{A})$ with $t = (q, g, R, q')$, we have the following transitions in $\mathcal{SZG}^{\mathfrak{a}}(\mathcal{A})$:

- a transition $(q, Z, \text{free}) \Rightarrow_{\mathfrak{a}}^t (q', Z', \text{free})$,
- a transition $(q, Z, \text{slow}) \Rightarrow_{\mathfrak{a}}^t (q', Z', \text{slow})$ if for all clocks $x \in R$, $Z \wedge g$ is consistent with $x < 1$,

A new letter τ is introduced that adds transitions $(q, Z, \text{free}) \Rightarrow_{\mathfrak{a}}^{\tau} (q, Z, \text{slow})$.

A node of the form (q, Z, slow) is said to be a *slow node*. A path of $\mathcal{SZG}^{\mathfrak{a}}(\mathcal{A})$ is said to be *slow* if it has a suffix consisting entirely of slow nodes. The τ -transitions take a node (q, Z) from the *free* mode to the *slow* mode. Note that the transitions of the slow mode are constrained further. Figure 13 shows an example of an automaton and corresponding slow zone graph. The free mode is identical to the zone graph of the automaton. However, in the slow mode, the transition $q_0 \xrightarrow{\{x\}} q_1$ is not allowed from node $(q_0, x \geq 1, \text{slow})$ since x has been lifted. Hence, the only infinite paths in the slow mode instantiate the loop on state q_0 which correspond to the zeno runs of the automaton. The τ transitions allow to non deterministically guess a node which has a slow path.

The correctness follows from the fact that there is a cycle in $\mathcal{SZG}^{\mathfrak{a}}(\mathcal{A})$ consisting entirely of slow nodes iff \mathcal{A} has a Zeno run. This is detailed in the following two lemmas.

Lemma 4.5. *If \mathcal{A} has a Zeno run, then there exists an infinite slow path in $\mathcal{SZG}^{\mathfrak{a}}(\mathcal{A})$.*

Proof. Let ρ be a Zeno run of \mathcal{A} :

$$(q_0, v_0) \xrightarrow{\delta_0, t_0} (q_1, v_1) \xrightarrow{\delta_1, t_1} \dots$$

Let π be the corresponding path in $ZG^{\mathfrak{a}}(\mathcal{A})$:

$$(q_0, Z_0) \Rightarrow_{\mathfrak{a}}^{t_0} (q_1, Z_1) \Rightarrow_{\mathfrak{a}}^{t_1} \dots$$

We construct an infinite slow path in $\mathcal{SZG}^a(\mathcal{A})$ from the path π . Let X^l be the set of clocks that are lifted infinitely often in π and let X^r be the set of clocks that are reset infinitely often in π . Let π^i denote the suffix of π starting from the position i .

Clearly, there exists an index m such that all the clocks that are lifted in π^m belong to X^l and the ones that are reset in π^m belong to X^r . Since ρ is Zeno, we have $X^l \cap X^r = \emptyset$. This shows that all the clocks that are reset in π^m are never lifted in its transitions. Therefore, there exists an index $k \geq m$ such that for all $j \geq k$, Z_j is consistent with $x < 1$ for all clocks $x \in X^r$ and we get the following path of $\mathcal{SZG}^a(\mathcal{A})$:

$$(q_0, Z_0, \text{free}) \Rightarrow_a^{t_0} \dots (q_j, Z_j, \text{free}) \Rightarrow_a^\tau (q_j, Z_j, \text{slow}) \Rightarrow_a^{t_j} (q_{j+1}, Z_{j+1}, \text{slow}) \Rightarrow_a^{t_{j+1}} \dots \quad \square$$

Lemma 4.6. *If $\mathcal{SZG}^a(\mathcal{A})$ has an infinite slow path, then \mathcal{A} has a Zeno run.*

Proof. Let π be the slow path of $\mathcal{SZG}^a(\mathcal{A})$:

$$(q_0, Z_0, \text{free}) \Rightarrow_a^{t_1} \dots (q_j, Z_j, \text{free}) \Rightarrow_a^\tau (q_j, Z_j, \text{slow}) \Rightarrow_a^{t_j} (q_{j+1}, Z_{j+1}, \text{slow}) \Rightarrow_a^{t_{j+1}} \dots$$

Take the corresponding path in $ZG^a(\mathcal{A})$ and an instance $\rho = (q_0, v_0) \xrightarrow{\delta_0, t_0} (q_1, v_1) \dots$ which is a run of \mathcal{A} , as we have assumed that \mathbf{a} is a sound abstraction.

Let X^r be the set of clocks that are reset infinitely often and let X^l be the set of clocks that are lifted infinitely often in ρ . By the semantics of the slow mode and from our hypothesis of \mathbf{a} being lift-safe, after the index j , all clocks that are lifted once can never be reset again. Therefore, there exists an index $k \geq j$ such that the following hold:

- all clocks that are reset in ρ^k belong to X^r and all clocks that are lifted in a transition of ρ^k belong to X^l ,
- for all $x \in X^l$ and for all $i \geq k$, $v_i(x) \geq c$ where c is the maximum constant appearing in a lifting transition of ρ^k .

We now modify the time delays of ρ^k to construct a run that elapses a bounded amount of time. Pick the sequence of indices i_1, i_2, \dots in ρ^k such that $\delta_{i_m} > 0$, for all $m \in \mathbb{N}$. Define the new delays δ'_i for all $i \geq k$ as follows:

$$\delta'_i = \begin{cases} \min(\delta_i, \frac{1}{2^j}) & \text{if } i = i_j \text{ for some } j \\ 0 & \text{otherwise} \end{cases}$$

Consider the run ρ' obtained by elapsing δ'_i time units after the index k :

$$(q_0, v_0) \xrightarrow{\delta_0, t_0} \dots \xrightarrow{\delta_{k-1}, t_{k-1}} (q_k, v_k) \xrightarrow{\delta'_k, t_k} (q_{k+1}, v'_{k+1}) \xrightarrow{\delta'_{k+1}, t_{k+1}} \dots$$

Clearly, ρ' is Zeno. It remains to prove that ρ' is a run of \mathcal{A} . Denote v_k by v'_k . We need to show that for all $i \geq k$, $v'_i + \delta'_i$ satisfies the guard in the transition t_i . Call this guard g_i . Clearly, since $v'_i + \delta'_i \leq v_i + \delta_i$ by definition, if g_i is of form $x < c$ or $x \leq c$ then it is satisfied by the new valuation. Let us now consider the case when g_i is of the form $x \geq c$ or $x > c$. If $c \geq 1$, then we know that $x \in X^l$ from the assumption on k . But since $v_k(x) \geq c$ and x is not reset anywhere in ρ^k , $v'_i(x) \geq c$ for all i and hence the new valuation satisfies g_i . We are left with the case when g_i is of the form $x > 0$. However this follows since by definition of the new δ'_i , $v'_i + \delta'_i = 0$ iff $v_i + \delta_i = 0$. \square

From the definition of $\mathcal{SZG}^a(\mathcal{A})$ it follows clearly that for each node (q, Z) of the zone graph there are two nodes in $\mathcal{SZG}^a(\mathcal{A})$: (q, Z, free) and (q, Z, slow) . We thus get the following theorem.

Proposition 4.7. *Let \mathbf{a} be a lift-safe abstraction. The automaton \mathcal{A} has a Zeno run iff $\mathcal{SZG}^a(\mathcal{A})$ has an infinite slow path. The number of reachable nodes of $\mathcal{SZG}^a(\mathcal{A})$ is at most twice the number of reachable nodes in $ZG^a(\mathcal{A})$.*

We now turn our attention towards some of the abstractions existing in the literature. We observe that both Extra_M and Extra_M^+ are lift-safe and hence the Zenoness problem can be solved using the slow zone graph construction. However, in accordance to the NP-hardness of the problem for Extra_{LU} , we get that Extra_{LU} is not lift-safe.

Theorem 4.8. *The abstractions Extra_M and Extra_M^+ are lift-safe. The Zenoness problem is solved in polynomial time for Extra_M and Extra_M^+ .*

Proof. Observe that for every clock that is lifted, the bound M is at least 1. It is now direct from the definitions that Extra_M and Extra_M^+ are lift-safe. A polynomial algorithm is easily obtained from Proposition 4.7. \square

Theorem 4.9. *The abstractions Extra_{LU} and Extra_{LU}^+ are not lift-safe. The Zenoness problem for Extra_{LU} and Extra_{LU}^+ is NP-complete.*

Proof. That Extra_{LU} and Extra_{LU}^+ are not lift-safe follows from the proof of Proposition 4.2. We show the NP-membership using a technique similar to the slow zone graph construction. Since Extra_{LU} is not lift-safe, the reachable zones in $ZG^{LU}(\mathcal{A})$ do not maintain the information about the clocks that have been lifted. Therefore, at some reachable zone (q, Z) we non-deterministically guess the set of clocks W that are allowed to be lifted in the future and go to a node (q, Z, W) . From now on, there are transitions $(q, Z, W) \Rightarrow_a^t (q', Z', W)$ when:

- $(q, Z) \Rightarrow_a^t (q', Z')$ is a transition in $ZG^{LU}(\mathcal{A})$,
- if t contains a guard $x \geq c$ with $c \geq 1$, then $x \in W$,
- if t resets a clock x , then $x \notin W$

If a cycle is obtained that contains (q, Z, W) , then the clocks that are reset and lifted in this cycle are disjoint and hence \mathcal{A} has a Zeno run.

This shows that if \mathcal{A} has a Zeno run we can non-deterministically choose a path of the above form and the length of this path is bounded by twice the number of zones in $ZG^{LU}(\mathcal{A})$ (which is our other input). This proves the NP-membership. The NP-hardness is proven in Proposition 4.2. \square

4.3. Weakening the U bounds. We saw in Theorem 4.9 that the extrapolation Extra_{LU} is not lift-safe. This is due to clocks x that are lifted but have $U(x) = -\infty$. These are exactly the clocks x with $L(x) \geq 1$ and $U(x) = -\infty$. We propose to weaken the U bounds so that the information about a clock being lifted is remembered in the abstracted zone.

Definition 4.10 (Weak U bounds). Given the bounds $L(x)$ and $U(x)$ for each clock $x \in X$, the *weak upper bound* $\bar{U}(x)$ is given by: $\bar{U}(x) = 1$ if $L(x) \geq 1$ and $U(x) = -\infty$, and $\bar{U}(x) = U(x)$ otherwise.

Let $\text{Extra}_{L\bar{U}}$ denote the Extra_{LU} abstraction, but with \bar{U} bound for each clock instead of U . This definition ensures that for all lifted clocks, that is, for all $x \in \text{Lf}(\mathcal{A})$, if a zone entails that $x \geq 1$ then $\text{Extra}_{L\bar{U}}(Z)$ also entails that $x \geq 1$. This is summarized by the following proposition, the proof of which follows by definitions and Proposition 4.7.

Theorem 4.11. *For all zones Z , $\text{Extra}_{L\bar{U}}$ is lift-safe. The Zenoness problem is solved in polynomial time for $\text{Extra}_{L\bar{U}}$.*

The Zenoness problem is polynomial for $\text{Extra}_{L\bar{U}}$, however, there is a price to pay. Weakening the U bounds leads to zone graphs exponentially bigger in some cases. For example, for the automaton \mathcal{A}_ϕ^Z that was used to prove the NP-hardness of the Zenoness problem with Extra_{LU} , note that the zone graph $ZG^{L\bar{U}}(\mathcal{A}_\phi^Z)$ obtained by applying $\text{Extra}_{L\bar{U}}$ is exponentially bigger than $ZG^{LU}(\mathcal{A}_\phi^Z)$. This leads to a slow zone graph $\mathcal{SZG}^{L\bar{U}}(\mathcal{A}_\phi^Z)$ with size polynomial in $ZG^{L\bar{U}}(\mathcal{A}_\phi^Z)$. Similar to $\text{Extra}_{L\bar{U}}$ we can define $\text{Extra}_{L\bar{U}}^+$ which is again lift-safe and yields a polynomial algorithm for the Zenoness problem.

5. DISCUSSION ON LU -EXTRAPOLATIONS

In this section, we discuss two observations arising out of the analysis of the non-Zeno/Zeno runs in an automaton. The first observation relates to an optimization in the reachability and liveness algorithms for timed automata. For the second observation, we consider the weak LU -extrapolations $\text{Extra}_{L\bar{U}}$ and $\text{Extra}_{\bar{L}U}$ and look at when these abstractions coincide with the LU -extrapolation Extra_{LU} . We note that this happens for a wide class of timed automata.

5.1. Optimization. Although this paper focuses on the complexity of finding Zeno and non-Zeno behaviours from abstract zone graphs, our analysis showing the NP-hardness of the non-Zenoness problem on LU -abstract zone graphs leads to an interesting side-effect for the classical reachability and liveness problems for timed automata. We have pointed out in the introduction that the reachability and liveness problems are solved via the abstract zone graph. The LU -extrapolations are the standard abstractions used in state-of-the-art implementations [3] as they give rise to small abstract zone graphs. The following observation helps in reducing the abstract zone graph even further in some cases.

Recall the proof of NP-completeness of NZP^{LU} given in Theorem 2.2. For a 3CNF formula ϕ we built an automaton \mathcal{A}_ϕ that has a non-Zeno run iff ϕ is satisfiable. The rest of the proof relies on the crucial fact that the zone graph $ZG^{LU}(\mathcal{A}_\phi)$ is isomorphic to \mathcal{A} . This was indeed possible as $L(x)$ was $-\infty$ for all x thanks to the guards of the form $x \leq 0$. Note that modifying $x \leq 0$ to $x = 0$ does not change the semantics of the automaton, but obliges $L(x)$ to be 0 for all clocks. In this case, the zone graph $ZG^{LU}(\mathcal{A}_\phi)$ is no longer isomorphic to \mathcal{A}_ϕ and in fact it is exponentially larger than \mathcal{A}_ϕ .

This gives us the easy optimization for analyzing an automaton \mathcal{A} for both reachability and liveness. Since both these algorithms go through the zone graph construction, reducing the abstract zone graph, and even trying to get a zone graph isomorphic to the automaton, can produce considerable gain. The optimization consists in changing all the guards in \mathcal{A} that are of the form $x = 0$ to $x \leq 0$ and in removing all the guards $x \geq 0$. Thus, we make sure that $L(x) = -\infty$ unless there is a guard $x \geq c$ or $x > c$ in the automaton. This modification has been incorporated in UPPAAL 4.1.5. Experimental results have shown a

remarkable gain, in particular for timed Petri nets as the translation to timed automata may generate many guards like $x = 0$ and $x \geq 0$. For instance, checking reachability on a model of Fischer’s protocol only explored 2541 nodes instead of 23042 nodes thanks to this optimization.

5.2. How weak are the weak LU -extrapolations. We saw that slightly weakening the LU -bounds makes the NZP^a and ZP^a polynomial (Definitions 3.11, 4.10 and Theorems 3.12, 4.11). Of course, this would indeed increase the size of the zone graph instead. However in many cases the weak abstractions coincide with Extra_{LU} and Extra_{LU}^+ respectively. This shows that in all these cases, one can use the zone graph crafted by Extra_{LU} itself, or alternatively Extra_{LU}^+ , and additionally checking for Zeno behaviours is not costly either. Hence one can expect an efficient procedure for checking Zeno behaviours for these classes of automata.

Assume that we are given an automaton \mathcal{A} . Recall the abstraction Extra_{LU} . This abstraction makes use of weak L bounds for every clock (cf. Definition 3.11). If all clocks that are checked for $x \leq 0$ have a guard of the form $x \geq c$ then the weak abstraction coincides with Extra_{LU} . Notice that this is particularly the case for Timed Automata that do not have zero checks (i.e. guards like $x \leq 0$). Most models of “real systems” do not have such guards.

For the abstraction Extra_{LU} , the abstraction makes use of weak U bounds (cf. Definition 4.10). Notice that if all clocks that are checked for a lower bound guard are also checked for an upper bound then the two abstractions coincide. So, the wide class of systems where each clock is both bounded from above (i.e. $x \leq c$) and from below (i.e. $x \geq c'$) have polynomial-time detection of Zeno runs, even using Extra_{LU} and Extra_{LU}^+ .

6. PSPACE-COMPLETENESS OF (NON-)ZENO RUN DETECTION WITH INPUT \mathcal{A}

In the previous sections, we have characterized the complexity of finding (non-)Zeno runs given an automaton \mathcal{A} and an abstract zone graph $ZG^a(\mathcal{A})$. We now show that in the classical setting, where automaton \mathcal{A} is the only input, the two problems turn out to be harder. We prove the following theorem.

Theorem 6.1. *Given an automaton \mathcal{A} , deciding whether there exists a non-Zeno run is PSPACE-complete. Similarly for deciding if there exists a Zeno run.*

Our proof follows the same lines as the proof of PSPACE-completeness of the emptiness problem for timed automata [1, 8].

PSPACE-MEMBERSHIP

In Theorems 3.10 and 4.9 we have proved that given \mathcal{A} and $ZG^{LU}(\mathcal{A})$, there is a non-deterministic polynomial algorithm for NZP^{LU} and ZP^{LU} . Essentially both the algorithms do the following. They begin by non-deterministically guessing a node (q, Z) of $ZG^{LU}(\mathcal{A})$ and augmenting it with a guessed subset of clocks $S \subseteq X$ to give the node (q, Z, S) . Starting from this node, the algorithms construct a cycle of $ZG^{LU}(\mathcal{A})$ containing (q, Z) and satisfying certain constraints specified by this newly augmented component:

$$(q, Z, S) \Rightarrow^{t_1} (q_1, Z_1, S_1) \Rightarrow^{t_2} \dots \Rightarrow^{t_n} (q_n, Z_n, S_n) \Rightarrow^t (q, Z, S')$$

Since Z can be represented in space $\mathcal{O}(|X|^2)$ using a DBM, nodes of the form (q, Z, S) can be represented in space polynomial in the size of \mathcal{A} . Note that each integer in the (LU-abstracted) zones Z that we consider is less than the maximum constant occurring in \mathcal{A} . To find the above cycle, it is enough to maintain the initially guessed node (q, Z, S) and the current node whose successor has to be computed. Clearly, the non-deterministic algorithm needs space that is polynomial in the size of the input \mathcal{A} . By Savitch's theorem, this shows that deciding if a timed automaton has a non-Zeno run, or dually a Zeno run, is in PSPACE.

PSPACE-HARDNESS

The problem of deciding if a deterministic Linear Bounded Automaton¹ (LBA) \mathcal{B} accepts a word w is known to be PSPACE-complete [15]. We reduce the acceptance problem for deterministic LBAs to the problem of deciding if a timed automaton has a Zeno or a non-Zeno run.

Let \mathcal{B} be a deterministic LBA and let w be a finite word on the input alphabet of \mathcal{B} . Without loss of generality, we can assume that \mathcal{B} has a single accepting state q_F from which there are no outgoing transitions. We also assume that the tape alphabet of \mathcal{B} is $\Gamma = \{1, \dots, k-1\}$. Let n be the length of the input word w (hence the size of the tape of \mathcal{B}).

We build a timed automaton \mathcal{A} that reads the sequence σ of configurations of \mathcal{B} on input w encoded as:

$$\gamma_1^0 \gamma_2^0 \cdots \gamma_n^0 k \ \gamma_1^1 \gamma_2^1 \cdots \gamma_n^1 k \cdots k \ \gamma_1^i \gamma_2^i \cdots \gamma_n^i k \cdots$$

where:

- $\gamma_1^0 \gamma_2^0 \cdots \gamma_n^0$ is the word w , which is the initial content in the tape;
- $\gamma_1^i \gamma_2^i \cdots \gamma_n^i$ is the content in the tape after the first i transitions of \mathcal{B} .

For every word w , there is a unique encoding σ as \mathcal{B} is deterministic. Observe that σ is a sequence of integers in $1, \dots, k$ and k acts as a separator between successive configurations. The automaton \mathcal{A} that we construct below accepts the sequence σ iff \mathcal{B} accepts w .

Call $\gamma_1^i \gamma_2^i \cdots \gamma_n^i$ as the i^{th} block. Each block i can be mapped to a position $p_i \in \{1, \dots, n\}$ which represents the position of the tape head after the i^{th} transition. The position p_0 is the initial position of the tape head which is 1. Similarly, each block i can be mapped to a state q_i of the the LBA \mathcal{B} representing the state of \mathcal{B} after the first i transitions.

We construct the automaton \mathcal{A} as follows.

States. The states of the automaton encode the state of \mathcal{B} and the position of the tape head. So each state of the automaton is of the form (q, p) where q is a state of \mathcal{B} and $p \in \{1, \dots, n\}$ is a position of the tape head. There is an extra auxiliary state $(q_{init}, 0)$ to read the initial block of σ which is the word w itself. The goal is to make the automaton come to (q_i, p_i) after reading the first i blocks:

$$\underbrace{\gamma_1^0 \gamma_2^0 \cdots \gamma_n^0 k}_{(q_{init}, 0)} \ \underbrace{\gamma_1^1 \gamma_2^1 \cdots \gamma_n^1 k}_{(q_0, p_0)} \cdots k \ \underbrace{\gamma_1^i \gamma_2^i \cdots \gamma_n^i k}_{(q_{i-1}, p_{i-1})} \ \underbrace{\gamma_1^{i+1} \gamma_2^{i+1} \cdots \gamma_n^{i+1} k}_{(q_i, p_i)} \cdots$$

¹Linear Bounded Automata are Turing Machines with tape bounded by the length of the input word.

The initial block is read in the initial state $(q_{init}, 0)$ after which the automaton moves to (q_0, p_0) . In general, after reading block i , the automaton should move to (q_i, p_i) which represents the state q_i of \mathcal{B} and the position p_i of the tape head at the time of taking the i^{th} transition. While reading the $i + 1^{th}$ block from state (q_i, p_i) the automaton has to check if the symbol at position p_i of the block corresponds to the modification of the $i + 1^{th}$ transition of \mathcal{B} which is of the form $(q_i, \gamma, \gamma', \Delta, q_{i+1})$.

Clocks. We intend to make the automaton \mathcal{A} spend $k + 1$ time units at each symbol. This is facilitated by a clock x . Spending $k + 1$ time units will also help us to recognize the current symbol which is a number between 1 and k . To this regard, to read a symbol $s \in \sigma$, we use a transition with guard $(x = s)$ followed by a transition with guard $(x = k + 1)$ that resets x . As reading a symbol requires $(k + 1)$ time units, reading a tape configuration (followed by separator symbol k) takes $(n + 1) \cdot (k + 1)$ time units.

To store the currently read symbol, we introduce a clock x_j for each cell j of the tape. If the currently read symbol is γ_j^i , then clock x_j is reset on the transition with guard $x = \gamma_j^i$. Hence, when the symbol γ_j^{i+1} is read, the previous content of the cell j , given by the symbol γ_j^i , is remembered in x_j by the value $(n + 1) \cdot (k + 1) - \gamma_j^i + \gamma_j^{i+1}$. This is illustrated in (6.1).

$$\begin{array}{c} \begin{array}{ccccccc} & \overbrace{\gamma_j^i \text{ t.u.}} & & \overbrace{\gamma_j^{i+1} \text{ t.u.}} & & & \\ \dots & \underbrace{\xrightarrow{(x=\gamma_j^i), \{x_j\}} \xrightarrow{(x=k+1), \{x\}} \dots \xrightarrow{(x=\gamma_j^{i+1}), \{x_j\}} \xrightarrow{(x=k+1), \{x\}} \dots} & & & & & \\ & \underbrace{\hspace{10em}} & & & & & \\ & (n+1) \cdot (k+1) \text{ time units} & & & & & \end{array} \end{array} \quad (6.1)$$

Transitions. Consider a state (q, p) of \mathcal{A} . For each transition $(q, \gamma, \gamma', \Delta, q')$ of \mathcal{B} , there is a sequence of transitions in \mathcal{A} that reads a block and does the following:

- ensures that the p^{th} symbol corresponds to the modification of the p^{th} tape cell forced by this transition,
- ensures that all other symbols are left unchanged corresponding to all other cells being unchanged,
- moves to state $(q', p + \Delta)$ after reading the block.

Moreover, the cells have to be read in the right order, that is, cell 1 should be read followed by cell 2, etc. Recall that x_j is the clock associated with every cell. For every $j \neq p$, we check if $x_j = (n + 1) \cdot (k + 1)$ and for $j = p$ we check if $x_j = (n + 1) \cdot (k + 1) - \gamma + \gamma'$. This will ensure the first two conditions above and will also ensure that the cells are read in the correct succession.

The complete widget for transition $(q, \gamma, \gamma', \Delta, q')$ is depicted in Figure 14. There is one such widget in \mathcal{A} for each state (q, p) such that $p + \Delta$ is a valid position (i.e. $p + \Delta \in \{1, \dots, n\}$).

Initialization. We need to read the word w from state $(q_{init}, 0)$ and assign the initial value of the clocks x_1, \dots, x_n to w_1, \dots, w_n where w_j represents the j^{th} symbol of w . As w is given as an input, we can easily add transitions from $(q_{init}, 0)$ to ensure this and jump to $(q_0, 1)$.

Observe that since \mathcal{B} is deterministic, \mathcal{A} is also deterministic. Furthermore, \mathcal{A} is time-deterministic as all the guards are equalities. Hence, \mathcal{A} has a single run given the word

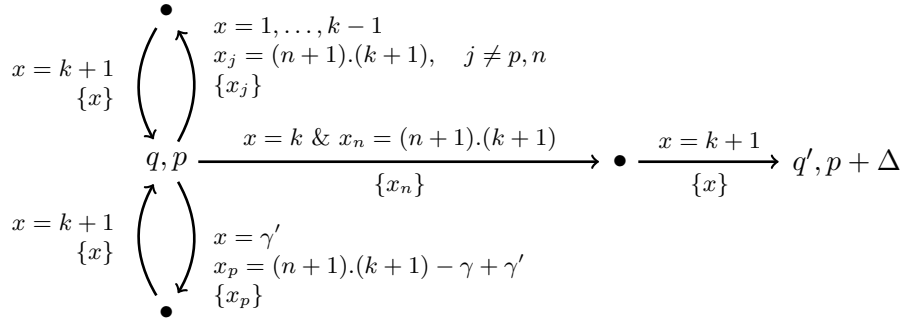


Figure 14: Widget for transition $(q, \gamma, \gamma', \Delta, q')$ on state (q, p) .

w . Furthermore, if \mathcal{B} does not terminate on w , the corresponding run of \mathcal{A} is infinite and non-Zeno.

Recall that q_F is the sole accepting state of \mathcal{B} and there are no transitions outgoing from q_F . From the construction described above, one easily gets the following theorem:

Theorem 6.2. *\mathcal{A} reaches a state (q_F, p) iff \mathcal{B} reaches q_F on input w . The size of \mathcal{A} is polynomial in the size of \mathcal{B} and w .*

Existence of a Non-Zeno Run. We show that an algorithm for deciding if \mathcal{A} has a non-Zeno run yields an algorithm to decide if \mathcal{B} accepts w . This algorithm has two phases.

In the first phase, it determines if \mathcal{A} has a non-Zeno run:

- if the answer is *yes*, we can conclude that \mathcal{B} does not accept w . Indeed, if \mathcal{A} has a non-Zeno run, then it does not reach (q_F, p) for any p as the run is infinite (recall q_F is a sink state by hypothesis), hence neither does \mathcal{B} reach q_F ;
- if the answer is *no*, we cannot conclude. We only gain information that \mathcal{A} has no infinite run, but it may stop in a state (q_F, p) as well as in a non-accepting state.

In the second phase, we transform \mathcal{A} into \mathcal{A}' by adding a loop on all (q_F, p) with guard $(x \geq 1)$ and that resets x . Now, if the run of \mathcal{A}' is infinite, then it visits some (q_F, p) . Furthermore, it is the only non-Zeno run in \mathcal{A}' as we know from the first phase that \mathcal{A} has no infinite run. We now ask if \mathcal{A}' has a non-Zeno run:

- if the answer is *yes*, we can conclude that \mathcal{B} accepts w ;
- if the answer is *no*, the run of \mathcal{A}' is finite and does not reach any (q_F, p) . We can conclude that \mathcal{B} does not accept w .

Existence of a Zeno Run. Now, we show that an algorithm that decides if \mathcal{A} has a Zeno run yields an algorithm to decide if \mathcal{B} accepts w . Recall that \mathcal{A} is deterministic: it has a unique run and if that run is infinite, then it is non-Zeno.

We transform \mathcal{A} into \mathcal{A}' by adding a loop on all states (q_F, p) with guard $(x \leq 0)$. Then we ask if \mathcal{A}' has a Zeno run:

- if the answer is *yes*, then some (q_F, p) has to be reachable, hence \mathcal{B} reaches q_F and accepts w ;
- if the answer is *no*, then no (q_F, p) is reachable, and \mathcal{B} does not accept w .

7. CONCLUSION

We have shown a striking fact that the problem of deciding existence of Zeno or non-Zeno behaviours from abstract zone graphs depends heavily on the abstractions, to the extent that the problem changes from being polynomial to becoming NP-complete as the abstractions get coarser. Of course, it is but natural that checking for Zeno/non-Zeno behaviours becomes difficult when the abstraction gets coarser, as lesser information is maintained. However, the fact this difficulty ranges from a low polynomial to NP-hardness is surprising.

We have proved NP-completeness for the coarse abstractions Extra_{LU} and Extra_{LU}^+ . In contrast, the fundamental problems of finding accepting runs for finitary accepting conditions (reachability), and for Büchi accepting conditions, over abstract zone graphs have a mere linear complexity, independent of the abstraction. As a consequence of the difficulty of detecting non-Zeno runs, the Büchi emptiness problem which consists in finding a run that is both accepting and non-Zeno is NP-complete for abstractions Extra_{LU} and Extra_{LU}^+ .

On the positive side, from our study on the conditions for an abstraction to give a polynomial solution, we see that a small modification of the LU-extrapolation works. We have defined two weaker abstractions: $\text{Extra}_{\bar{L}U}$ for detecting non-Zeno runs and $\text{Extra}_{\bar{L}U}^+$ for detecting Zeno runs. The weak bounds \bar{L} and \bar{U} can also be used with Extra_{LU}^+ to achieve similar results. Despite leading to a polynomial solution for checking Zeno or non-Zeno behaviours from abstract zone graphs, these abstractions transfer the complexity to the input: they could lead to exponentially bigger abstract zone graphs themselves. However, for a fairly large class of automata described in the previous section, we see that this is not the case as the weak abstractions coincide with Extra_{LU} .

While working with abstract zone graphs, coarse abstractions (and hence small abstract zone graphs) are essential to handle big models of timed automata. These, as we have seen, work against the Zenoness questions in the general case. Our results therefore provide a theoretical motivation to look for cheaper substitutes to the notion of Zenoness.

All the abstractions we have considered are convex abstractions. However, there also exist non-convex abstractions [5, 3] that are known to be coarser than the convex ones. Since non-convex sets are particularly difficult to manipulate, only the convex abstractions have been considered for implementation. Recently, new algorithms have been introduced to solve the reachability problem efficiently using non-convex abstractions [11, 13]. Future work includes adaptation of these algorithms to the detection of (non-)Zeno behaviors.

REFERENCES

- [1] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
- [2] Rajeev Alur and P. Madhusudan. Decision problems for timed automata: A survey. In Marco Bernardo and Flavio Corradini, editors, *SFM*, volume 3185 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2004.
- [3] Gerd Behrmann, Patricia Bouyer, Kim Guldstrand Larsen, and Radek Pelánek. Lower and upper bounds in zone-based abstractions of timed automata. *STTT*, 8(3):204–215, 2006.
- [4] Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Hakansson, Paul Pettersson, Wang Yi, and Martijn Hendriks. UPPAAL 4.0. In *QEST*, pages 125–126. IEEE Computer Society, 2006.
- [5] Patricia Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004.

- [6] Howard Bowman and Rodolfo Gómez. How to stop time stopping. *Formal Asp. Comput.*, 18(4):459–493, 2006.
- [7] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. KRONOS: A model-checking tool for real-time systems (tool-presentation for ftrtft '98). In Anders P. Ravn and Hans Rischel, editors, *FTRTFT*, volume 1486 of *Lecture Notes in Computer Science*, pages 298–302. Springer, 1998.
- [8] Costas Courcoubetis and Mihalis Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1(4):385–415, 1992.
- [9] David L. Dill. Timing assumptions and verification of finite-state concurrent systems. In Joseph Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1989.
- [10] Rodolfo Gómez and Howard Bowman. Efficient detection of Zeno runs in timed automata. In Jean-François Raskin and P. S. Thiagarajan, editors, *FORMATS*, volume 4763 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 2007.
- [11] Frédéric Herbreteau, Dileep Kini, B. Srivathsan, and Igor Walukiewicz. Using non-convex approximations for efficient analysis of timed automata. In Supratik Chakraborty and Amit Kumar, editors, *FSTTCS*, volume 13 of *LIPICs*, pages 78–89. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.
- [12] Frédéric Herbreteau and B. Srivathsan. Coarse abstractions make Zeno behaviours difficult to detect. In Joost-Pieter Katoen and Barbara König, editors, *CONCUR*, volume 6901 of *Lecture Notes in Computer Science*, pages 92–107. Springer, 2011.
- [13] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Better abstractions for timed automata. In *LICS*, pages 375–384. IEEE, 2012.
- [14] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Efficient emptiness check for timed Büchi automata. *Formal Methods in System Design*, 40(2):122–146, 2012.
- [15] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
- [16] Guangyuan Li. Checking timed Büchi automata emptiness using LU-abstractions. In Joël Ouaknine and Frits W. Vaandrager, editors, *FORMATS*, volume 5813 of *Lecture Notes in Computer Science*, pages 228–242. Springer, 2009.
- [17] Jonas Rinast and Sibylle Schupp. Static detection of Zeno runs in UPPAAL networks based on synchronization matrices and two data-variable heuristics. In Marcin Jurdzinski and Dejan Nickovic, editors, *FORMATS*, volume 7595 of *Lecture Notes in Computer Science*, pages 220–235. Springer, 2012.
- [18] Stavros Tripakis. Verifying progress in timed systems. In Joost-Pieter Katoen, editor, *ARTS*, volume 1601 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 1999.
- [19] Stavros Tripakis. Checking timed Büchi automata emptiness on simulation graphs. *ACM Trans. Comput. Log.*, 10(3), 2009.
- [20] Stavros Tripakis, Sergio Yovine, and Ahmed Bouajjani. Checking timed Büchi automata emptiness efficiently. *Formal Methods in System Design*, 26(3):267–292, 2005.
- [21] Farn Wang. REDLIB for the formal verification of embedded systems. In *ISoLA*, pages 341–346. IEEE, 2006.