Symbolic Quantitative Robustness Analysis of Timed Automata

Ocan Sankur

Symbolic Quantitative Robustness Analysis of Timed Automata

Ocan Sankur

Université Libre de Bruxelles, Brussels, Belgium

Abstract. We study the robust safety problem for timed automata under guard imprecisions which consists in computing an imprecision parameter under which a safety specification holds. We give a symbolic semi-algorithm for the problem based on a parametric data structure, and evaluate its performance in comparison with a recently published one, and with a binary search on enlargement values.

1 Introduction

Timed automata [2] are a well-established formal model for real-time systems. They can be used to model systems as finite automata, while using, in addition, a finite number of clocks to impose timing constraints on the transitions. Efficient model checking algorithms have been developed and implemented in tools such as Uppaal [6], IF [12]. Timed automata are, however, abstract models, and therefore make idealistic assumptions on timings, such as perfect continuity of clocks, infinite-precision time measures and instantaneous reaction times.

As for any abstract formalism, once desired properties of a system are proven on the model, a crucial question that remains is the *robustness* of these properties against the assumptions that have been made. What is the extent to which the assumptions behind the model can be relaxed while a given property still holds?

In this work, we are interested in the robustness against timing imprecisions. An important amount of work has been done in the timed automata literature to endow timed automata with a realistic semantics, and take imprecisions into account, *e.g.* [18,15,1]. The works [24] and [14] showed that perturbations on clocks, *i.e.* imprecisions or clock drifts, regardless of how small they are, may drastically change the behavior in some models. These observations mean that there is a need for verification tools to check the robustness of timed automata, that is, whether the behavior of a given timed automaton is preserved in the presence of perturbations, and to compute safe bounds on such perturbations.

We consider the robustness of timed automata for safety properties under timing imprecisions modeled by *guard enlargement*, consisting in relaxing each guard of the form $x \in [a, b]$ to $x \in [a - \delta, b + \delta]$ where δ is a parameter. Our goal is to decide if for some $\delta > 0$, the enlarged timed automaton satisfies its specification (Problem 1), and if this is the case, compute a safe upper bound

Supported by the ERC starting grant inVEST (FP7-279499)

on δ (Problem 2). We insist on the importance of both problems: while the first one decides the robustness of the model, the second one *quantifies* it by actually giving a bound under which the model is correct. This would allow one for instance to choose an appropriate hardware to implement the model [15,1].

Background The formulation of Problem 1 has been studied starting with [24,14] for safety properties, and extended to LTL and richer specifications, *e.g.* [9,10] using region-based techniques which cannot be applied efficiently. A symbolic zone-based algorithm was given in [13] for *flat* timed automata, that is, without nested cycles by applying *acceleration* on its cycles. Problem 2 has been answered in [20] for flat timed automata, where the given algorithm computes the *largest* upper bound on δ satisfying the specification. The flatness is a rather restrictive hypothesis since, for instance, it is easily violated when the system is obtained by composition of timed automata that contain cycles. Recently, a zone-based algorithm and a tool to solve Problem 1 for *general* timed automata was given [21]; but the algorithm does not compute any bound on δ. The latter algorithm is based, roughly, on extending the standard forward exploration of the state space augmented with the acceleration of all cycles encountered during the search, with some tricks to optimize the computations. In [22], refinements between interfaces are studied in a game-based framework including syntactic enlargement to account for imprecisions. In [25,26] the authors use the fact that replacing all guards by closed ones allow one to verify *finite* paths (and the case of a periodic external synchronization) but this does not help in the analysis of the accumulation of imprecisions, nor can it allow one to compute a bound on δ.

Results In this paper, we present a symbolic procedure to simultaneously solve Problems 1 and 2 for general timed automata; if the given model is robust, a safe upper bound on δ (which may not be the largest one) is output. The procedure is a semi-algorithm since we do not know whether it terminates although it did terminate on most of our experiments. It consists in a state-space exploration with an efficient parametric data structure which treats the enlargement δ as an unknown parameter, combined with an acceleration procedure for some of the cycles. We do not systematically accelerate cycles, but rather adopt a "lazy" approach: during the exploration, when the accumulated imprecisions go beyond a threshold, we accelerate some cycles that may be responsible for this accumulation. This greatly reduces the computation overhead compared to a systematic acceleration. We also adapt several abstraction operations such as LU abstraction [5], and closure inclusion [19] to the parametric setting to reduce the state space. We ran experiments to evaluate the performance of our procedure. Compared to [21], ours terminated faster in most cases, and sometimes with several orders of magnitude. To truly evaluate the gain of a parametric analysis, we also compared with a binary search on the values of δ using an exact model checker. Our procedure was often faster except against a low precision binary search (*i.e.* with few iterations). Section 6 contains a more detailed discussion.

2 Definitions

Given a finite set of clock $\mathcal{C}$, we call *valuations* the elements of $\mathbb{R}^{\mathcal{C}}_{\geq 0}$. For $R \subseteq \mathcal{C}$ and a valuation v, $v[R \leftarrow 0]$ is the valuation defined by $v[R \leftarrow 0](x) = v(x)$ for $x \in \mathcal{C} \setminus R$ and $v[R \leftarrow 0](x) = 0$ for $x \in R$. Given $d \in \mathbb{R}_{\geq 0}$ and a valuation v, $v + d$ is defined by $(v + d)(x) = v(x) + d$ for all $x \in \mathcal{C}$. We extend these operations to sets of valuations in the obvious way. We write $\mathbf{0}$ for the valuation that assigns 0 to every clock. An *atomic guard* is a formula of the form $k \leq x$ or $x \leq l$ where $x, y \in \mathcal{C}$, $k, l \in \mathbb{Q}$. A *guard* is a conjunction of atomic guards. A valuation v satisfies a guard g, denoted $v \models g$, if all atomic guards are satisfied when each $x \in \mathcal{C}$ is replaced by $v(x)$. We write $\Phi_{\mathcal{C}}$ for the set of guards built on $\mathcal{C}$.

A *timed automaton* $\mathcal{A}$ is a tuple $(\mathcal{L}, \mathsf{Inv}, \ell_0, \mathcal{C}, E)$, where $\mathcal{L}$ is a finite set of locations, $\mathsf{Inv} : \mathcal{L} \to \Phi_{\mathcal{C}}$ the invariants, $\mathcal{C}$ is a finite set of clocks, $E \subseteq \mathcal{L} \times \Phi_{\mathcal{C}} \times 2^{\mathcal{C}} \times \mathcal{L}$ is a set of edges, and $\ell_0 \in \mathcal{L}$ is the initial location. An edge $e = (\ell, g, R, \ell')$ is also written as $\ell \xrightarrow{g,R} \ell'$. For any location ℓ, let $E(\ell)$ denote the set of edges leaving ℓ. Following the literature on robustness in timed automata (e.g. [24,14]) we only consider timed automata with closed and rectangular guards (that is, we do not allow constraints of the form $x - y \leq k$). We also assume that all invariants contain an upper bound for all clocks.

A *run* of $\mathcal{A}$ is a sequence $q_1 e_1 q_2 e_2 \dots q_n$ where $q_i \in \mathcal{L} \times \mathbb{R}^{\mathcal{C}}_{\geq 0}$, and writing $q_i = (\ell, v)$, we have $v \in \mathsf{Inv}(\ell)$, and either $e_i \in \mathbb{R}_{>0}$, in which case $q_{i+1} = (\ell, v + e_i)$, or $e_i = (\ell, g, R, \ell') \in E$, in which case $v \models g$ and $q_{i+1} = (\ell', v[R \leftarrow 0])$. We say that the run r is *along* $e_1 e_2 \dots e_{n-1}$. A *path* is a sequence of edges whose endpoint locations are equal. Given a path $\rho = e_1 e_2 \dots e_{n-1}$ and states q, q', we write $q \xrightarrow{\rho} q'$ if there is a run from q to q' along $e_1 e_2 \dots e_{n-1}$. We write $q \Rightarrow q'$ if there is a path ρ with $q \xrightarrow{\rho} q'$. We also note $q \xrightarrow{\rho^+} q'$ if there is a run from q to q' along an arbitrary (positive) number of repetitions of ρ. A *cycle* of a timed automaton is a path that ends in the location it starts. As in [24,14], we assume that all cycles of considered timed automata reset all clocks at least once. Such cycles are called *progress cycles*.

Regions The first decidability results on timed automata relied on a finite partition of the state space to so called *regions*, which can be defined using simple constraints on clocks [2]. We say that $\frac{1}{\eta}$ is the *granularity* of a timed automaton $\mathcal{A}$, if η is the smallest integer such that all constants of $\mathcal{A}$ are multiples of $\frac{1}{\eta}$. We generalize the definition of regions to arbitrary *granularities*. Let us denote $\mathbb{N}_\eta = \frac{1}{\eta}\mathbb{N}$. Consider a timed automaton $\mathcal{A}$, with granularity $\frac{1}{\eta}$, and consider a bound function $\alpha : \mathcal{C} \to \mathbb{N}_\eta$ mapping each clock to a bound. An α, η*-region* is defined by choosing

- for each clock $x \in \mathcal{C}$, a constraint among $\{x = k \mid k \in \mathbb{N}_\eta, k \leq \alpha(x)\} \cup \{k - \frac{1}{\eta} < x < k \mid k \in \mathbb{N}_\eta, \frac{1}{\eta} \leq k \leq \alpha(x)\} \cup \{x > \alpha(x)\}$.
- for each pair $x, y \in \mathcal{C}$ for which we chose the constraints $k - \frac{1}{\eta} < x < k$, and $l - \frac{1}{\eta} < y < l$, choose one constraint among $\mathsf{frac}(x) < \mathsf{frac}(y)$, $\mathsf{frac}(x) = \mathsf{frac}(y)$, or $\mathsf{frac}(x) > \mathsf{frac}(y)$, where $\mathsf{frac}(\cdot)$ denotes the fractional part.

It can be shown that α, η-regions finitely partition the state space $\mathbb{R}^{\mathcal{C}}$. For $\eta = 1$, this is the usual definition of regions. Given timed automata with rational constants, one often rescales the constants to work with integers. In the context of enlargement, however, it will be more convenient to work directly with rationals.

Difference-Bound Matrices Because the number of regions is exponential in the input size, the region-based algorithms are not practical. Symbolic algorithms were rather given as an efficient solution based on *zones* which are convex subsets of the state space definable by clock constraints. Formally, a zone Z is a convex subset of $\mathbb{R}^{\mathcal{C}}$ definable by a conjunction of constraints of the form $x \leq k$, $l \leq x$, and $x - y \leq m$ where $x, y \in \mathcal{C}$, $k, l \in \mathbb{Q}_{\geq 0}$ and $m \in \mathbb{Q}$. Note that because all guards of the timed automata we consider are closed, all zones that appear during a state-space exploration are closed. Hence, we do not need to distinguish strict and non-strict inequalities as done for general timed automata.

We recall a few basic operations defined on zones. Let $\mathsf{Post}_{\geq 0}(Z)$ denote the zone describing the time-successors of Z, i.e., $\mathsf{Post}_{\geq 0}(Z) = \{v \in \mathbb{R}^C_{\geq 0} \mid \exists t \geq 0, v - t \in Z\}$; and similarly $\mathsf{Pre}_{\geq 0}(Z) = \{v \in \mathbb{R}^C_{\geq 0} \mid \exists t \geq 0, v + t \in Z\}$. Given $R \subseteq \mathcal{C}$, we let $\mathsf{Reset}_R(Z)$ be the zone $\{v \in \mathbb{R}^C_{\geq 0} \mid \exists v' \in Z, v = v'[R \leftarrow 0]\}$, and $\mathsf{Free}_R(Z) = \{v \in \mathbb{R}^C_{\geq 0} \mid \exists v' \in Z, v' = v[R \leftarrow 0]\}$. Intersection is denoted $Z \cap Z'$. Zones can be represented by *difference-bound matrices (DBM)* which are $|\mathcal{C}_0| \times |\mathcal{C}_0|$-matrices with values in $\mathbb{Q}$ [16]. Let us define $\mathcal{C}_0 = \mathcal{C} \cup \{0\}$, where 0 is seen as a clock whose value is always 0. Intuitively, each component $(x, y) \in \mathcal{C}_0^2$ of a DBM stores a bound on the difference $x - y$. For any DBM M, let $[\![M]\!]$ denote the zone it defines. DBMs admit *reduced forms* (a.k.a. normal form), and successor computation can be done efficiently (in $O(|\mathcal{C}|^3)$). We refer the reader to [7] for details. All of the above operations can be computed with DBMs. By a slight abuse of notation, we will use the same operations for DBMs as for zones, for instance, we will write $M' = \mathsf{Post}_{\geq 0}(M)$ where M and M' are reduced DBMs such that $[\![M']\!] = \mathsf{Post}_{\geq 0}[\![M]\!]$. We define an *extended zone* as a pair (ℓ, Z) where ℓ is a location and Z a zone. Given an edge $e = (\ell, g, R, \ell')$, and an extended zone (ℓ, Z), we define $\mathsf{Post}_e\big((\ell, Z)\big) = \mathsf{Inv}(\ell') \cap \mathsf{Post}_{\geq 0}(g \cap \mathsf{Reset}_R(Z))$, and $\mathsf{Pre}_e\big((\ell, Z)\big) = \mathsf{Pre}_{\geq 0}(g \cap \mathsf{Free}_R(\mathsf{Inv}(\ell') \cap Z))$. For a path $\rho = e_1 e_2 \dots e_n$, we define Post_ρ and Pre_ρ by iteratively applying Post_{e_i} and Pre_{e_i} respectively.

Enlargement We model timing imprecisions in timed automata by the *enlargements* of the guards and invariants of by rational values $\nu > 0$. The *enlargement* of an atomic guard $k \leq x$ (resp. $x \leq l$) is denoted $(k \leq x)_\nu = k - \nu \leq x$ (resp. $(x \leq l)_\nu = x \leq l + \nu$). The enlargement of a guard g, denoted $(g)_\nu$ is obtained by enlarging all its conjuncts. We denote by $\mathcal{A}_\nu$ the timed automaton obtained from $\mathcal{A}$ by enlarging all its guards and invariants by ν.

If ν is known, one could analyze $\mathcal{A}_\nu$ with known techniques, since this is still a timed automaton (with a possibly different granularity). Here, we are rather interesting in a parametric analysis. We thus consider a symbolic parameter δ. The *parametric enlargement* of a guard g, denoted $(g)_\delta$ is defined by replacing ν by the symbol δ in the above definition. We will always denote rational enlargement

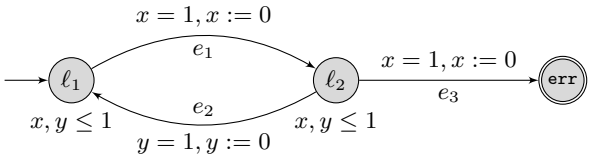

Fig. 1. A timed automaton representing the two processes P_1 and P_2 instantiated with period $p = 1$, and a buffer size of 1. The guard under the locations are the invariants. The edge e_1 represents the arrival of a token in the buffer (period of 1) while e_2 represents process P_2 reading a token from the buffer. The error state is reached via e_3 if two tokens are pushed to the buffer without any read in between.
Without enlargement, any reachable state at location ℓ_2 satisfies $x = 0 \wedge y = 1$, so the error state is not reachable. Under enlargement by $\nu = \frac{1}{10}$, after the first transition, location ℓ_2 is reached by the set of states $1 - \nu \leq y \leq 1 + \nu \wedge 0 \leq x \leq 2\nu$ due to the enlargement of the guards and invariants. A simple calculation shows that the set of reachable states at location ℓ_2 after k cycles is $1 - (2k + 1)\nu \leq y \leq 1 + \nu \wedge 0 \leq x \leq 2k\nu \wedge 1 - (2k + 1)\nu \leq y - x \leq 1 + \nu$. Thus, for $k = 5$, we get $y \leq 1 + \nu \wedge x \leq 1 \wedge -\nu \leq y - x \leq 1 + \nu$, and $x = 1 \wedge y = 1$ is in this set, from which the error state is reachable.

values by ν, and the parameter symbol by δ. Similarly, *parametrically enlarged timed automata* are denoted $\mathcal{A}_\delta$. For $\nu > 0$, the *instantiation* of a parametrically enlarged guard $(g)_\delta$ by ν is denoted $(g)_\delta[\delta \leftarrow \nu]$ which is $(g)_\nu$.

Accumulation of Imprecisions In some timed automata even the smallest enlargement can lead to drastically different behaviors due to the accumulation of the imprecisions over long runs [24]. As an example, consider the following simple problem. Two processes P_1, P_2 execute on different machines and communicate via a finite buffer. Every p time units, Process P_1 finishes a computation and pushes a token to the buffer; while P_2 reads a token from the buffer with the same period. We assume P_2 has an offset of p. The buffer will clearly not overflow in this system. However, assuming the slightest delay in the execution of P_2, or the slightest decrease in the execution time of P_1 leads to a buffer overflow since the delays will accumulate indefinitely. Figure 1 represents this system.

3 Accelerating Cycles

The original robust reachability algorithm of [24,14] consists in an exploration of the region graph, augmented with the addition of the images of all cycles *neighboring* reachable states. The idea is that when the guards are enlarged, these neighboring cycles become reachable, and they precisely capture all states that become reachable in the timed automaton for all values of δ. Thus, this algorithm computes the set $\cap_{\nu>0}\mathsf{Reach}(\mathcal{A}_\nu)$, where $\mathsf{Reach}(\mathcal{A}_\nu)$ denotes the states that are reachable in $\mathcal{A}_\nu$. A symbolic algorithm for this problem was given in [13] for *flat* timed automata, *i.e.* without nested cycles, and later improved in [20].

In this section, we summarize some of these results from [24,14,13,20] that we use in the rest of the paper. Let us fix a timed automaton $(\mathcal{L}, \mathsf{Inv}, \ell_0, \mathcal{C}, E)$, and a cycle ρ. The following lemma shows the effect of repeating cycles in enlarged timed automata, formalizing our observations on Fig. 1.

Lemma 1 ([20]). *Consider any extended zone (ℓ, Z), and a progress cycle ρ of $\mathcal{A}$ that starts in ℓ. If $\mathsf{Pre}^*_\rho(\top) \cap Z \neq \emptyset$, then starting from any state of $\mathsf{Pre}^*_\rho(\top) \cap Z$, for any $\nu > 0$, all states of $\mathsf{Post}^*_{(\rho)_\nu}(\top)$ are reachable in $\mathcal{A}_\nu$, by repeating ρ.*

As an example, consider Fig. 1. For the cycle $\rho = e_2 e_1$ that starts at ℓ_2, we have $\mathsf{Pre}^*_\rho(\top) = x, y \leq 1 \wedge x - y \leq 0$, and $\mathsf{Post}^*_{(\rho)_\nu}(\top) = x, y \leq 1 + \nu \wedge x - y \leq 0$. Since the point $(0, 1)$ is reachable and belongs to Pre^*_ρ, all states of $\mathsf{Post}^*_{(\rho)_\nu}(\top)$ are reachable, and in particular $(1, 1)$ from which the error state is reachable.

It is known that the above lemma does not hold for non-progress cycles; nevertheless, it was shown that in this case, $\mathsf{Post}^*_{(\rho)_\nu}(\top)$ is an *over-approximation* of the states reachable by repeating ρ under enlargement [11]. Thus, the algorithm of [24,14] may have false negatives (may answer "not robust" even though it is) but not false positives on timed automata with arbitrary cycles.

Conversely, it has been shown that any state that belongs to $\cap_{\nu>0}\mathsf{Reach}(\mathcal{A}_\nu)$ is reachable along an alternation of exact runs and repetitions of enlarged cycles.

Lemma 2 ([14]). *Assume that $q \to q'$ in $\mathcal{A}_\nu$ for all $\nu > 0$. There exists a path $\pi_0\rho_0\pi_1\rho_0 \dots \pi_n$ of $\mathcal{A}$ and states $q = q_0, q'_0, q_1, q'_1, \dots, q_n = q'$, such that for all $0 \leq i \leq n-1$, and any $\nu > 0$, $q_i \xrightarrow{\pi_i} q'_i$, $q'_i \in \mathsf{Pre}^*_{\rho_i}(\top)$, and $q_{i+1} \in \mathsf{Post}^*_{(\rho_i)_\nu}(\top)$.*

Notice that the sequence of states $q_0, q'_0, \dots$ is independent of $\nu > 0$ in the above lemma, and that the enlargement is only used in $\mathsf{Post}^*_{((\rho)_i)_\nu}(\top)$.

The algorithms of [13,20] consist in a usual forward exploration on zones augmented with the application of Lemma 1, by enumerating *all* cycles in a *flat* timed automaton. This cannot be extended to general timed automata since the cycles that appear in Lemma 2 are not necessarily simple. This has been a major obstacle against general robust safety algorithms for timed automata.

4 Infinitesimally Enlarged DBMs

We define *infinitesimally enlarged DBMs (IEDBM)*, a parameterized extension of DBMs, which we will use to explore the state space of enlarged timed automata. These were first defined in [14] to be used solely as a proof technique. Here, we extend this data structure with additional properties and explicit computations of the bounds on parameter δ, and show how it can be used to efficiently explore the state space.

We fix a clock set $\mathcal{C}_0$ including the 0 clock. An *infinitesimally enlarged DBM (IEDBM)* is a pair $(M, P)_{\langle 0,\delta_0\rangle}$ where M is a DBM and P is a $|\mathcal{C}_0| \times |\mathcal{C}_0|$ matrix over $\mathbb{N}$, called the *enlargement matrix*. The value $\delta_0 \in (0, \infty)$ is an upper bound on the unknown parameter δ. Intuitively, an IEDBM $(M, P)_{\langle 0,\delta_0\rangle}$ represents the *set* of DBMs $M + \nu P$ where $\nu \in [0, \delta_0)$. Figure 2 shows an example. We often see,

abusively, an IEDBM as a matrix over pairs $(m,p) \in \mathbb{Z} \times \mathbb{N}$. The component (x,y) is denoted by $(M,P)_{\langle 0,\delta_0\rangle}[x,y]$. For simplicity, we always consider the half-open intervals of the form $[0,\delta_0)$ even though ν can be chosen equal to δ_0 in some cases. This is not a loss of generality since we are interested in the small values of ν.

IEDBMs will allow us to reason on the parametric state space of enlarged timed automata "for small values of δ", which means that our computations on the data structure will hold for all $\nu \in [0,\delta_0)$, where $\delta_0 > 0$ is bound that is possibly updated to a smaller value after each operation. For instance, given sets $Z_1 = 1 \le x \le 2+3\nu$ and $Z_2 = x \le 3$, for unknown δ, and assume we want to compute their intersection. We will write $Z_1 \cap Z_2 = 1 \le x \le 2+3\delta$ and chose $\delta_0 \le \frac{1}{3}$. To make these simplifications, we need to compare pairs of IEDBM components in a similar spirit. For instance, to make the above simplification, we write $(2,3)_{\langle 0,\frac{1}{3}\rangle} \le (3,0)_{\langle 0,\frac{1}{3}\rangle}$, which means that $2+3\nu \le 3$ for all $\nu \in [0,\frac{1}{3})$. We formalize this in the next subsection. To ease reading, we may omit δ_0 from IEDBMs if it is clear from the context.

$$\begin{array}{c|ccc} & 0 & x & y \\ 0 & 0 & 1 & 1 \\ x & 3 & 0 & 2 \\ y & 2 & 1 & 0 \end{array} + \delta \begin{pmatrix} 0 & 0 & 1 \\ 2 & 0 & 3 \\ 3 & 3 & 0 \end{pmatrix}$$

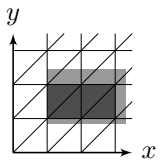

Fig. 2. An IEDBM (above) representing the parametric set $1 \le x \le 3+2\delta \wedge 1-\delta \le y \le 2+3\delta$. The set is represented (below) for $\delta = 0.15$.

4.1 Operations on IEDBMs

We are now going to formalize the simplifications and the operations done on IEDBMs. In order to use our data structure for parametric exploration, we need to define basic operations on timed automata. Similar to DBMs, IEDBMs have *reduced forms*. To define the reduction operations, we define the + and min operations.

We define the sum as $(m,p)_{\langle 0,\delta_0\rangle} + (n,q)_{\langle 0,\delta_1\rangle} = (m+n, p+q)_{\langle 0,\min(\delta_0,\delta_1)\rangle}$. As an example of this operation, assume that we have the constraints $x - y \le 2+3\delta$, and $y \le 1+\delta$ with the respective upper bounds δ_0 and δ_1. Then, by summation, we may deduce $x \le 3+4\delta$ with the upper bound $\delta_2 = \min(\delta_0,\delta_1)$.

Lemma 3. *Given $(m_1,p_1)_{\langle 0,\delta_1\rangle}, (m_2,p_2)_{\langle 0,\delta_2\rangle}$, one can compute $i \in \{1,2\}$, and δ_3 such that $\forall \nu \in [0,\delta_3)$, $m_i + \nu p_i = \min(m_1+\nu p_1, m_2+\nu p_2)$. We denote this by $(m_i,p_i)_{\langle 0,\delta_3\rangle} = \min((m_1,p_1)_{\langle 0,\delta_1\rangle}, (m_2,p_2)_{\langle 0,\delta_2\rangle})$.*

We write $(m_1,p_1)_{\langle 0,\delta_1\rangle} \le (m_2,p_2)_{\langle 0,\delta_2\rangle}$ iff $\min\big((m_1,p_1)_{\langle 0,\delta_1\rangle}, (m_2,p_2)_{\langle 0,\delta_2\rangle}\big) = (m_1,p_1)_{\langle 0,\delta_3\rangle}$ for some δ_3; and $(m_1,p_1)_{\langle 0,\delta_1\rangle} < (m_2,p_2)_{\langle 0,\delta_2\rangle}$ if $m_1 \neq m_2 \vee p_1 \neq p_2$.

Intuitively, just like in regular DBMs, we will use the minimum operation to compute *conjunctions* of constraints. For instance, we already saw above that $x \le 2+3\nu \wedge x \le 3$ is simplified to $x \le 2+3\nu$ given $\nu \in [0,\frac{1}{3})$. This will be written $\min\big((2,3)_{\langle 0,\infty\rangle}, (3,0)_{\langle 0,\infty\rangle}\big) = (2,3)_{\frac{1}{3}}$. It should be clear that for any Boolean

combination Φ of inequalities written between elements of the form $(m,p)_{\langle 0,\delta_1\rangle}$, one can compute, by Lemma 3, a bound δ_0 such that either $\Phi[\delta \leftarrow \nu]$ holds for all $\nu \in [0,\delta_0)$, or $\neg\Phi[\delta \leftarrow \nu]$ holds for all $\nu \in [0,\delta_0)$.

If different upper bounds on δ are given for two elements to be compared, then we first update the bounds to the minimum of the two bounds. More generally, we assume that all components of an IEDBM have the same bound δ_0.

We say that an IEDBM $(M,P)_{\langle 0,\delta_0\rangle}$ is *reduced* if for all $x,y,z \in \mathcal{C}_0$, $(M,P)_{\langle 0,\delta_0\rangle}[x,y] \leq (M,P)_{\langle 0,\delta_0\rangle}[x,z] + (M,P)_{\langle 0,\delta_0\rangle}[z,y]$. IEDBMs can be reduced by the usual Floyd-Warshall algorithm, using the above min and + operations:

Lemma 4. *Any IEDBM $(M,P)_{\langle 0,\delta_0\rangle}$ can be reduced in time $O(|\mathcal{C}_0|^3)$. Moreover, if $(M,P)_{\langle 0,\delta_0\rangle}$ is reduced, then for all $\nu \in [0,\delta_0)$, $M+\nu P$ is a reduced DBM.*

When we consider the complexity of minimization as in Lemma 3, we assume that operations on rationals are elementary operations (*i.e.* they can be performed in constant time). For a more precise analysis, one can incorporate the cost of these computations; for instance, the reduction operation in the previous lemma makes $O(|\mathcal{C}_0|^3)$ minimization operations, so as many operations on rationals.

We define the *parametric inclusion* by $(M,P)_{\langle 0,\delta_1\rangle} \sqsubseteq (N,Q)_{\langle 0,\delta_2\rangle}$ if, and only if for all $x,y \in \mathcal{C}$, $(M,P)[x,y] \leq (N,Q)[x,y]$.

Lemma 5. *One can compute, given $(N_1,Q_1)_{\langle 0,\delta_1\rangle}, (N_2,Q_2)_{\langle 0,\delta_2\rangle}$, and $R \subseteq \mathcal{C}$, and in time $O(|\mathcal{C}|^3)$,*

- *a reduced IEDBM $(M,P)_{\langle 0,\delta_0\rangle}$, written $(N_1,Q_1)_{\langle 0,\delta_1\rangle} \sqcap (N_2,Q_2)_{\langle 0,\delta_2\rangle}$, such that $M+\nu P = (N_1+\nu Q_1) \cap (N_2+\nu Q_2)$ for all $\nu \in [0,\delta_0)$,*
- *a reduced IEDBM $(M,P)_{\langle 0,\delta_0\rangle}$, written $\mathtt{PReset}_R\big((N_1,Q_1)_{\langle 0,\delta_1\rangle}\big)$, such that $M+\nu P = \mathsf{Reset}_R\big(N_1+\nu Q_1\big)$ for all $\nu \in [0,\delta_0)$,*
- *a reduced IEDBM $(M,P)_{\langle 0,\delta_0\rangle}$, written $\mathtt{PPost}_{\geq 0}\big((N_1,Q_1)_{\langle 0,\delta_2\rangle}\big)$, such that $M+\nu P = \mathsf{Post}_{\geq 0}\big(N_1+\nu Q_1\big)$ for all $\nu \in [0,\delta_0)$.*

We are going to define the *parametric post* operation along an edge e. By a slight abuse of notation, we will see any (non-enlarged) guard g as the IEDBM $(g,\mathbf{0})_{\langle 0,\infty\rangle}$. When we consider the enlargement $(g)_\delta$ of a guard, this will also refer to the corresponding IEDBM with $\delta_0 = \infty$. By combining these operations, for a given edge e, we define $\mathtt{PPost}_e\big((M,P)_{\langle 0,\delta_0\rangle}\big) = \mathtt{PPost}_{\geq 0}\big(\mathtt{PReset}_R\big(g \sqcap (M,P)_{\langle 0,\delta_0\rangle}\big)\big)$, where g is the guard of e, and R its reset set. By Lemma 5 this corresponds to $\mathsf{Post}_e\big(M+\delta P\big)$ for sufficiently small δ.

We refer to pairs of locations and IEDBMs as *symbolic states*. We extend the parametric post operator to symbolic states by $\mathtt{PPost}_e\big((\ell,Z)\big) = (\ell',Z')$ where $e=(\ell,g,R,\ell')$, and $Z' = \mathtt{PPost}_e\big(Z\big)$.

Lemma 6. *For any sequence of edges $e_1 \dots e_n$, and symbolic state (ℓ,Z), if $\mathtt{PPost}_{(e_1)_\delta (e_2)_\delta \dots (e_{n-1})_\delta}((\ell,Z)) = (\ell',Z')$, and $(\ell',Z') \neq \emptyset$, then there exists $\delta_0 > 0$ such that for all $\nu \in [0,\delta_0)$, and state $q' \in (\ell',Z')[\delta \leftarrow \nu]$, there exist $q \in (\ell,Z)[\delta \leftarrow \nu]$ such that $(\ell_1,q) \xrightarrow{(e_1)_\nu \dots (e_{n-1})_\nu} (\ell_n,q')$.*

Let the *width* of $(M,P)_{\langle 0,\delta_0\rangle}$ be defined as $\mathsf{width}(M,P) = \max_{x,y\in\mathcal{C}_0} P_{x,y}$.

4.2 Parametric Abstractions

We will first recall some abstractions applied on zones in a non-parametric setting, then generalize them to symbolic states described by IEDBMs.

Closures and LU-abstraction Let $\alpha : \mathcal{C} \to \mathbb{N}$ be a bound function, and η a granularity. The α, η-*closure* of a zone Z is the union of the α, η-regions which intersects it. It is known that when α denotes the maximal constants to which each clock is compared in a timed automaton $\mathcal{A}$, and η the granularity of $\mathcal{A}$, a forward exploration based on α, η-closures is sound and complete [8]. However because closures are not convex, other abstractions have been in use in practical tools; one of them is *LU-abstraction*, where the idea is to relax some of the facets of a zone taking into consideration the maximal constants that appear in the guards of the timed automaton. We will recall the formal definition of LU-abstraction by adapting it to DBMs with only non-strict inequalities by a slight change. The correctness of the abstraction is preserved (proved in Lemma 7).

For a timed automaton $\mathcal{A}$ with granularity η, we define the two bound functions $L, U : \mathcal{C} \to \mathbb{N}_\eta$, called the *LU-bounds* where $L(x)$ (resp. $U(x)$) is the largest constant c such that the constraint $x \geq c$ (resp. $x \leq c$) appears in some guard or invariant. Given LU-bounds L, U, for any DBM M, we define $M' = \mathsf{Extra}^+_{LU}(M)$ as follows.

$$M'_{x,y} = \begin{cases} \infty & \text{if } M_{x,y} > L(x), \text{ or } -M_{0,x} > L(x) \\ \infty & \text{if } -M_{0,y} > U(y), x \neq 0 \\ -U(y) - 1 & \text{if } -M_{0,y} > U(y), x = 0 \\ M_{x,y} & \text{otherwise.} \end{cases} \tag{1}$$

We recall the correctness of LU-abstractions and closures for reachability properties. Given LU-bounds L, U we write $\alpha = \max(L, U)$ for the function $\alpha(x) = \max(L(x), U(x))$ for all $x \in \mathcal{C}$.

Lemma 7. *For any timed automaton $\mathcal{A}$ with granularity η, its LU-bounds L, U, and any path $e_1 e_2 \dots e_n$ and extended zone (ℓ, Z), define $q_0 = q'_0 = q''_0 = (\ell, Z)$, and let for $0 \leq i < n$, $q_{i+1} = \mathsf{Post}_{e_i}(q_i)$, $q'_{i+1} = \mathsf{Extra}^+_{LU}(\mathsf{Post}_{e_i}(q'_i))$, and $q''_{i+1} = \mathsf{Closure}_{\alpha,\eta}(\mathsf{Extra}^+_{LU}(q''_i))$. Then, $q_n \neq \emptyset \Leftrightarrow q'_n \neq \emptyset \Leftrightarrow q''_n \neq \emptyset$.*

One can thus explore the state space of a timed automaton while systematically applying LU-abstraction at each step. In practice, one does not apply closures since they do not yield convex sets. Nevertheless, a $O(|\mathcal{C}|^2)$-time algorithm was given in [19] to decide whether $M \subseteq \mathsf{Closure}_\alpha(N)$. Thus, when the regular inclusion test is replaced with the latter one, the exploration becomes equivalent to an exploration using closures [19,8].

Parametric Closures and LU-abstraction We would like to use these abstractions in our parametric setting. We will show how these sets can be computed parametrically using IEDBMs. Observe that when we consider parametrically enlarged timed automata, the LU-bounds also depend on δ. Let us denote the *parametric*

LU-bounds by $L_\delta(x)$ (resp. $U_\delta(x)$) which is the maximum *parametric* constant, in the sense of Lemma 3, which appears in the guards of $\mathcal{A}_\delta$ as a lower bound (resp. upper bound) on x. We define the *parametric LU-abstraction*, for any IEDBM $(M,P)_{\langle 0,\delta_0\rangle}$ by $(M',P')_{\langle 0,\delta_1\rangle} = \mathtt{PExtra}^+_{L_\delta U_\delta}((M,P)_{\langle 0,\delta_0\rangle})$ obtained by applying (1) where M is replaced by (M,P), L and U by L_δ and U_δ respectively. The new upper bound δ_1 is computed so that all comparisons in (1) hold.

Lemma 8. *Consider any enlarged timed automaton $\mathcal{A}_\delta$ and its parametric LU-bounds L_δ, U_δ. For any $(M,P)_{\langle 0,\delta_0\rangle}$, if we write $(M',P')_{\langle 0,\delta_1\rangle} = \mathtt{PExtra}^+_{L_\delta U_\delta}((M,P)_{\langle 0,\delta_0\rangle})$, then for all $\nu \in [0,\delta_1)$, $M'+\nu P' = \mathsf{Extra}^+_{LU}(M+\nu P)$.*

Thus, LU-abstractions of enlarged zones have uniform representations for small δ.

For an enlarged timed automaton $\mathcal{A}_\delta$ we define $\alpha_\delta = \max(L_\delta, U_\delta)$. For $\nu > 0$, we will denote by α_ν the function obtained from α_δ by instantiating δ to ν. By a slight abuse of notation, we define the *α_ν-closure* of a zone as its (α_ν,η)-closure where η is the granularity of $\mathcal{A}_\nu$. Here $\mathcal{A}$ will be clear from the context, so there will be no ambiguity. We now adapt the inclusion algorithm of [19] to IEDBMs.

Lemma 9. *Given IEDBMs $Z = (M,P)_{\langle 0,\delta_0\rangle}$ and $Z' = (N,Q)_{\langle 0,\delta_0\rangle}$, we have $\forall \delta_0 > 0, \exists \nu \in [0,\delta_0), M+\nu P \not\subseteq \mathsf{Closure}_{\alpha_\nu}(N+\nu Q)$, iff, writing $Z' = N+\nu Q$, there exist $x,y \in \mathcal{C}$ such that*

1. $\big(Z'_{x,0} < Z_{x,0}$ *and* $Z'_{x,0} \le \alpha_\delta(x)\big)$, *or* $\big(Z'_{0,x} < Z_{0,x}$ *and* $Z_{0,x} + \alpha_\delta(x) \ge 0\big)$,
2. *or* $Z_{0,x} + \alpha_\delta(x) \ge 0$, *and* $Z'_{y,x} < Z_{y,x}$, *and* $Z'_{y,x} \le \alpha_\delta(y) + Z_{0,x}$.

Moreover, if this condition doesn't hold, then one can compute δ_1 under which we do have the inclusion $M+\nu P \subseteq \mathsf{Closure}_{\alpha_\nu}(N+\nu Q)$.

Notation 10 *We denote the successor operation followed by LU-abstraction as $\mathsf{ExPost}(\cdot) = \mathsf{Extra}^+_{LU}(\mathsf{Post}(\cdot))$. For the parametric version, we denote $\mathtt{PExPost}(\cdot) = \mathtt{PExtra}^+_{L_\delta U_\delta}(\mathtt{PPost}(\cdot))$, where the bounds L_δ, U_δ are implicit. We furthermore denote by $\sqsubseteq^c$ the* parametric inclusion *check defined by Lemma 9.*

We implicitly assume that when a parametric inclusion check $\sqsubseteq^c$ is performed, the upper bound δ_0 is globally updated to the new bound δ_1 given by Lemma 9.

4.3 Parametric Cycle Acceleration

In [20] a parametric data structure based on the *parameterized DBMs* of [3] was used to represent the state space under *all* values of δ rather than for small values. The corresponding algorithms are based on linear arithmetics of reals. This results in a more complicated data structure which is also more general. IEDBMs simplify this representation by storing the state space only for small values of δ, that is $\delta \in [0,\delta_0)$. To compute cycle accelerations, we recall a result of [20] which bounds the number of iterations to compute pre and post fixpoints of a given cycle.

Lemma 11 ([20]). *Let $N = |\mathcal{C}|^2$. For any cycle ρ, if $\mathtt{PPost}^*_{(\rho)_\delta}(\top) \neq \emptyset$ then $\mathtt{PPost}^*_{(\rho)_\delta}(\top) = \mathtt{PPost}^N_{(\rho)_\delta}(\top)$, and if $\mathtt{PPre}^*_\rho(\top) \neq \emptyset$ then $\mathtt{PPre}^*_\rho(\top) = \mathsf{Pre}^N_\rho(\top)$.*

Data: Timed automaton $\mathcal{A} = (\mathcal{L}, \mathsf{Inv}, \ell_0, \mathcal{C}, E)$, and target location ℓ_T.

1 Wait := $\{(\ell_0, Z_0)_{\langle\infty\rangle}\}$, Passed := $\emptyset$, $(\ell_0, Z_0).K := K_0$;
2 **while** *Wait* $\neq \emptyset$ **do**
3 (ℓ, Z) := pop(Wait), Add (ℓ, Z) to Passed;
4 **if** $\ell = \ell_T$ **then return** Unsafe;
5 **if** $\mathsf{width}(Z) > (\ell, Z).K$ **then**
6 Let π denote the prefix that ends in (ℓ, Z), along edges $e_1 e_2 \dots e_{|\pi|-1}$;
7 **foreach** *cycle* $\rho = e_i e_{i+1} \dots e_j$ **do**
8 **if** $\mathtt{PPre}^*_\rho(\top) \cap \pi_i \neq \emptyset$ *and* $\forall q \in$ Passed, $\mathtt{PPost}^*_{(\rho)_\delta}(\top) \not\sqsubseteq^c q$ **then**
9 Add $\mathtt{PPost}^*_{(\rho)_\delta}(\top)$ as a successor to π_j, and to Wait;
10 **end**
11 **end**
12 **if** *no fixpoint was added* **then** $(\ell, Z).K = (\ell, Z).K + K_0$;
13 **foreach** $e \in E(\ell)$ *s.t.* $\forall q \in$ Passed, $\mathtt{PExPost}_{e_\delta}((\ell, Z)) \not\sqsubseteq^c q$ **do**
14 $(\ell', Z') := \mathtt{PExPost}_{e_\delta}((\ell, Z))$;
15 Add (ℓ', Z') to Wait;
16 $(\ell', Z').\text{parent} := (\ell, Z)$;
17 $(\ell', Z').K := (\ell, Z).K$;
18 **end**
19 **end**
20 **return** Safe;

Algorithm 1: Symbolic robust safety semi-algorithm. Here (ℓ_0, Z_0) is the initial state symbolic state, and K_0 is a positive constant. We have two containers *Wait* and *Passed* storing symbolic states. The search tree is formed by assigning to each visited state (ℓ, Z) a parent denoted $(\ell, Z).\mathit{parent}$ (Line 16). We also associate to each symbolic state a bound K on width, denoted $(\ell, Z).K$.

5 Symbolic Robust Safety

Our semi-algorithm consists of a zone-based exploration with IEDBMs using the parametric LU-abstraction and the inclusion algorithm $\sqsubseteq^c$ of Lemma 9. It is easy to see that an exploration based on IEDBMs may not terminate in general (see *e.g.* Fig. 1). Nevertheless, we apply acceleration on well chosen cycles while it is exploring the state space, and it terminated in most of our experiments. To choose the cycles to accelerate, we adopt a lazy approach: we fix a bound K, and run the forward search using IEDBMs until the target is reached or some symbolic state has width greater than K. In the latter case, we examine the prefix of the current state, and accelerate its cycles by Lemma 1. If no new state is obtained, then we increment the bound K for the current branch and continue the exploration. We thus interpret a large width as the accumulation of imprecisions due to cycles. No cycle may be responsible for a large width, in which case we increase the width threshold and continue the exploration.

We establish the correctness of our semi-algorithm in the following lemma. When it answers Unsafe, it has found a path that ends in the target state, and the proof shows that such a run exists in all $\mathcal{A}_\nu$ for $\nu > 0$. If it answers Safe, then it has terminated without visiting the target state. If δ_0 denotes the upper

bound on δ after termination, the proof shows that for all $\nu \in [0, \delta_0)$, an exact exploration applied on $\mathcal{A}_\nu$ would visit the same symbolic states as our algorithm when the IEDBMs are instantiated with $\delta \leftarrow \nu$. In other words, the exploration search tree uniformly represents all the search trees that would be generated by an exact algorithm applied on $\mathcal{A}_\nu$ for $\nu \in [0, \delta_0)$.

Lemma 12 (Correctness). *For any timed automaton $\mathcal{A}$ and location ℓ_T, if Algorithm 1 answers Unsafe then for all $\nu > 0$, ℓ_T is reachable in $\mathcal{A}_\nu$ from the initial state. If it answers Safe, then if δ_0 denotes the upper bound on δ after termination, then for all $\nu \in [0, \delta_0)$, $\mathcal{A}_\nu$ does not visit ℓ_T.*

6 Experimental Evaluation

In this section, we evaluate the performance of our semi-algorithm on several benchmarks from the literature; most of which are available from `www.uppaal.org`, and have been considered in [21], with the exception of the scheduling tests (Sched *) which were constructed from the experiments of [17]. We implemented Alg. 1 in OCaml in a tool called `Symrob` (*sym*bolic *rob*ustness, available from `www.ulb.ac.be/di/verif/sankur`). We consider two other competing algorithms: the first one is the recently published tool Verifix [21] which solves the infinitesimal robust safety problem but does not output any bound on δ. The second algorithm is our implementation of a binary search on the values of δ which iteratively calls an exact model checker until a given precision is reached.

The exact model checking algorithm is a forward exploration with DBMs using LU extrapolation and the inclusion test of [19] implemented in `Symrob`. We do not use advanced tricks such as symmetry reduction, federations of zones, and clock decision diagrams; see *e.g.* [4]. The reason is that our goal here is to compare *algorithms* rather than *software tools*. These optimizations could be added to the exact model checker but also to our robust model checker (by adapting to IEDBMs), but we leave this for future work.

In Table 1, the number of visited symbolic states (as IEDBMs for `Symrob` and as DBMs for `Verifix`) and the running times are given. On most benchmarks `Symrob` terminated faster and visited less states. We also note that `Symrob` actually computed the *largest* δ below which safety holds for the benchmarks CSMA/CD and Fischer. One can indeed check that syntactically enlarging the guards by 1/3 (resp. 1/2) makes the respective classes of benchmarks unsafe (Recall that the upper bound δ_0 is always strict in IEDBMs). On one benchmark, `Verifix` wrongly classified the model as non-robust, which could be due to a bug or to the presence of non-progress cycles in the model (see [11]).

Table 2 shows the performance of the binary search for varying precision $\epsilon \in \{\frac{1}{10}, \frac{1}{20}, \frac{1}{40}\}$. With precision $\frac{1}{10}$, the binary search was sometimes faster than `Symrob` (*e.g.* on CSMA/CD), and sometimes slower (*e.g.* Fischer); moreover, the computed value of δ was underestimated in some cases (*e.g.* CSMA/CD and Fischer benchmarks). With precision $\frac{1}{20}$, more precision was obtained on δ but at a cost of an execution time that is often worse than that of `Symrob` and

Table 1. Comparison between `Symrob` (breadth-first search, instantiated with $K_0 = 10$) and `Verifix` [21]. The running time of the exact model checking implemented in `Symrob` is given for reference in the column "Exact" (the specification was satisfied without enlargement in all models). Note that the visited number of states is not always proportional to the running time due to additional operations performed for acceleration in both cases. The experiments were performed on an Intel Xeon 2.67 GHz machine.

Benchmark	**Robust** – δ		**Visited States**		**Time**		
	Symrob	Verifix	Symrob	Verifix	Symrob	Verifix	Exact
CSMA/CD 9	Yes – 1/3	Yes	147,739	1,064,811	61s	294s	42s
CSMA/CD 10	Yes – 1/3	Yes	398,354	846,098	202s	276s	87s
CSMA/CD 11	Yes – 1/3	Yes	1,041,883	2,780,493	12m	26m	5m
Fischer 7	Yes – 1/2	Yes	35,029	81,600	11s	12s	6s
Fischer 8	Yes – 1/2	Yes	150,651	348,370	45s	240s	24s
Fischer 9	Yes – 1/2	Yes	627,199	1,447,313	4m	160m	2m20s
MutEx 3	Yes – 1000/11	Yes	37,369	984,305	3s	131s	3s
MutEx 4	No	No	195,709	146,893	16s	41s	4s
MutEx 4 fixed	Yes – 1/7	–	5,125,927	–	38m	>24h	7m
Lip Sync	–	No	–	29,647,533	>24h	14h	5s
Sched A	Yes – 1/4	**No***	9,217	16,995	11s	248s	2s
Sched B	No	–	50,383	–	105s	>24h	40s
Sched C	No	No	5,075	5,356	3s	29s	2s
Sched D	No	No	15,075	928	2s	0.5s	0.5s
Sched E	No	No	31,566	317	5s	0.5s	0.5s

Table 2. Performance of binary search where the initial enlargement is 8, and the required precision ϵ is either 1/10, 1/20 or 1/40. Note that when the model is not robust, the binary search is inconclusive. Nonetheless, in these cases, we do know that the model is unsafe for the smallest δ for which we model-checked the model. In these experiments the choice of the initial condition (here, $\delta = 8$) wasn't significant since the first iterations always took negligeable time compared to the case $\delta < 1$.

Benchmark	**Robust** – δ		**Visited States**		**Time**		
	$\epsilon = 1/10$	$\epsilon = 1/20$	$\epsilon = 1/10$	$\epsilon = 1/20$	$\epsilon = 1/10$	$\epsilon = 1/20$	$\epsilon = 1/40$
CSMA/CD 9	Yes – 1/4	Yes – 5/16	151,366	301,754	43s	85s	123s
CSMA/CD 10	Yes – 1/4	Yes – 5/16	399,359	797,914	142s	290s	428s
CSMA/CD 11	Yes – 1/4	Yes – 5/16	1,043,098	2,085,224	8m20s	17m	26m
Fischer 7	Yes – 3/8	Yes – 7/16	75,983	111,012	15s	21s	31s
Fischer 8	Yes – 3/8	Yes – 7/16	311,512	462,163	53s	80s	129s
Fischer 9	Yes – 3/8	Yes – 7/16	1,271,193	1,898,392	5m	7m30s	12m
MutEx 3	Yes – 8	Yes – 8	37,369	37,369	2s	2s	2s
MutEx 4	Inconclusive		1,369,963	1,565,572	1m5s	1m15s	1m30s
MutEx 4 fix'd	Yes – 5/8	Yes – 9/16	6,394,419	9,864,904	9m30s	17m	25m
Lip Sync	Inconclusive		–	–	>24h	>24h	>24h
Sched A	Yes – 7/16	Yes – 15/32	27,820	37,101	6s	9s	11s
Sched B	Inconclusive		109,478	336,394	35s	140s	20m
Sched C	Inconclusive		10,813	36,646	2s	6s	56s
Sched D	Inconclusive		27,312	182,676	2s	9s	60s
Sched E	Inconclusive		98,168	358,027	6s	17s	95s

systematically more states to visit. Increasing the precision to $\frac{1}{40}$ leads to even longer execution times. On non-robust models, a low precision analysis is often fast, but since the result is inconclusive, one rather increases the precision, leading to high execution times. The binary search can be made complete by choosing the precision exponentially small [11] but this is too costly in practice.

7 Conclusion

We presented a symbolic procedure to solve the quantitative robust safety problem for timed automata based on infinitesimally enlarged DBMs. A good performance is obtained thanks to the abstraction operators we lifted to the parametric setting, and to the lazy approach used to accelerate cycles. Although no termination guarantee is given, we were able to treat several case studies from the literature, demonstrating the feasability of robustness verification, and the running time was often comparable to that of exact model checking. Our experiments show that binary search is often fast if run with low precision; however, as precision is increased the gain of a parametric analysis becomes clear. Thus, both approaches might be considered depending on the given model.

An improvement over binary search for a problem of refinement in timed games is reported in [23]; this might be extended to our problem as well. Both our tool and Verifix fail when a large number of cycles needs to be accelerated, and this is difficult to predict. An improvement could be obtained by combining our lazy acceleration technique using the combined computation of the cycles of [21]. An extension to LTL objectives could be possible using [9].

References

1. K. Altisen and S. Tripakis. Implementation of timed automata: An issue of semantics or modeling? In FORMATS'05, LNCS 3829, p. 273–288. Springer, 2005.
2. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
3. A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In CAV'01, LNCS 1855, p. 419–434. Springer, 2000.
4. G. Behrmann, J. Bengtsson, A. David, K. G. Larsen, P. Pettersson, and W. Yi. Uppaal implementation secrets. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 2469, p. 3–22. Springer Berlin Heidelberg, 2002.
5. G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelanek. Lower and upper bounds in zone-based abstractions of timed automata. *Int. J. Softw. Tools Technol. Transf.*, 8(3):204–215, 2006.
6. G. Behrmann, A. David, K. G. Larsen, J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. UPPAAL 4.0. In QEST'06, p. 125–126, 2006.
7. J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, LNCS 2098, p. 87–124. Springer, 2004.
8. P. Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004.

9. P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of linear-time properties in timed automata. In LATIN'06, LNCS 3887, p. 238–249. Springer, 2006.
10. P. Bouyer, N. Markey, and P.-A. Reynier. Robust analysis of timed automata via channel machines. In FoSSaCS'08, LNCS 4962, p. 157–171. Springer, 2008.
11. P. Bouyer, N. Markey, and O. Sankur. Robust model-checking of timed automata via pumping in channel machines. In FORMATS'11, LNCS 6919, p. 97–112, Aalborg, Denmark, 2011. Springer.
12. M. Bozga, S. Graf, and L. Mounier. If-2.0: A validation environment for component-based real-time systems. In CAV'04, LNCS 2404, p. 343–348. Springer, 2002.
13. C. Daws and P. Kordy. Symbolic robustness analysis of timed automata. In FORMATS'06, LNCS 4202, p. 143–155. Springer, 2006.
14. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robust safety of timed automata. *Formal Methods in System Design*, 33(1-3):45–84, 2008.
15. M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: From timed models to timed implementations. *Formal Aspects of Computing*, 17(3):319–341, 2005.
16. D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In AVMFSS'89, LNCS 407, p. 197–212. Springer, 1990.
17. G. Geeraerts, J. Goossens, and M. Lindström. Multiprocessor schedulability of arbitrary-deadline sporadic tasks: Complexity and antichain algorithm. *Real-Time Systems, The International Journal of Time-Critical Computing Systems*, 48(2), 2013.
18. V. Gupta, Th. A. Henzinger, and R. Jagadeesan. Robust timed automata. In HART'97, LNCS 1201, p. 331–345. Springer, 1997.
19. F. Herbreteau, D. Kini, B. Srivathsan, and I. Walukiewicz. Using non-convex approximations for efficient analysis of timed automata. In FSTTCS'11, p. 78–89, 2011.
20. R. Jaubert and P.-A. Reynier. Quantitative robustness analysis of flat timed automata. In FOSSACS'11, LNCS 6604, p. 229–244. Springer, 2011.
21. P. Kordy, R. Langerak, S. Mauw, and J. W. Polderman. A symbolic algorithm for the analysis of robust timed automata. In FM'14, LNCS 8442, p. 351–366. Springer, 2014.
22. K. G. Larsen, A. Legay, L.-M. Traonouez, and A. Wasowski. Robust specification of real time components. In FORMATS'11, LNCS 6919, p. 129–144. Springer, 2011.
23. A. Legay and L.-M. Traonouez. Pyecdar: Towards open source implementation for timed systems. In *ATVA'13*, LNCS 8172, p. 460–463. Springer, 2013.
24. A. Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.
25. M. Swaminathan and M. Franzle. A symbolic decision procedure for robust safety of timed systems. In TIME'07, p. 192–192, 2007.
26. M. Swaminathan, M. Frnzle, and J.-P. Katoen. The surprising robustness of (closed) timed automata against clock-drift. In IFIP TCS'08, p. 537–553. Springer, 2008.

A Proofs of Section 3 (Accelerating Cycles)

In this section, we present the proof of Lemma 1.

Let us first recall two fundamental properties of cycles from [24]. Define $L_\rho = \{q \mid q \xrightarrow{\rho} q\}$ the set of *limit points of cycle* ρ. The set of limit points play an

important role in the analysis of enlarged timed automata; it was shown that for any $\nu > 0$, there is a run between any pair of states of L_ρ in $\mathcal{A}_\nu$.

Lemma 13 ([24,14]). *For any cycle ρ and $\nu > 0$, and any $q, q' \in L_\rho$, $q \xrightarrow{(\rho)^+_\nu} q'$.*

Now, before proving Lemma 1, we just need the following property on regions which states that L_ρ is reachable and co-reachable from $\bar{r}$[1], where r denotes a region from which there is a run along ρ.

Lemma 14 ([24,14]). *Consider any cycle ρ and region r such that $q \xrightarrow{\rho} q'$ for some $q, q' \in [\![\bar{r}]\!]$. Then, for any $q \in [\![\bar{r}]\!]$, there exist $q_0, q_1 \in L_\rho$ such that $q \xrightarrow{\rho^+} q_0$, and $q_1 \xrightarrow{\rho^+} q$.*

Intuitively, the proof of Lemma 1 follows by the previous two lemmas: in fact, by repeating ρ one can reach any state of L_ρ (Lemmas 14, 13), and that from some state of L_ρ the desired target state is reachable (Lemma 14). This is the original proof idea of [24]. Here we follow this idea to generalize it to zones given by the fixpoints $\mathsf{Pre}^*(\cdot)$ and $\mathsf{Post}^*(\cdot)$.

Proof (Lemma 1). Note first that both fixpoints converge since all clocks are bounded above so the number of zones in $\mathcal{A}$ and $\mathcal{A}_\nu$ is finite. Let $q \in \mathsf{Pre}^*_\rho(\top)$. By definitition of this fixpoint, for any $n > 0$, there exists a run in $\mathcal{A}_0$ with $q = q_1 \xrightarrow{\rho} \dots \xrightarrow{\rho} q_n$. So there exist $i < j$ such that $r = \mathsf{reg}(q_i) = \mathsf{reg}(q_j)$. Note that since the clocks are bounded $\bar{r}$ is compact. It follows that $\bar{r} \cap L_\rho \neq \emptyset$. Moreover, we know by [24] that the set L_ρ is a union of regions. So there exists a region $r' \subseteq \bar{r}$ such that $r' \subseteq L_\rho$. Now, because all guards are closed, if r is reachable by repeating ρ, r' is also reachable along the same path. In other terms, $q \xrightarrow{\rho^+} q_0$ for some $q_0 \in L_\rho$. Now, similarly, for any $q' \in \mathsf{Post}^*_{(\rho)_\nu}(\top)$, we consider a long run $q'_1 \xrightarrow{(\rho)_\nu} \dots \xrightarrow{(\rho)_\nu} q'_n = q'$, so that some region is seen twice, and we deduce (as above) that $L_\rho \ni q_1 \xrightarrow{(\rho)^+_\nu} q'$. Furthermore, by Lemma 13, we have $q_0 \xrightarrow{(\rho)^+_\nu} q_1$, which proves the claim.

B Proofs of Section 4

B.1 Operations on IEDBMs

Proof (Lemma 3). We define the minimum as follows.

$$\min\left((m,p)_{\langle 0,\delta_0\rangle}, (n,q)_{\langle 0,\delta_0\rangle}\right) = \begin{cases} (m,p)_{\langle 0,\delta_0\rangle} & \text{if } m \leq n \wedge p \leq q \\ (m,p)_{\langle 0,\min(\delta_0,\frac{n-m}{p-q})\rangle} & \text{if } m < n \wedge p > q \\ (n,q)_{\langle 0,\delta_0\rangle} & \text{if } n \leq m \wedge q \leq p \\ (n,q)_{\langle 0,\min(\delta_0,\frac{m-n}{q-p})\rangle} & \text{if } n < m \wedge q > p \end{cases} \quad (2)$$

[1] Here, $\bar{r}$ denotes the topological closure of r.

If $m \leq n, p \leq q$, then clearly $m + \nu p \leq n + \nu q$ for all $\nu \geq 0$. In particular, this holds for $\nu \in [0, \nu)$. If $m < n$ and $p > q$, then $m + \nu p \leq n + \nu q \Leftrightarrow \nu \leq \frac{n-m}{p-q}$. The second line follows. The other two cases are symmetric.

Proof (Lemma 4). As for usual DBMs, we consider the Floyd-Warshall shortest path algorithm applied on the matrix (M, P) seen as an adjacency matrix. We recall the algorithm:

```
1 for i ∈ C_0 do
2 |   for j ∈ C_0 do
3 |   |   for k ∈ C_0 do
4 |   |   |   (M,P)[i,j] := min((M,P)[i,j], (M,P)[i,k] + (M,P)[k,j]);
5 |   |   end
6 |   end
7 end
```

Note that the validity of this reduction algorithm applied on usual DBMs –*i.e.* with P = 0, is well known (see [7]).

The algorithm terminates after performing $|\mathcal{C}_0|^3$ operations on the components of the matrix. Let $(M', P')_{\langle 0,\delta_1\rangle}$ denote the result of the above algorithm applied on IEDBM $(M, P)_{\langle 0,\delta_0\rangle}$. For all $\nu \in [0, \delta_1)$, let $A_\nu = M + \nu P$, and let A'_ν be the result of the above reduction algorithm applied on the DBM A_ν. We know that all A'_ν are reduced for $\nu \in [0, \delta_1)$.

Now, by unraveling the computations of the algorithm, each component x, y of (M', P') can be written as a finite expression (of depth $O(|\mathcal{C}_0|^3)$) involving the components of (M, P). Moreover, by definition of the operations on IEDBMs, this equality holds when we instantiate δ by any $\nu \in [0, \delta_1)$. In particular, we must have $A'_\nu[x, y] = (M' - \nu P')[x, y]$. In fact, the algorithm applied on A_ν performs the same operations in the same order, and the minima are resolved in favor of the same expression, because ν is assumed to be in $[0, \delta_1)$. It follows that each $M' - \nu P'$ is a reduced DBM.

It is now easy to see that (M', P') is a reduced IEDBM. In fact, assuming otherwise means that for some $x, y, z \in \mathcal{C}_0$, $(M', P')[x, z] + (M', P')[z, y] > (M', P')[x, y]$, which implies that $(M' - \nu P')[x, z] + (M' - \nu P')[z, y] > (M' - \nu P')[x, y]$ for all $\nu \in [0, \delta_2)$ for some $\delta_2 > 0$. This contradicts the fact that A'_ν is reduced.

Proof (Lemma 5). Recall that all operations on classic DBMs consist in first modifying the components of the given DBM, and then reducing [7].

For the intersection of two DBMs $N_1 + \nu Q_1, N_2 + \nu Q_2$, one first assigns to each cell (x, y) the minimum of $(N_1 + \nu Q_1)[x, y]$ and $(N_2 + \nu Q_2)[x, y]$, and then reduces the resulting DBM. For IEDBMs, one similarly assigns the minimum to each cell according to Lemma 3. After this operation, all cells are equal to $\min((N_1 + \nu Q_1), (N_2 + \nu Q_2))$ for all $\nu \in [0, \delta_1)$, where δ_1 is the bound on δ obtained after these updates. After reduction by Lemma 4, let δ_2 denote the new

upper bound on δ. Then for all $\nu \in [0, \delta_2)$, the obtained IEDBM is equal, when δ is instantiated to ν, to the reduced DBM $(N_1 + \nu Q_1) \cap (N_2 + \nu Q_2)$.

The reset operation requires one to free all constraints on the rows and column involving R (replacing them with ∞), and then setting each column $x \in R$ to 0, and reducing. The time successors operation is defined by freeing the first column (that is, removing all upper bounds), and reducing. Similarly to the intersection case, because each operation only requires a finite number of steps, it is easily shown that when these operations are performed on IEDBM (M, P), the resulting IEDBM represents the result of the corresponding operation applied on $M - \nu P$ for small enough ν.

In all cases one modifies each component at most one, and then reduces so the overall complexity of each operation is $O(|\mathcal{C}_0|^3)$.

Proof (Lemma 6). Assume $\mathtt{PPost}_{(e_1)_\delta (e_2)_\delta \dots (e_{n-1})_\delta}((\ell, Z)) = (\ell', Z')_{\langle 0, \delta_0 \rangle}$, and $(\ell', Z')_{\langle 0, \delta_0 \rangle} \neq \emptyset$. By 5, it follows by a straightforward induction on the length of the given path that $\mathsf{Post}_{(e_1)_\nu \dots (e_{n-1})_\nu}(\ell, Z[\delta \leftarrow \nu]) = (\ell', Z'[\delta \leftarrow \nu]) \neq \emptyset$ for all $\nu \in [0, \delta_0)$. Then the existence of the stated run follows from the properties of the non-parametric zones and the $\mathsf{Post}(\cdot)$ operation.

Proof (Lemma 7). For the originial definition of Extra^+_{LU}, these properties are proven in [8,5]. However the original definition takes into account strict and non-strict inequalities in DBMs, and the third case of (1) assigns $(-U_y, <)$. Here, we only need to prove that the modified operator is still sound and complete for reachability.

If $\mathsf{Extra}^+_{LU}{}^o$ denotes the original definition, we therefore have $\mathsf{Extra}^+_{LU}(Z) \subseteq \mathsf{Extra}^+_{LU}{}^o(Z)$. Moreover, the condition $-M_{0,y} > U_y$ means that $-M_{0,y} \geq U_y + 1$, so we also have $Z \subseteq \mathsf{Extra}^+_{LU}(Z)$ (since we have $Z \subseteq \mathsf{Extra}^+_{LU}{}^o(Z)$ and other components are identical in Extra^+_{LU} and $\mathsf{Extra}^+_{LU}{}^o$). Since the abstraction $\mathsf{Extra}^+_{LU}{}^o$ is sound and complete, it follows that Extra^+_{LU} is also sound and complete.

B.2 Parametric Abstractions

Proof (Lemma 8). The proof follows the same ideas for IEDBMs seen above. For fixed $\nu > 0$, one computes $\mathsf{Extra}^+_{LU}(M + \nu P)$ simply by replacing M by $M + \nu P$ in (1), L by L_ν, and U by U_ν. But the definition of Extra^+_{LU} relies on a finite number of inequalities whose satisfactions are constant for computable small enough δ_0. So the result follows from Lemma 3.

Before proving the correctness of the parametric inclusion test of Lemma 9, we recall the exact inclusion test of [19] for DBMs.

Lemma 15 ([19]). *Given a timed automaton $\mathcal{A}$, its LU-bounds L, U and $\alpha = \max(L, U)$, if M and N denote two DBMs, we have $M \not\subseteq \mathsf{Closure}_\alpha(N)$ if, and only if there exist $x, y \in \mathcal{C}$ such that*

1. *$N_{x,0} < M_{x,0}$ and $N_{x,0} \leq \alpha(x)$, or*
2. *$N_{0,x} < M_{0,x}$ and $M_{0,x} + \alpha(x) \geq 0$, or*

3. $M_{0,x} + \alpha(x) \geq 0$*, and* $N_{y,x} < M_{y,x}$*, and* $N_{y,x} \leq \alpha(y) + M_{0,x}$*.*

Proof (Lemma 9). Let us recall the property to be tested:

$$\forall \delta_0 > 0, \exists \nu \in [0, \delta_0), M + \nu P \not\subseteq \mathsf{Closure}_{\alpha_\nu}(N + \nu Q). \tag{3}$$

Assume there exist $x, y \in \mathcal{C}$ satisfying one of the conditions, say ${Z'}_{x,0} < Z_{x,0}$ and ${Z'}_{x,0} \leq \alpha_\delta(x)$. For $\nu > 0$, let us define $Z_\nu = M + \nu P$, and $Z'_\nu = N + \nu Q$. Let $\delta_1 > 0$ small enough so that

$${Z'_\nu}^+[x,0] < Z_\nu[x,0] \text{ and } {Z'}_\nu^+[x,0] \leq \alpha_\nu(x)$$

for all $\nu \in [0, \delta_1)$. By Lemma 15, this implies that $M + \nu P \not\subseteq \mathsf{Closure}_{\alpha_\nu}(N + \nu Q)$ for all $\nu \in [0, \delta_1)$, so (3) holds.

Conversely, assume that none of the three conditions hold for no pair $x, y \in \mathcal{C}$. If $\phi_1(x,y), \phi_2(x,y), \phi_3(x,y)$ denote the three conditions written for a pair $x, y \in \mathcal{C}$, then we have $\Phi = \wedge_{x,y \in \mathcal{C}}(\neg\phi_1(x,y) \wedge \neg\phi_2(x,y) \wedge \neg\phi_3(x,y))$. So let $\delta_1 > 0$ be small enough such that Φ holds when when instantiated by any $\nu \in [0, \delta_1)$ (that is, when written with Z_ν instead of Z, and Z'_ν instead of Z'). By Lemma 15, this means that for all $\nu \in [0, \delta_1)$, $M + \nu P \subseteq \mathsf{Closure}_{\alpha_\nu}(N + \nu Q)$ which is the converse of (3).

C Proof of Lemma 12

Assume the algorithm returns Unsafe.
We consider the path π of the search tree from the initial state to location ℓ_T. Let $(\ell_1, Z_1) \dots (\ell_n, Z_n)$ be the set of symbolic states visited by the path, and $e_1 e_2 \dots e_{n-1}$ the transitions along the search tree. By construction, each e_j is either a regular edge, in which case $(\ell_{j+1}, Z_{j+1}) = \mathtt{PExPost}_{(e_j)_\delta}(\ell_j, Z_j)$, or it is a cycle ρ that starts at location ℓ_j, such that $\mathtt{PPre}^*_\rho(\top) \sqcap (\ell_j, Z_j) \neq \emptyset$ and $(\ell_{j+1}, Z_{j+1}) = \mathtt{PPost}^*_{(\rho)_\delta}(\top)$. Note that the algorithm rather checks $\mathtt{PPre}^*_\rho(\top) \sqcap (\ell_i, Z_i) \neq \emptyset$ but this implies that $\mathtt{PPre}^*_\rho(\top) \sqcap (\ell_j, Z_j) \neq \emptyset$, given that $(\ell_j, Z_j) = \mathtt{PPost}_{(\rho)_\delta}((\ell_i, Z_i))$, by definition of the $\mathtt{PPre}^*_\rho(\top)$ fixpoint.

For all $\nu > 0$, we construct a run from the initial state to ℓ_T in $\mathcal{A}_\nu$ by induction on the number of accelerated cycles visited by π. Let us assume that $\nu \in (0, \delta_0]$ where δ_0 is smaller than all upper bounds computed during the execution of the algorithm. We will construct a run for $\mathcal{A}_\nu$ for arbitrary $\nu \in (0, \delta_0]$. Moreover, since any such run is also a run of $\mathcal{A}_{\nu'}$ for $\nu' > \delta_0$, the result will follow.

For the base case, assume that π is only made of regular edges. We have then $\mathtt{PExPost}_{(e_1)_\delta (e_2)_\delta \dots (e_{n-1})_\delta}((\ell_1, Z_1)) = (\ell_n, Z_n)$. This means that $\mathsf{ExPost}_{(e_1)_\nu \dots (e_{n-1})_\nu}((\ell_1, Z_1)[\delta \leftarrow \nu]) = (\ell_n, Z_n)[\delta \leftarrow \nu]$. By Lemma 7, $\mathsf{Post}_{(e_1)_\nu \dots (e_{n-1})_\nu}((\ell_1, Z_1)[\delta \leftarrow \nu]) \neq \emptyset$, there exists a run from the initial state which is in $(\ell_1, Z_1)[\delta \leftarrow \nu]$ to a state in $(\ell_n, Z_n)[\delta \leftarrow \nu]$ in $\mathcal{A}_\nu$.

Assume now that π contains at least one accelerated cycle. Consider e_{i_0} the last such accelerated cycle. By the base case, from some state q_1 in $(\ell_{i_0+1}, Z_{i_0+1})[\delta \leftarrow \nu]$, there is a run r_1 to $(\ell_n, Z_n)[\delta \leftarrow \nu]$ along edges

$(e_{i_0+1})_\nu \dots (e_{n-1})_\nu$. Let us choose any state $q_2 \in \big(\mathtt{PPre}^*_\rho(\top) \sqcap (\ell_{i_0}, Z_{i_0})\big)[\delta \leftarrow \nu]$. By Lemma 1, there is a run r_2 from q_2 to q_1. By induction, there is a run r_3 in $\mathcal{A}_\nu$ from the initial state to q_2. Then, $r_3 r_2 r_1$ yields the desired run.

Assume the algorithm returns Safe.
Let δ_0 denote the upper bound on δ after the termination of the algorithm. We are going to show that $\mathcal{A}_\nu$ is safe for all $\nu \in [0, \delta_0)$. We define algorithm Alg_ν from Algorithm 1 by instantiating the parameter δ by ν in IEDBMs. The resulting symbolic states are now written as $M + \nu P$ which are simply DBMs. The new algorithm will work with DBMs, but we will still write the symbolic states in the form $M + \nu P$. This allows us to define the *width* in this case which is defined again as the maximal of the components of P. More precisely, Alg_ν is defined by the following mofidications applied on Algorithm 1: All occurrences of δ are replaced by ν, and each parametric operation is replaced by its non-parametric counterpart, *e.g.* $\mathtt{PPost}(\cdot)$ by $\mathsf{Post}(\cdot)$. In other terms, given a timed automaton $\mathcal{A}$, Alg_ν simply explores the state space of the timed automaton $\mathcal{A}_\nu$ using DBMs and moreover applies acceleration on some cycles.

Let us first argue that if Alg_ν answers Safe, then $\mathcal{A}_\nu$ is indeed safe. This is easy to see since if we ignore the acceleration phase, Alg_ν is simply the standard forward exploration algorithm using DBMs and LU extrapolation, and the post operation is always applied on the guards enlarged by ν. This suffices to prove that when Alg_ν answers Safe, $\mathcal{A}_\nu$ is safe. In fact, the acceleration phase can only add new states to explore, so any state (ℓ, Z) visited in the basic forward exploration (without acceleration) will be still visited; more precisely, a state (ℓ, Z') with $Z \sqsubseteq^c Z'$ will be visited by Alg_ν. So if location ℓ_T is reachable in $\mathcal{A}_\nu$, Alg_ν would visit a state at this location. (As a side note, observe that by Lemma 1, it is easy to see that the acceleration phase actually only yields states that are reachable in $\mathcal{A}_\nu$. So Alg_ν is sound and complete.)

Now, to finish the proof, we are going to show that if (ℓ, Z) denotes the parametric symbolic state visited by Algorithm 1 at iteration i, then Alg_ν visits the state $(\ell, Z)[\delta \leftarrow \nu]$ at step i. In other terms, Alg_ν constructs exactly the same search tree as Alg. 1 where δ is instantiated to ν. This follows immediately by the properties of the IEDBMs. Recall that by the choice of δ_0 *all* operations performed on IEDBMs during the execution of Alg. 1 hold when we instantiate the parameter δ to any value in $[0, \delta_0)$. In particular, for each parametric successor computation of Alg. 1, Alg_ν performs the same computation where δ is replaced by ν, and the same inclusion tests. Now, if we assume, towards a contradiction, that Alg_ν encounters location ℓ_T in the search tree, this shows that the same prefix (where ν is replaced by parameter δ) is also explored by Alg. 1.