Reachability in Timed Automata with Diagonal Constraints

Paul Gastin
LSV, CNRS, ENS Paris-Saclay, Université Paris–Saclay, France
gastin@lsv.fr

Sayan Mukherjee
Chennai Mathematical Institute, India
sayanm@cmi.ac.in

B. Srivathsan
Chennai Mathematical Institute, India
sri@cmi.ac.in

Abstract

We consider the reachability problem for timed automata having diagonal constraints (like $x - y < 5$) as guards in transitions. The best algorithms for timed automata proceed by enumerating reachable sets of its configurations, stored in a data structure called "zones". Simulation relations between zones are essential to ensure termination and efficiency. The algorithm employs a simulation test $Z \preccurlyeq Z'$ which ascertains that zone Z does not reach more states than zone Z', and hence further enumeration from Z is not necessary. No effective simulations are known for timed automata containing diagonal constraints as guards. We propose a simulation relation $\preccurlyeq^d_{LU}$ for timed automata with diagonal constraints. On the negative side, we show that deciding $Z \not\preccurlyeq^d_{LU} Z'$ is NP-complete. On the positive side, we identify a witness for $Z \not\preccurlyeq^d_{LU} Z'$ and propose an algorithm to decide the existence of such a witness using an SMT solver. The shape of the witness reveals that the simulation test is likely to be efficient in practice.

2012 ACM Subject Classification Theory of computation → Verification by model checking

Keywords and phrases Timed Automata, Reachability, Zones, Diagonal constraints

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2018.28

Related Version A full version of the paper is available at [12], `https://arxiv.org/abs/1806.11007`.

Funding Partly supported by UMI RELAX, Infosys foundation (India) and Tata Consultancy Services - Innovation Labs (Pune, India)

1 Introduction

Timed automata [1] are models of real-time systems. They are finite automata equipped with real valued variables called *clocks*. These clocks can be used to constrain the time difference between events: for instance when an event a occurs a clock x can be set to 0 in the transition reading a, and when an event b occurs, the transition reading b can check if $x \leq 4$. These constraints on clocks are called *guards* and clocks which are made 0 in a transition are said to be *reset* in the transition. Guards of the form $x - y > 5$ are called *diagonal constraints*. They are convenient for checking conditions about events in the past: when an event c occurs, we want to check that between events a, b which occurred previously (in the said order), the time gap is at least 5. One can then reset a clock x at a, y at b

and check for $x - y > 5$ at c. It is known that such diagonal constraints do not add to the expressive power: each timed automaton can be converted into an equivalent one with no diagonal guards, that is, a *diagonal-free* automaton [4]. However, this conversion leads to an exponential blowup in the number of states, which is unavoidable in general [6].

State reachability is a basic question in timed automata verification. The problem is to decide if there exists a run of the automaton from the initial state to a given accepting state. This is known to be PSPACE-complete [1]. In practice, the best algorithms for reachability proceed by a forward analysis of the automaton: starting from its initial state, enumerate reachable sets of its configurations stored in the form of a data structure called *zones*. Zones are conjunctions of difference constraints (like $x - y < 6 \ \wedge \ w > 4$) which can be efficiently represented and manipulated using Difference Bound Matrices [11]. *Abstractions* of zones are necessary for termination and efficiency of this enumeration. These abstractions are functions with a finite range mapping each set of configurations to a bigger set. For diagonal free timed automata various implementable abstraction functions are known [2, 16]. For timed automata with diagonal constraints, no such abstraction functions are known and such a forward analysis method does not work. A naïve method would be to analyze the equivalent diagonal free automaton, but then this introduces a (systematic) blowup.

Abstractions of zones can be used in two ways during the forward analysis: explicitly or implicitly. In the explicit case, each time a new zone Z appears, the abstraction function $\mathfrak{a}$ is applied on it and $\mathfrak{a}(Z)$ is stored. Further enumeration starts from $\mathfrak{a}(Z)$. For this explicit method to work, $\mathfrak{a}(Z)$ needs an efficient representation. Hence only abstractions where $\mathfrak{a}(Z)$ is also a zone (also called *convex abstractions*) are used. Extra^+_{LU}[2] is the best known convex abstraction for diagonal free automata and is implemented in the state-of-the-art tool UPPAAL [3]. In the implicit case, zones are not extrapolated and are stored as they are. Each time a new zone Z appears, it is checked if there exists an already visited zone Z' such that $Z \subseteq \mathfrak{a}(Z')$. Intuitively this means that zone Z cannot see more states than Z' and hence the enumeration at Z can stop. Given that $\mathfrak{a}$ has finite range, the computation terminates. Since abstractions of zones are not stored explicitly, there is no restriction for $\mathfrak{a}$ to result in a zone, but an efficient inclusion test $Z \subseteq \mathfrak{a}(Z')$ is necessary as this test is performed each time a new zone appears. For diagonal-free automata, the best known abstraction is $\mathfrak{a}_{\preccurlyeq LU}$ and it subsumes Extra^+_{LU}. The inclusion test $Z \subseteq \mathfrak{a}_{\preccurlyeq LU}(Z')$ can be done in $\mathcal{O}(|X|^2)$ where X is the number of clocks [16]. In both cases - explicit or implicit - it is important to have an abstraction that transforms zones into as big sets as possible, so that the enumeration can terminate with fewer zone visits.

In this paper, we are interested in the implicit method for timed automata with diagonal constraints. Since the abstractions that are usually used are based on simulation relations, the inclusion test $Z \subseteq \mathfrak{a}(Z')$ boils down to a simulation test $Z \preccurlyeq Z'$ between zones. In particular, the $\mathfrak{a}_{\preccurlyeq LU}$ abstraction is based on a simulation relation $\preccurlyeq_{LU}$ [2]. We choose to take this point of view: from the next section, we refrain from using abstractions and present them as simulations instead. We propose a simulation $\preccurlyeq^d_{LU}$ that is sound for diagonal constraints. Contrary to the diagonal free case, we show that the simulation test $Z \not\preccurlyeq^d_{LU} Z'$ is NP-complete. But on the positive side, we give a characterization of a witness for the fact that $Z \not\preccurlyeq^d_{LU} Z'$ and encode the existence of such a witness as the satisfiability of a formula in linear arithmetic. This gives an algorithm for $Z \not\preccurlyeq^d_{LU} Z'$. The shape of the witness shows that in practice the number of potential candidates would be low and the simulation test is likely to be efficient. We have implemented our algorithm in a prototype tool. Preliminary experiments demonstrate that the number of zones enumerated using $\preccurlyeq^d_{LU}$ simulation drastically reduces compared to the number of zones obtained by doing the diagonal free conversion followed by a forward analysis using $\preccurlyeq_{LU}$. This simulation relation $\preccurlyeq^d_{LU}$ and the associated simulation test also open the door for extending optimizations studied for diagonal free automata [15], to the case of diagonal constraints; and also extending analysis of priced timed automata with diagonal constraints [7, 18].

Related work. Convex abstractions used for diagonal free timed automata had been in use also for diagonal constraints in tools like UPPAAL and KRONOS [19]. It was shown in [5] that this is incorrect: there are automata with diagonal constraints for which using Extra^+_{LU} will give a yes answer to the reachability problem, whereas the accepting state is not actually reachable in the automaton. This is because the extra valuations added during the computation enable guards which were originally not enabled in the automaton, leading to spurious executions. A non convex abstraction for diagonal constraints appears in [5], but the corresponding inclusion test is not known. The current algorithm for diagonal constraints proceeds by an abstraction refinement method [8].

Organization of the paper. Section 2 gives the preliminary definitions. In Section 3, we propose a simulation relation $\preccurlyeq^d_{LU}$ between zones and observe some of its properties. Section 4 gives an algorithm for $Z \not\preccurlyeq^d_{LU} Z'$ via reduction to an SMT formula. Section 5 shows that $Z \not\preccurlyeq^d_{LU} Z'$ is NP-hard by a reduction from 3-SAT. We report some experiments and conclude in Section 6. Missing proofs can be found in the extended version [12].

2 Preliminaries

Let $\mathbb{N}$ denote the set of natural numbers, $\mathbb{Z}$ the set of integers and $\mathbb{R}_{\geq 0}$ the set of non-negative reals. We denote the power set of a set S by $\mathcal{P}(S)$. A *clock* is a variable that ranges over $\mathbb{R}_{\geq 0}$. Fix a finite set of clocks X. A *valuation* v is a function which maps each clock $x \in X$ to a value in $\mathbb{R}_{\geq 0}$. Let $\Phi(X)$ denote the set of *clock constraints* ϕ formed using the following grammar: $\phi := x \sim c \mid x - y \sim c \mid \phi \wedge \phi$, where $x, y \in X$, $c \in \mathbb{N}$ and $\sim \in \{<, \leq, =, \geq, >\}$ Constraints of the form $x - y \sim c$ are called *diagonal constraints*. For a clock constraint ϕ, we write $v \models \phi$ if the constraint given by ϕ is satisfied by replacing each clock x in ϕ with $v(x)$. For $\delta \in \mathbb{R}_{\geq 0}$, we write $v + \delta$ for the valuation defined by $(v + \delta)(x) = v(x) + \delta$ for all clocks x. For a set R of clocks, we write $[R]v$ for the valuation obtained by setting each clock $x \in R$ to 0 and each $x \notin R$ to $v(x)$.

▶ **Definition 1** (Timed Automata). A *timed automaton* $\mathcal{A}$ is a tuple (Q, X, Δ, q_0, F) where Q is a finite set of states, X is a finite set of clocks, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is a set of accepting states and $\Delta \subseteq Q \times \Phi(X) \times \mathcal{P}(X) \times Q$ is the transition relation. Each transition in Δ is of the form (q, g, R, q') where $g \in \Phi(X)$ is called the *guard* of the transition and $R \subseteq X$ is the set of clocks that are said to be *reset* at the transition.

Timed automata with no diagonal constraints are called *diagonal-free*. The semantics of timed automata is described as a transition system over the space of its *configurations*. A configuration is a pair (q, v) where $q \in Q$ is a state and v is a valuation. There are two kinds of transitions. *Delay* transitions are given by $(q, v) \to^{\delta} (q, v + \delta)$ for each $\delta \in \mathbb{R}_{\geq 0}$, and *action* transitions are given by $(q, v) \to^{t} (q', v')$ for each transition $t \in \Delta$ of the form (q, g, R, q'), if $v \models g$ and $v' = [R]v$. The initial configuration is $(q_0, \mathbf{0})$ where $\mathbf{0}$ denotes the valuation mapping each clock to 0. Note that the above transition system is infinite. A *run* of a timed automaton is an alternating sequence of delay and action transitions starting from the initial configuration: $(q_0, \mathbf{0}) \to^{\delta_0}\to^{t_0} (q_1, v_1) \to^{\delta_1}\to^{t_1} \cdots (q_n, v_n)$. A run of the above form is said to be accepting if the last state $q_n \in F$. The *reachability problem* for timed automata is the following: given an automaton $\mathcal{A}$, decide if there exists an accepting run. This problem is known to be PSPACE-complete [1]. As the space of configurations is infinite, the main challenge in solving this problem involves computing a finite (and as small as possible) abstraction of the timed automaton semantics. In this section, we recall the reachability algorithm for the diagonal free case. For the rest of the section we fix a timed automaton $\mathcal{A}$.

Instead of working with configurations, standard solutions in timed automata analysis work with sets of valuations. The "successor" operation is naturally extended to the case of sets. For every transition t of $\mathcal{A}$ and every set of valuations W, we have a transition $\Rightarrow^t$ defined as follows: $(q, W) \Rightarrow^t (q', W')$ where $W' = \{v' \mid \exists v \in W,\ \exists \delta \in \mathbb{R}_{\geq 0} :\ (q, v) \rightarrow^t \rightarrow^\delta (q', v')\}$. Note that in the definition we have a $\rightarrow^\delta$ following the $\rightarrow^t$. This ensures that the $\Rightarrow$ successors (where $\Rightarrow\ = \bigcup_{t\in\Delta} \Rightarrow^t$) are closed under time successors. Moreover, the sets which occur during timed automata analysis using the $\Rightarrow$ relation have a special structure, and are called *zones*. A zone is a set of valuations which can be described using a conjunction of constraints of the form: $x \sim c$ or $x - y \sim c$ where $x, y \in X$ and $c \in \mathbb{N}$. Zones can be efficiently represented using Difference Bound Matrices (DBMs). To each automaton $\mathcal{A}$, we associate a transition system consisting of (state, zone) pairs: the *zone graph* $ZG(\mathcal{A})$ is a transition system whose nodes are of the form (q, Z) where q is a state of $\mathcal{A}$ and Z is a zone. The initial node is (q_0, Z_0) with $Z_0 = \{\mathbf{0} + \delta \mid \delta \geq 0\}$. Transitions are given by $\Rightarrow$.

▶ **Lemma 2.** *The zone graph $ZG(\mathcal{A})$ is sound and complete for reachability [9].*

Although the zone graph is a more succinct representation than the space of configurations, it could still be infinite. The reachability algorithm employs *simulation relations* between zones to obtain a finite abstraction of the zone graph that is sound and complete[1].

We start by defining this notion of simulations at the level of configurations. A *(time-abstract) simulation* between pairs of configurations of $\mathcal{A}$ is a reflexive and transitive relation $(q, v) \preccurlyeq (q', v')$ such that: $q = q'$; for every $(q, v) \rightarrow^\delta (q, v + \delta)$ there exists δ' such that $(q, v') \rightarrow^{\delta'} (q, v' + \delta')$ satisfying $(q, v + \delta) \preccurlyeq (q, v' + \delta')$; and if $(q, v) \rightarrow^t (q_1, v_1)$, then there exists $(q, v') \rightarrow^t (q_1, v'_1)$ satisfying $(q_1, v_1) \preccurlyeq (q_1, v'_1)$ for the same transition t. We say that (q, v) is simulated by (q', v'). We write $v \preccurlyeq v'$ if $(q, v) \preccurlyeq (q, v')$ for all states q. Simulations can be extended to relate zones in the natural way: we write $Z \preccurlyeq Z'$ if for all $v \in Z$ there exists $v' \in Z'$ such that $v \preccurlyeq v'$. A simulation relation $\preccurlyeq$ is said to be *finite* if there exists $N \in \mathbb{N}$ such that for all $n > N$ and every sequence of zones $\{Z_1, Z_2, \ldots, Z_n\}$, there exists $i < j \leq n$ such that $Z_j \preccurlyeq Z_i$.

Reachability algorithm. The input to the algorithm is a timed automaton $\mathcal{A}$. The algorithm maintains two lists, Passed and Waiting, and makes use of a finite simulation relation $\preccurlyeq$ between zones. The initial node (q_0, Z_0) is added to the Waiting list. Wlog. we assume that q_0 is not accepting. The algorithm repeatedly performs the following steps:

Step 1. If Waiting is empty, then return "$\mathcal{A}$ has no accepting run"; else pick (and remove) a node (q, Z) from Waiting.

Step 2. For each successor $(q, Z) \Rightarrow (q_1, Z_1)$ such that $Z_1 \neq \emptyset$ perform the following operations: if q_1 is accepting, return "$\mathcal{A}$ has an accepting run"; else check if there exists

[1] Existing reachability algorithms make use of what are known as abstraction operators [2, 16], which are based on simulation relations. Instead of abstractions, we choose to present the algorithm directly using simulations between zones.

a node (q_1, Z_1') in Passed or Waiting such that $Z_1 \preccurlyeq Z_1'$: if yes, ignore the node (q_1, Z_1), otherwise add (q_1, Z_1) to Waiting.

Step 3. Add (q, Z) to Passed and proceed to Step 1.

▶ **Theorem 3.** *The reachability algorithm terminates with a correct answer.*

The reachability algorithm relies on an operation $Z_1 \preccurlyeq Z_1'$, where $\preccurlyeq$ is some finite simulation relation as defined earlier. It has been shown that for the simulation relation $\preccurlyeq_{LU}$ of [2] which works for diagonal free automata, checking $Z \preccurlyeq_{LU} Z'$ can be done in time $O(|X|^2)$ [16]. Hence in diagonal free timed automata, this simulation test is as efficient as checking normal inclusion $Z \subseteq Z'$. The successor computation can also be implemented in $O(|X|^2)$ [20] using Difference Bound Matrices. These matrices can also be viewed as graphs. We recall this graph-based representation of zones and some of its properties.

▶ **Definition 4** (Distance graph)**.** A *distance graph* G has clocks as vertices, with an additional special vertex x_0 representing constant 0. Between every two vertices there is an edge with a *weight* of the form $(\triangleleft, c)$ where $c \in \mathbb{Z}$ and $\triangleleft \in \{\leq, <\}$ or $(\triangleleft, c) = (<, \infty)$. An edge $x \xrightarrow{\triangleleft c} y$ represents a constraint $y - x \triangleleft c$: or in words, the distance from x to y is bounded by c. We let $[\![G]\!]$ be the set of valuations of clock variables satisfying all the constraints given by the edges of G with the restriction that the value of x_0 is 0.

We will sometimes write 0 instead of x_0 for clarity. An arithmetic over the weights $(\triangleleft, c)$ can be defined as follows [3].

Equality $(\triangleleft_1, c_1) = (\triangleleft_2, c_2)$ if $c_1 = c_2$ and $\triangleleft_1 = \triangleleft_2$.
Addition $(\triangleleft_1, c_1) + (\triangleleft_2, c_2) = (\triangleleft, c_1 + c_2)$ where $\triangleleft = <$ iff either $\triangleleft_1$ or $\triangleleft_2$ is $<$.
Total order $(\triangleleft_1, c_1) < (\triangleleft_2, c_2)$ if either $c_1 < c_2$ or ($c_1 = c_2$ and $\triangleleft_1 = <$ and $\triangleleft_2 = \leq$).

This arithmetic lets us talk about the weight of a path as the sum of the weights of its edges.

A cycle in a distance graph G is said to be *negative* if the sum of the weights of its edges is at most $(<, 0)$. A distance graph is in *canonical form* if there are no negative cycles and the weight of the edge from x to y is the lower bound of the weights of paths from x to y. Given a distance graph, its canonical form can be computed by using an all-pairs shortest paths algorithm like Floyd-Warshall's [3] in time $\mathcal{O}((|X| + 1)^3)$ where $|X|$ is the number of clocks. Note that the number of vertices in the distance graph is $|X| + 1$. A folklore result is that: a distance graph G has no negative cycles iff $[\![G]\!] \neq \emptyset$. Given two distance graphs G_1, G_2 (not necessarily in their canonical form), we define $\min(G_1, G_2)$ to be the distance graph obtained by setting for each $x \to y$ the minimum of the corresponding weights in G_1 and G_2. For two distance graphs G_1 and G_2, we have $[\![\min(G_1, G_2)]\!] = [\![G_1]\!] \cap [\![G_2]\!]$.

A simulation relation for timed automata with diagonal constraints was proposed in [5], but it has not been used in the reachability algorithm since no algorithm for the zone simulation test was known.

3 A new simulation relation in the presence of diagonal constraints

In this section, we introduce a new simulation relation $\preccurlyeq^d_{LU}$ which extends the $\preccurlyeq_{LU}$ simulation of [2]. For this, we first assume that all guards in timed automata are rewritten in the form $x - y \triangleleft c$ or $x \triangleleft c$, where $c \in \mathbb{Z}$ and $\triangleleft \in \{<, \leq\}$. We will also assume that X is a set of clocks including the 0 clock.

▶ **Definition 5** (LU-bounds)**.** An *LU bounds function* is a pair of functions $L : X \times X \mapsto \mathbb{Z} \cup \{\infty\}$ and $U : X \times X \mapsto \mathbb{Z} \cup \{-\infty\}$ mapping each clock difference $x - y$ to a constant or ∞ or $-\infty$ such that the conditions below are satisfied (we write $L(x - y), U(x - y)$ for $L(x, y)$ and $U(x, y)$ respectively):

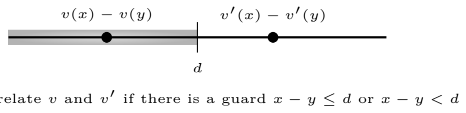
$v(x) - v(y)$
$v'(x) - v'(y)$
d
Do not relate v and v' if there is a guard $x - y \leq d$ or $x - y < d$

Figure 1 Black dots illustrate the values of $v(x) - v(y)$ and $v'(x) - v'(y)$. The value $v(x) - v(y)$ satisfies the guard $x - y \triangleleft d$ but $v'(x) - v'(y)$ does not satisfy the same guard.

- either $L(x-y) = \infty$ and $U(x-y) = -\infty$, or $L(x-y) \leq U(x-y)$ for all distinct pairs of clocks $x, y \in X$,
- $L(x-0) = 0$ and $U(0-x) = 0$ for all non zero clocks $x \in X$

The L stands for *lower* and U stands for *upper*. Intuitively, each LU-bounds function corresponds to a set of guards given by $x - y \triangleleft c$ with $L(x-y) \leq c \leq U(x-y)$. We will now define a simulation relation $\preccurlyeq^d_{LU}$ between valuations parameterized by LU-bounds. The idea is to give a relation $v \preccurlyeq^d_{LU} v'$ such that v' satisfies all guards compatible with the parameter LU that v satisfies. To achieve this, the situation as illustrated in Figure 1 needs to be avoided. This is formalized by the following definition and the subsequent lemma.

▶ **Definition 6** (LU-preorder $\preccurlyeq^d_{LU}$)**.** Let LU be a bounds function. A valuation v' simulates a valuation v with respect to LU, written as $v \preccurlyeq^d_{LU} v'$, if for every pair of distinct clocks $x, y \in X$ the following hold:
- $v'(x) - v'(y) < L(x-y)$ if $v(x) - v(y) < L(x-y)$
- $v'(x) - v'(y) \leq v(x) - v(y)$ if $L(x-y) \leq v(x) - v(y) \leq U(x-y)$

For a valuation v, we write $\langle v \rangle_{LU}$ for the set of all v' such that $v \preccurlyeq^d_{LU} v'$.

▶ **Lemma 7.** *Let x, y be distinct clocks in X, and $x - y \triangleleft c$ with $c \in \mathbb{Z}$ be a guard. Let LU be a bounds function such that $L(x-y) \leq c \leq U(x-y)$. Then, for every pair of valuations v, v' such that $v \preccurlyeq^d_{LU} v'$, if valuation $v \models x - y \triangleleft c$ then $v' \models x - y \triangleleft c$.*

The next lemma shows that time delay preserves $\preccurlyeq^d_{LU}$ from two valuations v and v' with $v \preccurlyeq^d_{LU} v'$. In fact, it is strong in the sense that if we delay δ from v, then the same delay from v' satisfies the LU preorder conditions. The proof of this lemma uses the fact that $L(x-0) = 0$ and $U(0-x) = 0$ for all non-zero clocks x.

▶ **Lemma 8.** *Let LU be a bounds function. For every pair of valuations v and v', if $v \preccurlyeq^d_{LU} v'$, then $v + \delta \preccurlyeq^d_{LU} v' + \delta$ for all $\delta \geq 0$.*

The next lemma shows that resets preserve $\preccurlyeq^d_{LU}$ under certain conditions on LU.

▶ **Lemma 9.** *Let LU be a bounds function satisfying $U(x-0) \geq U(x-y)$ for all $y \in X$ and $L(0-y) \leq L(x-y)$ for all $x \in X$. Then, $v \preccurlyeq^d_{LU} v'$ implies $[R]v \preccurlyeq^d_{LU} [R]v'$ for every $R \subseteq X$.*

The LU preorder can be extended to configurations: $(q, v) \preccurlyeq^d_{LU} (q, v')$ if $v \preccurlyeq^d_{LU} v'$. The above three lemmas give the necessary ingredients to generate an LU bounds function from a timed automaton $\mathcal{A}$ such that the associated LU preorder is a simulation on its space of configurations.

Let $\mathcal{G}$ be a set of constraints. We construct a new set $\overline{\mathcal{G}}$ from $\mathcal{G}$ in the following way:
- Add all the constraints of $\mathcal{G}$ to $\overline{\mathcal{G}}$
- For each clock $x \in X$, add the constraints $x \leq 0$ and $-x \leq 0$ to $\overline{\mathcal{G}}$
- For each constraint $x - y \triangleleft c \in \mathcal{G}$, add the constraints $x \triangleleft c$ and $-y \triangleleft c$ to $\overline{\mathcal{G}}$
- Remove all constraints of the form $x \triangleleft c_1$ where $c_1 \in \mathbb{R}_{<0}$ and constraints of the form $-x \triangleleft c_2$ where $c_2 \in \mathbb{R}_{>0}$ from $\overline{\mathcal{G}}$.

We define an LU-bounds function on $\overline{\mathcal{G}}$ in the natural way: for each pair of clocks $x, y \in X$, we set $L(x-y) = \min\{c \mid x - y \triangleleft c \in \overline{\mathcal{G}}\}$ and $U(x-y) = \max\{c \mid x - y \triangleleft c \in \overline{\mathcal{G}}\}$. If there are no guards of the form $x - y \triangleleft c$ in $\overline{\mathcal{G}}$, then we set $L(x-y)$ to be ∞ and $U(x-y)$ to be $-\infty$. Note that since $\overline{\mathcal{G}}$ contains the constraints $x \leq 0$ and has no constraints $x \triangleleft c$ where $c \in \mathbb{R}_{<0}$, $L(x-0) = 0$ for all $x \in X$. Similarly, $U(0-x) = 0$ for all $x \in X$. For a timed automaton $\mathcal{A}$, let $\mathcal{G}_{\mathcal{A}}$ be the set of guards present in $\mathcal{A}$. The *LU-bounds of $\mathcal{A}$* is the LU-bounds function defined on $\overline{\mathcal{G}_{\mathcal{A}}}$. The next theorem follows from Lemmas 7, 8 and 9.

▶ **Theorem 10.** *For every timed automaton $\mathcal{A}$, the relation $\preccurlyeq^d_{LU}$ obtained from the LU-bounds of $\mathcal{A}$ is a simulation relation on its configurations.*

We use this simulation relation extended to zones in the reachability algorithm, as described in Page 4. To do so, we need to give an algorithm for the simulation test $Z \preccurlyeq^d_{LU} Z'$, and show that $\preccurlyeq^d_{LU}$ is finite. Correctness and termination follow from Theorem 3. We first describe the simulation test, and then prove finiteness. Observe that $Z \not\preccurlyeq^d_{LU} Z'$ iff there exists $v \in Z$ such that $\langle v \rangle_{LU} \cap Z' = \emptyset$. We give a distance graph representation for $\langle v \rangle_{LU}$.

▶ **Definition 11** (Distance graph for $\langle v \rangle_{LU}$)**.** Given a valuation v and an LU bounds function, we construct distance graph $G^{LU}_{\langle v \rangle}$ as follows. For every pair of distinct clocks $x, y \in X$, add the edges:

- $y \to x$ with weight $(<, L(x-y))$, if $v(x) - v(y) < L(x-y)$,
- $y \to x$ with weight $(\leq, v(x) - v(y))$, if $L(x-y) \leq v(x) - v(y) \leq U(x-y)$.

Using Definition 6 we can show that $[\![G^{LU}_{\langle v \rangle}]\!]$ equals $\langle v \rangle_{LU}$. The properties of distance graphs as described in Page 5 then lead to the following theorem.

▶ **Theorem 12.** *Let Z, Z' be zones such that Z' is non-empty, and let LU be a bounds function. Let $G_{Z'}$ be the canonical distance graph of Z'. Then, $Z \not\preccurlyeq^d_{LU} Z'$ iff there is a valuation $v \in Z$ and a negative cycle in $\min(G^{LU}_{\langle v \rangle}, G_{Z'})$ in which no two consecutive edges are from $G_{Z'}$.*

A witness to the fact that $Z \not\preccurlyeq^d_{LU} Z'$ is therefore a $v \in Z$ and a negative cycle of a certain shape given by Theorem 12. As explained in Section 4, existence of such a witness can be encoded as satisfiability of a formula in linear arithmetic. This gives an NP procedure. A satisfying assignment to the formula reveals a valuation $v \in Z$ and a corresponding negative cycle across $G^{LU}_{\langle v \rangle}$ and $G_{Z'}$. Although there is no fixed bound on the length of this negative cycle (contrary to the diagonal free case), note that each $y \to x$ edge from $G^{LU}_{\langle v \rangle}$ in the negative cycle needs to have finite $U(x-y)$ and $L(x-y)$ constants **(apart from $x \to 0$ edges)**. If for an automaton, many pairs of clocks have no diagonal constraints (which we believe occurs often in practice) then this simulation test would need to enumerate only a small number of cycles.

The final step is to show that $\preccurlyeq^d_{LU}$ is finite. We make use of a notation: we write $\downarrow Z$ to be the set of valuations u such that $u \preccurlyeq^d_{LU} v$ for some $v \in Z$. Note that $Z \not\preccurlyeq^d_{LU} Z'$ implies $\downarrow Z \neq \downarrow Z'$.

▶ **Theorem 13.** *The simulation relation $\preccurlyeq^d_{LU}$ is finite for every LU bounds function.*

Proof. We will first show that for any zone Z, $\downarrow Z$ is a union of *d-regions* (parameterized by LU) which are defined below. We will subsequently show that there are only finitely many d-regions. The observation that $Z \not\preccurlyeq^d_{LU} Z'$ implies $\downarrow Z \neq \downarrow Z'$ then proves the theorem.

Given a valuation v and LU-bounds function, we define the following relations over pairs of clocks:

- $y \xrightarrow{1} x$ if $v(x) - v(y) < L(x-y)$
- $y \xrightarrow{2} x$ if $L(x-y) \leq v(x) - v(y) \leq U(x-y)$

A d-region R is a set of valuations that satisfies the following:

- all valuations in R have the same $\xrightarrow{1}$ and $\xrightarrow{2}$ relations.
- for every subset $S = \{y_1 \xrightarrow{2} x_1, y_2 \xrightarrow{2} x_2, \dots, y_k \xrightarrow{2} x_k\}$ of ordered pairs of clocks, every valuation in R satisfies one of the following constraints: either $\left(\sum_{i=1}^{i=k} x_i - y_i = c\right)$ or $c - 1 < \left(\sum_{i=1}^{i=k} x_i - y_i\right) < c$ for an integer c satisfying $\sum_{i=1}^{i=k} L(x_i - y_i) \leq c \leq \sum_{i=1}^{i=k} U(x_i - y_i)$.

We will now show that if a d-region R intersects $\downarrow Z$ then $R \subseteq \downarrow Z$. Let $v \in R$ be such that $v \in \downarrow Z$. Let v' be another valuation in R. Suppose $v' \notin \downarrow Z$. Then $\langle v' \rangle_{LU} \cap Z = \emptyset$. That is, $\min(G^{LU}_{\langle v' \rangle}, G_Z)$ has a negative cycle; let us call it $N_{v'}$. Let N_v be the cycle $N_{v'}$ with the edges coming from $G^{LU}_{\langle v' \rangle}$ replaced with the same edges from $G^{LU}_{\langle v \rangle}$. We want to show that N_v is negative. Since v and v' come from the same region R, we have:

- The weight of a type 1 edge $y_i \xrightarrow{1} x_i$ is $(<, L(x_i - y_i))$ in both N_v and $N_{v'}$. Let $(<, S_1)$ be the sum of the weights of the type 1 edges. This sum is the same in N_v and $N_{v'}$.
- We let $(\leq, S_2) = (\leq, \sum_i v(x_i) - v(y_i))$ and $(\leq, S_2') = (\leq, \sum_i v'(x_i) - v'(y_i))$ be the sum of the weights of type 2 edges $y_i \xrightarrow{2} x_i$ in N_v and $N_{v'}$ respectively. Then, for some integer c, either $S_2 = S_2' = c$ or $c - 1 < S_2 < c$ and $c - 1 < S_2' < c$.

Also the edges coming from G_Z have the same weight in N_v and $N_{v'}$. Call $(\triangleleft_3, S_3)$ the sum of the weights of the edges coming from G_Z. Finally, let $(\triangleleft, S = S_1 + S_2 + S_3)$ and $(\triangleleft, S' = S_1 + S_2' + S_3)$ be the weights of N_v and $N_{v'}$ respectively. Since $N_{v'}$ is negative, $(\triangleleft, S')$ is at most $(<, 0)$. Now, S_1 and S_3 are integers, and using the relation between S_2 and S_2', we deduce that N_v is also negative. This entails $\langle v \rangle_{LU} \cap Z = \emptyset$, and contradicts the assumption $v \in \downarrow Z$. We get $R \subseteq \downarrow Z$, thereby showing that each $\downarrow Z$ is a union of d-regions.

Each d-region depends only on the $\xrightarrow{1}$ and $\xrightarrow{2}$ relations and the values of c for each subset S of $\xrightarrow{2}$ edges. Since number of clocks is finite, the number of possible relations $\xrightarrow{1}$ and $\xrightarrow{2}$ is finite. For each such relations, the possible values for the constants c is finite. Thus there are only finitely many d-regions. ◀

4 Algorithm for $Z \not\preccurlyeq^d_{LU} Z'$

Theorem 12 gives a witness for the fact that $Z \not\preccurlyeq^d_{LU} Z'$. In this section, we encode the existence of this witness as an SMT formula over linear arithmetic. For clarity of exposition, we will also restrict to timed automata having no strict constraints as guards, that is, every guard is of the form $x - y \leq c$ or $x \leq c$. This would in particular imply that in the zones obtained during the forward analysis, there will be no strict constraints.

▶ **Definition 14** (Satisfiability modulo Linear Arithmetic)**.** Let Prop be a set of propositional variables, and Vars a set of variables ranging over reals. An atomic term is a constraint of the form $c_1x_1 + c_2x_1 + \cdots + c_kx_k \sim d$ where $c_1, \dots, c_n, d \in \mathbb{Z}$ and $x_1, x_2, \dots, x_k \in$ Vars and $\sim \in \{\leq, <, =, >, \geq\}$. A formula in *linear arithmetic* is a boolean combination of propositional variables and atomic terms. Formula ϕ is satisfiable if there exists an assignment of boolean values to propositions in Prop, and real values to variables in Vars such that replacing every occurrence of the variables and propositions by the assigment evaluates ϕ to true.

The next lemma follows from [17].

▶ **Lemma 15.** *Satisfiability of a formula in linear arithmetic is in* NP.

Fix two zones Z, Z' and a bounds function LU. Zones Z and Z' are given by their canonical distance graphs G_Z and $G_{Z'}$. We write c_{yx} for the weight of the edge $y \to x$ in G_Z and c'_{yx} for the weight of $y \to x$ in $G_{Z'}$. Further we assume that the set of clocks is $\{\mathbf{0}, \mathbf{1}, \dots, \mathbf{n}\}$. The final formula will be obtained by constructing suitable intermediate subformulas as explained below:

Step 1. Guess a $v \in Z$.
Step 2. Guess a subset of edges $y \to x$ which forms a cycle (or a disjoint union of cycles).
Step 3. Guess a colour for each edge $y \to x$ in the cycle: red or blue. No two consecutive edges in the cycle can both be red. Red edges correspond to edges from $G_{Z'}$. Blue edges correspond to edges from $G^{LU}_{\langle v \rangle}$.
Step 4. Assign weights to each edge $y \to x$: if it is coloured red, the weight is c'_{yx} (edge weight of $G_{Z'}$). If the edge $y \to x$ is blue, assign weight according to the following cases:
- $w_{yx} = (<, L(x-y))$ if $v(x) - v(y) < L(x-y)$
- $w_{yx} = (\leq, v(x) - v(y))$ if $L(x-y) \leq v(x) - v(y) \leq U(x-y)$

Add up the weights of all the edges (the comparison $<$ or $\leq$ component of the weight can be maintained using a boolean). If there are no strict edges (that is with weight $<$) in the chosen cycle, check if the sum is < 0. Else, check if the sum is ≤ 0.

Formula for Step 1. We first guess a valuation $v \in Z$. We use real variables $v_0, v_1, \dots, v_n$ to denote a valuation. These variables should satisfy the constraints given by Z:

$$v_0 = 0 \quad \text{and} \quad \bigwedge_{x,y \in \{0,\dots,n\}} v_x - v_y \leq c_{yx} \tag{1}$$

Call the above formula $\Phi_1(\bar{v})$ where $\bar{v} = (v_0, \dots, v_n)$. A satisfying assignment to Φ_1 corresponds to a valuation in Z.

Formula for Step 2. We now need to guess a set of edges of the form $y \to x$ which forms a cycle, or a disjoint union of simple cycles. We will also ensure that no vertex appears in more than one cycle. We will use boolean variables e_{ij} for $i, j \in \{0, \dots, n\}$ and $i \neq j$.

The cycle must be non-empty.

$$\bigvee_{0 \leq i,j \leq n, j \neq i} e_{ij} \tag{2}$$

If we pick an incoming edge to a clock, then we need to pick an outgoing edge.

$$\bigwedge_{0 \leq i \leq n} \Big(\bigvee_{0 \leq j \leq n, j \neq i} e_{ji} \Big) \implies \Big(\bigvee_{0 \leq j \leq n, j \neq i} e_{ij} \Big) \tag{3}$$

We do not pick more than one outgoing or incoming edges for each clock.

$$\bigwedge_{0 \leq i \leq n} \quad \bigwedge_{\substack{0 \leq j,k \leq n \\ j \neq k, i \neq j, i \neq k}} \neg(e_{ij} \wedge e_{ik}) \ \wedge \ \neg(e_{ji} \wedge e_{ki}) \tag{4}$$

Conjunction of (2, 3, 4) gives a formula $\Phi_2(\bar{e})$ over variables $\bar{e} = \{e_{01}, \dots, e_{nn-1}\}$.

▶ **Lemma 16.** *Let $\sigma_2 : \bar{e} \mapsto \{\text{true}, \text{false}\}$ be an assignment which satisfies Φ_2. Then the set of edges $\{x \to y\}$ such that $\sigma_2(e_{xy})$ is true forms a vertex-disjoint union of cycles.*

Formula for Step 3. To colour the edges of the cycle formed by e_{ij}, we will use boolean variables r_i for $0 \leq i \leq n$ to color the source of the red edges. Once the red edges are determined, the blue edges are also uniquely determined. Only edges chosen by $\bar{e}$ are coloured red, and no two consecutive edges can be coloured red.

$$\bigwedge_{0 \leq i \leq n} \Big(r_i \implies \bigvee_{0 \leq j \leq n} e_{ij} \wedge \neg r_j \Big) \tag{5}$$

Then, red edges are edges with corresponding source i satisfying r_i. So for all $i, j \in \{0, \ldots, n\}$ with $i \neq j$ we introduce the macro $\text{red}_{ij} := e_{ij} \wedge r_i$. Blue edges are those that have been chosen for the cycle and have not been coloured red: $\text{blue}_{ij} := e_{ij} \wedge \neg r_i$. Each blue edge should satisfy one of the two conditions mentioned in Definition 11.

$$\bigwedge_{i,j \in \{0,\ldots,n\}, i \neq j} \text{blue}_{ij} \implies v_j - v_i \leq U(j-i) \tag{6}$$

Conjunction of (5) and (6) gives formula Φ_3.

▶ **Lemma 17.** *Let σ_3 be an assignment to variables $\bar{v}$, $\bar{e}$ and $\bar{r}$. Suppose σ_3 is a satisfying assignment for $\Phi_1 \wedge \Phi_2 \wedge \Phi_3$. Then, the set of edges with $\sigma_3(e_{ij})$ being true forms a collection of vertex disjoint cycles using edges from $G_{Z'}$ or from $G^{LU}_{\langle v \rangle}$ for some $v \in Z$.*

Formula for Step 4. The last step is to add up weights of the red and blue edges. We make use of real-valued variables w_i for each source i of an edge. We associate weights of red and blue edges.

$$\begin{aligned}\bigwedge_{i,j \in \{0,\ldots,n\}, i \neq j} (\text{red}_{ij} \implies w_i = c'_{ij}) \wedge ((\text{blue}_{ij} \wedge \mathit{condition}_1) \implies w_i = L(j-i)) \\ \wedge((\text{blue}_{ij} \wedge \mathit{condition}_2) \implies w_i = v_j - v_i)\end{aligned} \tag{7}$$

where, $\mathit{condition}_1 := v_j - v_i < L(j-i)$ and $\mathit{condition}_2 := L(j-i) \leq v_j - v_i \leq U(j-i)$.

Uncoloured edges take weight 0,

$$\bigwedge_{0 \leq i \leq n} \Big(\bigwedge_{0 \leq j \leq n, j \neq i} \neg e_{ij} \Big) \implies (w_i = 0) \tag{8}$$

A boolean variable strict is true if one of the blue edges has a weight of the form $(<, c)$.

$$\text{strict} \iff \bigvee_{i,j \in \{0,\ldots,n\}, i \neq j} \text{blue}_{ij} \wedge \mathit{condition}_1 \tag{9}$$

The final formula checks if the sum of the weights is at most $(<, 0)$.

$$((\textstyle\sum_{0 \leq i \leq n} w_i) < 0) \vee [\text{ strict } \wedge ((\textstyle\sum_{0 \leq i \leq n} w_i) = 0)] \tag{10}$$

Conjunction of (7), (8) and (10) gives formula Φ_4. The final formula is $\Phi = \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge \Phi_4$.

▶ **Theorem 18.** *Formula Φ as constructed above is satisfiable iff $Z \not\preccurlyeq^d_{LU} Z'$.*

Note that there are $\mathcal{O}(n+1)$ real variables v_i, w_i, and $\mathcal{O}((n+1)^2)$ booleans e_{ij}, r_i. Given the representations of Z, Z' and the LU bounds, the entire formula Φ can be computed in $\mathcal{O}((n+1)^3)$, with formula (4) taking the maximum time. This gives an NP procedure for $Z \not\preccurlyeq^d_{LU} Z'$ (c.f. Lemma 15).

5 Checking $Z \not\preccurlyeq^d_{LU} Z'$ is NP-hard

We will consider a special kind of LU bounds, which already turns out to be hard. We say that an LU bounds is *symmetric* if $L(x-y) = -U(y-x)$ for all distinct pairs of clocks x, y. This symmetry gives rise to some nice properties which we will use to show hardness.

▶ **Lemma 19.** *Let v, v' be valuations and LU a symmetric bounds function. Then, $v \preccurlyeq^d_{LU} v'$ iff for all distinct pairs of clocks x, y (denoting $a = v(x) - v(y)$ and $a' = v'(x) - v'(y)$):*
- *either both a' and a are $< L(x-y)$*
- *or $L(x-y) \leq a' = a \leq U(x-y)$*
- *or both a' and a are $> U(x-y)$.*

Proof. Since $L(x-y) = -U(y-x)$, we deduce that $L(x-y) \leq a \leq U(x-y)$ iff $L(y-x) \leq -a \leq U(y-x)$. The rest follows by applying Definition 6 on a, a' and $-a, -a'$. ◀

Thanks to the above lemma, when LU is symmetric: $v \preccurlyeq^d_{LU} v'$ iff $v' \preccurlyeq^d_{LU} v$, and hence $\preccurlyeq^d_{LU}$ an equivalence over valuations. To make this explicit, we will write $v \simeq_{LU} v'$ instead of $v \preccurlyeq^d_{LU} v'$, and $[v]_{LU}$ instead of $\langle v \rangle_{LU}$ for symmetric LU. With this definition, for symmetric LU, we get $Z \not\preccurlyeq^d_{LU} Z'$ iff there exists $v \in Z$ such that for all $v' \simeq_{LU} v$, we have $v' \notin Z'$. Throughout this section, we will fix a symmetric LU bounds function.

The second condition in Lemma 19 constrains the difference between certain pairs of clocks to a constant value for all valuations in an equivalence class of $\simeq_{LU}$. We formalize this notion. Let v be a valuation. Two clocks x and y are said to be *tight in v* if $L(x-y) \leq v(x) - v(y) \leq U(x-y)$. We denote this by $x \multimapboth y$ (can be read as x and y are tied to each other). Notice that $\multimapboth$ is symmetric. Let $\multimapboth^*$ (can again be read as the tight relation) denote the reflexive and transitive closure of $\multimapboth$. The $\multimapboth^*$ relation is an equivalence over clocks. For every $v' \in [v]_{LU}$, Lemma 19 gives: $v'(x) - v'(y) = v(x) - v(y)$ when $x \multimapboth^* y$ and $v'(x) - v'(y) < L(x-y)$ when $x \not\multimapboth^* y$ and $v(x) - v(y) < L(x-y)$. Notice also that the $\multimapboth^*$ equivalence classes are identical for v' and v when $v' \simeq_{LU} v$.

Next, we make an observation about zones which do not have strict constraints (like $x - y < c$). We say that a zone Z is *topologically closed* if every edge $y \to x$ in the canonical distance graph of Z has weight of the form $(\leq, c)$ with $c \in \mathbb{Z}$, or $(<, \infty)$. A valuation v mapping each x to an integer is said to be an *integral valuation*. The next proposition says that for certain topologically closed zones Z and Z', if $Z \not\preccurlyeq^d_{LU} Z'$ then there is an integral valuation as a witness to this non-simulation. The proof of this proposition makes use of a non-trivial observation on zones. We refer the reader to [12] for more details.

▶ **Proposition 20.** *Let Z be a topologically closed zone s.t. the $\multimapboth^*$ equivalence classes of every valuation in Z are the same. Let LU be a symmetric bounds function. Let Z' be a zone with $Z \not\preccurlyeq^d_{LU} Z'$. Then, there exists an integral valuation $u \in Z$ such that $[u]_{LU} \cap Z'$ is empty.*

We now have the necessary ingredients to give the proof of NP-hardness. Consider the decision problem which takes as inputs two zones Z, Z' and outputs whether $Z \not\preccurlyeq^d_{LU} Z'$. We will give a polynomial time reduction from 3-SAT to this decision problem, showing that it is NP-hard.

Notation. Let Var be a finite set of propositional variables. A *literal* is either a variable p or its negation $\neg p$, and a 3-*clause* is a disjunction of three literals $(l_1 \vee l_2 \vee l_3)$. A 3-CNF formula is a conjunction of 3-clauses. For a literal l, we write $\text{Var}(l)$ for the variable corresponding to l. For a 3-CNF formula ϕ, we write $\text{Var}(\phi)$ for the variables present in ϕ. An *assignment* to a 3-CNF formula ϕ is a function from $\text{Var}(\phi)$ to $\{\text{true}, \text{false}\}$. For a clause C and an assignment

σ, we write $\sigma \models C$ if substituting $\sigma(p)$ for each variable p occurring in C evaluates the clause to true. For a formula ϕ and an assignment σ, we write $\sigma \models \phi$ if all clauses of ϕ evaluate to true under σ. A formula ϕ is said to be *satisfiable* if there exists an assignment such that $\sigma \models \phi$. For the rest of the section, fix a 3-CNF formula $\varphi := C_1 \wedge C_2 \wedge \cdots \wedge C_N$. Let $\text{Clauses}(\varphi)$ be the set $\{C_i \mid i \in \{1, \ldots, N\}\}$.

We start with the idea for the reduction. We know that φ is satisfiable iff there *exists* an assignment σ such that *for all* $C \in \text{Clauses}(\varphi) : \sigma \models C$. Correspondingly, we know that $Z \not\preccurlyeq^d_{LU} Z'$ iff there *exists* a $v \in Z$ such that *for all* $v' \simeq_{LU} v : v' \notin Z'$. Given φ, we want to construct two topologically closed zones Z, Z' such that φ is satisfiable iff $Z \not\preccurlyeq^d_{LU} Z'$. We want the (potential) $v \in Z$ for which every $v' \simeq_{LU} v$ satisfies $v' \notin Z'$ to encode the (potential) satisfying assignment for φ. In essence: valuations in Z should encode assignments, the equivalent valuations v' should encode clauses and the fact that $v' \notin Z'$ should correspond to the chosen clause being true. We now proceed with the details of the construction. Figure 2 illustrates the construction on an example. For each literal l^j_i of φ, we add three clocks x^j_i, y^j_i, z^j_i. There are $N + 1$ additional clocks $r_0, r_1, \ldots, r_N$, where r_0 is assumed to be the special 0 clock. We will assume that $L(x - y) = -M$, $U(x - y) = M$, $L(0 - x) = -M$, $U(x - 0) = M$ for all non-zero clocks x, y and an arbitrary constant $M > 3$. This gives a symmetric LU bounds function.

Construction of Z. Zone Z is described by three sets of constraints. The first set of constraints are between clocks of each literal. For every $i \in \{1, \ldots, N\}$ and $j \in \{1, 2, 3\}$:

$$y^j_i - x^j_i \geq 1 \quad \text{and} \quad z^j_i - y^j_i \geq 1 \quad \text{and} \quad z^j_i - x^j_i = 3 \tag{11}$$

The second set of constraints relates the distance between clocks of different literals. In addition, we use the r_i clocks as separators between clauses. For $i \in \{1, \ldots, N\}$:

$$x^1_i - r_{i-1} = 2M - 3 \text{ and } x^{j+1}_i - z^j_i = 2M - 3 \text{ for } j \in \{1, 2\} \text{ and } r_i - z^3_i = 2M \tag{12}$$

Constraints (11) and (12) ensure that for every valuation in Z we have the following order of clocks for each $i \in \{1, \ldots, N\}$:

$$r_{i-1} \;<\; x^1_i < y^1_i < z^1_i \;<\; x^2_i < y^2_i < z^2_i \;<\; x^3_i < y^3_i < z^3_i \;<\; r_i \tag{13}$$

In every valuation of Z, we have $x^j_i \multimap y^j_i \multimap z^j_i$ for every literal l^j_i. This is because we have assumed that $M > 3$ and we have restricted the gaps (absolute value of the differences) between x^j_i, y^j_i and y^j_i, z^j_i to be in the interval $[1, 2]$ (c.f. (11)). We do not want any other pair of clocks that are consecutive according to the above ordering to be tight. Hence we choose the rest of the gaps to be strictly more than M (c.f. (12)). Our choice of constraints ensures that each valuation in Z gives a $\multimap^*$ division where $\{x^j_i, y^j_i, z^j_i\}$ forms an equivalence class (let us call it a block), and each $\{r_i\}$ is an equivalence class. Note that for each $v \in Z$, we also have $v(r_i) - v(r_{i-1}) = 8M$ for $i \in \{1, \ldots, N\}$. We will next enforce that literals in φ involving the same variable have the same $y - x$ and $z - y$ values for their corresponding clocks. Without loss of generality, we assume that the three literals corresponding to the same clause have different variables. Therefore this condition is relevant for literals in different clauses, but with the same variable. For every l^j_i and $l^{j'}_{i'}$ such that $\text{Var}(l^j_i) = \text{Var}(l^{j'}_{i'})$ and $i' > i$:

$$y^{j'}_{i'} - y^j_i = (i' - i) \cdot 8M + (j' - j) \cdot 2M \tag{14}$$

Note that from (11) and (12) we can infer that the values of $v(x^{j'}_{i'}) - v(x^j_i)$ and $v(z^{j'}_{i'}) - v(z^j_i)$ are already equal to the right hand side of the above equation, as the x and z clocks are "fixed" and y is "flexible". Constraint (14) then ensures that $v(y^j_i) - v(x^j_i) = v(y^{j'}_{i'}) - v(x^{j'}_{i'})$ and $v(z^j_i) - v(y^j_i) = v(z^{j'}_{i'}) - v(y^{j'}_{i'})$ whenever l^j_i and $l^{j'}_{i'}$ with $i' > i$, have the same variable.

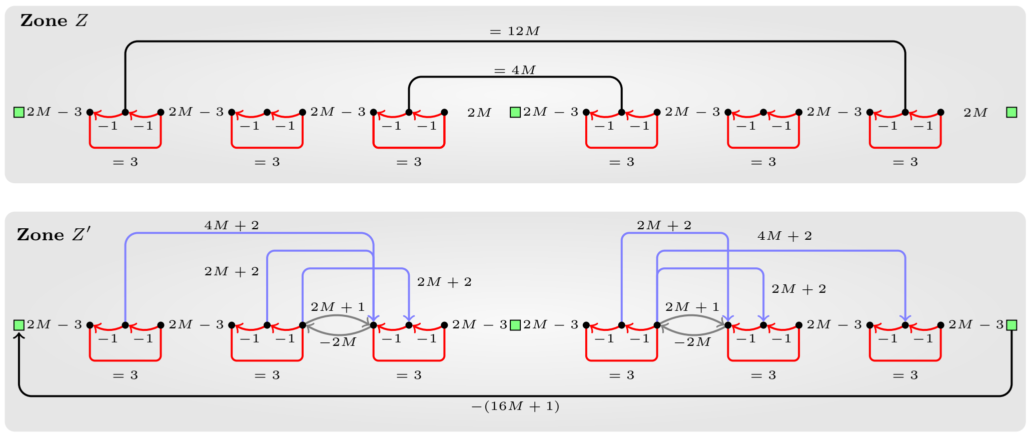

Figure 2 Illustration of the zone Z and Z' for the formula $(p_1 \vee p_2 \vee \neg p_3) \wedge (p_3 \vee \neg p_4 \vee \neg p_1)$. The separator clocks r_0, r_1, r_2 are shown by the green boxes (leftmost box is r_0, middle one is r_1 and the rightmost is r_2). The intermediate literal clocks are shown by the black dots: between r_0 and r_1 are $x_1^1, y_1^1, z_1^1, x_1^2, y_1^2, z_1^2, x_1^3, y_1^3, z_1^3$ in the same sequence. Similarly between r_1 and r_2 are the clocks $x_2^1, \dots, z_2^3$. An edge of the form $x \xrightarrow{c} y$ simply denotes the constraint $y - x \le c$, whereas edges $x \xrightarrow{=c} y$ mean that $y - x = c$. When we write c between two consecutive clocks, we mean that the difference between them equals c.

Encoding of assignments. By construction of Z, in every valuation $v \in Z$, we have $v(r_i)$, $v(x_i^j)$ and $v(z_i^j)$ to be fixed integers. The value of $v(y_i^j)$ can vary between $v(x_i^j)+1$ and $v(x_i^j)+2$. When this value is in the extremes, either 1 or 2, we get an integral valuation. We encode assignments by such integral valuations. An integral valuation v encodes the assignment σ_v given by: $\sigma_v(\mathrm{Var}(l_i^j)) = \text{true}$ if $v(y_i^j) - v(x_i^j) = 1$ and $\sigma_v(\mathrm{Var}(l_i^j)) = \text{false}$ if $v(y_i^j) - v(x_i^j) = 2$. By (14), the above assignment is well defined. Moreover, the zone Z contains an integral valuation for every possible assignment.

An assignment σ satisfies φ if every clause evaluates to true under σ. From a valuation v encoding this assignment σ, we need a mechanism to check whether each clause is true. This is where we will use the gaps which are not tight (that is the ones $> M$). Clauses will be identified by certain kind of shifts to these unbounded gaps in v. We will introduce some more notation. Let $T := \{(x_i^j, y_i^j, z_i^j) \mid i \in \{1, \dots, N\} \text{ and } j \in \{1,2,3\}\}$ be the triplets of clocks associated with each literal. A literal is said to be *positive* if it is a variable p, and it is *negative* if it is the negation $\neg p$ of some variable p. We will assume that in every clause of φ, the positive literals are written before the negative literals: for example, we write $p_1 \vee p_3 \vee \neg p_2$ instead of $p_1 \vee \neg p_2 \vee p_3$. For each clause C_i, let (e_i, f_i) be the pair of clocks corresponding to C_i in the border between positive and negative literals:

$$(e_i, f_i) := \begin{cases} (r_{i-1}, x_i^1) & \text{if all literals in } C_i \text{ are negative} \\ (z_i^j, x_i^{j+1}) & \text{if for } j \in \{1,2\},\ l_i^j \text{ is positive and } l_i^{j+1} \text{ is negative} \\ (z_i^3, r_i) & \text{if all literals in } C_i \text{ are positive} \end{cases} \tag{15}$$

Given the formula φ, the above border clocks are fixed. For a valuation $v \in Z$ and $i \in \{1, \dots, N\}$, define v_i to be the valuation such that:

- $v_i(y) - v_i(x) = v(y) - v(x)$ and $v_i(z) - v_i(y) = v(z) - v(y)$ for all $(x,y,z) \in T$
- $v_i(f_i) - v_i(e_i) = 2M + 1$ and $v_i(f_{i'}) - v_i(e_{i'}) = 2M$ for all $i' \neq i$,
- $v_i(r_0) = 0$ and all other differences between consecutive clocks (according to order given by (13)) is $2M - 3$.

Notice that $v_i \simeq_{LU} v$. Valuation v_i acts as a representative for the clause C_i, through the choice of the difference $2M+1$ in the border of C_i, and $2M$ in the other borders. We want to construct zone Z' such that when C_i is true, the valuation v_i forms a negative cycle with the constraints of Z', via the literal which is true in C_i.

Construction of Z'. Zone Z' is described by five sets of constraints. The first set of constraints are between the clocks of the same literal, and are identical to that in Z:

$$y_i^j - x_i^j \geq 1 \quad \text{and} \quad z_i^j - y_i^j \geq 1 \quad \text{and} \quad z_i^j - x_i^j = 3 \tag{16}$$

The second set of constraints are for border clocks in each clause. For each $i \in \{1, \dots, N\}$:

$$2M \leq f_i - e_i \leq 2M + 1 \tag{17}$$

where e_i and f_i are according to the definition in (15). The third set of constraints fix differences between consecutive blocks not involving border clocks to $2M-3$.

$$\begin{aligned} x_i^1 - r_{i-1} &= 2M - 3 \quad \text{if } (r_{i-1}, x_i^1) \neq (e_i, f_i) \text{ and} \\ x_i^{j+1} - z_i^j &= 2M - 3 \quad \text{for } j \in \{1,2\} \text{ when } (z_i^j, x_i^{j+1}) \neq (e_i, f_i) \quad \text{and} \\ r_i - z_i^3 &= 2M - 3 \quad \text{when } (z_i^3, r_i) \neq (e_i, f_i) \end{aligned} \tag{18}$$

From (16,17,18), we see that for every valuation in Z' the difference between separators, that is $r_i - r_{i-1}$, is between $8M$ and $8M+1$ with the flexibility coming from $f_i - e_i$. The fourth set of constraints ensures that at least one of the $f_i - e_i$ should be bigger than $2M$.

$$r_N - r_0 \geq (8M \cdot N) + 1 \tag{19}$$

So far, the constraints that we have chosen for Z' do not talk about clauses being true or false. Recall that valuation v_i where the border $v_i(f_i) - v_i(e_i) = 2M+1$ represents the choice of C_i for evaluation. The final set of constraints ensure that for every integral valuation v' in Z' which has $v'(f_i) - v'(e_i) = 2M+1$, every literal in C_i evaluates to false under the encoding scheme given in Page 13: that is, if l_i^j is positive then $v'(y_i^j) - v'(x_i^j)$ cannot be 1 and when l_i^j is negative, $v'(y_i^j) - v'(x_i^j)$ cannot be 2. For a positive literal l_i^j let $d_i^j \in \{0,1,2\}$ be the number of (x,y,z) blocks corresponding to positive literals between z_i^j and f_i (does not include j). Similarly, for a negative literal, let $d_i^j \in \{0,1,2\}$ be the number of blocks corresponding to negative literals between e_i and x_i^j (again, excludes j). We add the following constraints:

$$\begin{aligned} f_i - y_i^j &\leq d_i^j \cdot 2M + (2M+2) \quad \text{if } l_i^j \text{ is a positive literal} \\ y_i^j - e_i &\leq d_i^j \cdot 2M + (2M+2) \quad \text{if } l_i^j \text{ is a negative literal} \end{aligned} \tag{20}$$

▶ **Theorem 21.** *Formula φ is satisfiable iff $Z \not\preccurlyeq^d_{LU} Z'$ for the zones Z, Z' and LU bounds function described above.*

Proof sketch. Assume φ is satisfiable. Consider the valuation $v \in Z$ corresponding to the satisfying assignment. Pick an arbitrary $v' \simeq_{LU} v$. If v' were to lie in Z', by (19), at least one of the border differences should be $> 2M$. This forms a contradiction with the literal that is true in clause C_i due to (20).

Assume $Z \not\preccurlyeq^d_{LU} Z'$. As Z and Z' are topologically closed, and the $\multimap^*$ equivalence classes are same for every valuation in Z, by Proposition 20 there is an integral valuation v such that $[v]_{LU} \cap Z'$ is empty. This v gives a satisfying assignment. Mainly, each v_i corresponding to v will form a negative cycle with some literal clocks of C_i, and this literal will be made true by the assignment corresponding to v. ◀

Table 1 Experiments to compare forward analysis with diagonal constraints versus forward analysis on the equivalent diagonal free automaton. "Fischer K" is a model of a communication protocol with K processes as described in [18]. Cex 1 is the automaton in [5] which revealed the bug with the explicit abstraction method. Cex 2 is a similar version with more states, given in [18].

Model		Diagonal constraints + $\preccurlyeq^d_{LU}$		Diagonal free + $\mathfrak{a}_{\preccurlyeq LU}$	
Name	# clocks	# zones	time (in sec.)	# zones	time (in sec.)
Cex 1	4	8	0.08	22	0.02
Cex 2	8	273	30.1	2051	0.1
Fischer 4	8	933	18.3	73677	2.1
Fischer 5	10	4181	132.5	1926991	117.1

Theorem 21 leads to the following result.

▶ **Theorem 22.** *The decision problem $Z \not\preccurlyeq^d_{LU} Z'$ is NP-hard.*

6 Conclusion

In this paper, we have proposed a simulation $\preccurlyeq^d_{LU}$ and a simulation test $Z \preccurlyeq^d_{LU} Z'$ that facilitates a forward analysis procedure for timed automata with diagonal constraints. An abstraction function based on symmetric $\preccurlyeq^d_{LU}$ was already proposed in [5] in the context of forward analysis using explicit abstractions, but it was not used as no efficient storage mechanisms for non-convex abstractions are known. Moreover, no simulation test apart from a brute force check of enumerating over all regions was known either. Here, we provide a more refined simulation test, which in principle gives a more structured way of performing this enumeration. In the diagonal free case, this test can be performed in $O(|X|^2)$ [16]. But, as we show here, in the presence of diagonal constraints, $Z \not\preccurlyeq^d_{LU} Z'$ is NP-complete. Nevertheless, having this forward analysis framework creates the possibility to incorporate recent optimizations studied for diagonal free automata which crucially depend on this inclusion test, and have been indispensable in improving the performance substantially [14, 15]. Moreover, we believe that this framework can be extended to various other problems involving timed automata with diagonal constraints, for instance liveness verification and cost optimal reachability in priced timed automata.

We have implemented reachability for timed automata with diagonal constraints using simulation test $Z \preccurlyeq^d_{LU} Z'$ in a prototype tool T-Checker [13] which has been developed for diagonal free timed automata. The simulation test constructs an SMT formula in linear arithmetic and invokes the Z3 solver [10]. Preliminary experiments on models from [18] are reported in Table 1. For each model $\mathcal{A}$ (with diagonal constraints), the table compares the performance of running the forward analysis approach using $\preccurlyeq^d_{LU}$ on $\mathcal{A}$ (Columns 3 and 4) versus the forward analysis using (diagonal free variant) $\mathfrak{a}_{\preccurlyeq LU}$ [2] on the equivalent diagonal free automaton $\mathcal{A}_{df}$ (Columns 5 and 6). We observe that there is a significant decrease in the number of nodes explored while using $\preccurlyeq^d_{LU}$ on $\mathcal{A}$. The problem with $\mathcal{A}_{df}$ is that each state q of $\mathcal{A}$ has 2^d copies in $\mathcal{A}_{df}$ if d is the number of diagonal constraints (essentially, the states of $\mathcal{A}_{df}$ maintain the information about whether each diagonal is true or false when reaching this state). Therefore a simulation of the form $Z \preccurlyeq^d_{LU} Z'$ arising from (q, Z) and (q, Z') which occurs in the analysis of $\mathcal{A}$ might not be possible while analyzing $\mathcal{A}_{df}$ just because the corresponding paths reach different copies of q, say (q_1, Z) and (q_2, Z'). This prunes the search faster in $\mathcal{A}$. Indeed, exploiting the conciseness of diagonal constraints could be a valuable tool for modeling and verifying real-time systems. We believe that the performance of our algorithm in terms of time is encouraging: despite the preliminary nature of our implementation, our naïve SMT encoding and the underlying hardness of the simulation test, the time taken is comparable to the diagonal free conversion. Investigating efficient methods for $Z \preccurlyeq^d_{LU} Z'$ and comparing our method with other approaches [8] is part of future work.

References

1 Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

2 Gerd Behrmann, Patricia Bouyer, Kim G. Larsen, and Radek Pelánek. Lower and upper bounds in zone-based abstractions of timed automata. *International Journal on Software Tools for Technology Transfer*, 8(3):204–215, 2006.

3 Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 87–124. Springer, 2004.

4 Béatrice Bérard, Antoine Petit, Volker Diekert, and Paul Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2,3):145–182, 1998.

5 Patricia Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004.

6 Patricia Bouyer and Fabrice Chevalier. On conciseness of extensions of timed automata. *Journal of Automata, Languages and Combinatorics*, 10(4):393–405, 2005.

7 Patricia Bouyer, Maximilien Colange, and Nicolas Markey. Symbolic optimal reachability in weighted timed automata. In *Computer Aided Verification (CAV)*, volume 9779 of *Lecture Notes in Computer Science*, pages 513–530. Springer, 2016.

8 Patricia Bouyer, François Laroussinie, and Pierre-Alain Reynier. Diagonal constraints in timed automata: Forward analysis of timed systems. In *Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 3829 of *Lecture Notes in Computer Science*, pages 112–126. Springer, 2005.

9 Conrado Daws and Stavros Tripakis. Model checking of real-time reachability properties using abstractions. In *Tools and Algorithms for the Construction and Analysis of Systems, (TACAS)*, volume 1384 of *Lecture Notes in Computer Science*, pages 313–329. Springer, 1998.

10 Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

11 David L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1989.

12 Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Reachability in timed automata with diagonal constraints. *CoRR*, abs/1806.11007, 2018. `arXiv:1806.11007`.

13 Frédéric Herbreteau. TChecker. `http://www.labri.fr/perso/herbrete/tchecker/index.html`.

14 Frédéric Herbreteau, Dileep Kini, B. Srivathsan, and Igor Walukiewicz. Using non-convex approximations for efficient analysis of timed automata. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 13 of *LIPIcs*, pages 78–89. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

15 Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Lazy abstractions for timed automata. In *Computer Aided Verification (CAV)*, volume 8044 of *Lecture Notes in Computer Science*, pages 990–1005. Springer, 2013.

16 Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Better abstractions for timed automata. *Information and Computation*, 251:67–90, 2016.

17 Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

18 Pierre-Alain Reynier. Diagonal constraints handled efficiently in uppaal. In *Research report LSV-07-02*, Laboratoire Spécification et Vérification. ENS Cachan, France, 2007.

19 Sergio Yovine. Kronos: A verification tool for real-time systems. (Kronos user's manual release 2.2). *International Journal on Software Tools for Technology Transfer*, 1:123–133, 1997.

20 Jianhua Zhao, Xuandong Li, and Guoliang Zheng. A quadratic-time dbm-based successor algorithm for checking timed automata. *Information Processing Letters*, 96(3):101–105, 2005.