FI MU

Faculty of Informatics
Masaryk University Brno

On Clock-Aware LTL Properties of Timed Automata

by

Peter Bezděk
Nikola Beneš
Vojtěch Havel
Jiří Barnat
Ivana Černá

FI MU Report Series **FIMU-RS-2014-04**

 June 2014

Publications in the FI MU Report Series are in general accessible via WWW:

http://www.fi.muni.cz/reports/

Further information can be obtained by contacting:

Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic

On Clock-Aware LTL Properties of Timed Automata

Peter Bezděk Nikola Beneš* Vojtěch Havel
Jiří Barnat Ivana Černá†

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
{xbezdek1,xbenes3,xhavel1,barnat,cerna}@fi.muni.cz

June 10, 2014

Abstract

We introduce the *Clock-Aware Linear Temporal Logic* (CA-LTL) for expressing linear time properties of timed automata, and show how to apply the standard automata-based approach of Vardi and Wolper to check for the validity of a CA-LTL formula over the continuous-time semantics of a timed automaton. Our model checking procedure employs zone-based abstraction and a new concept of the so called ultra-regions. We also show that the Timed Büchi Automaton Emptiness problem is not the problem that the intended automata-based approach to CA-LTL model checking is reduced to. Finally, we give the necessary proofs of correctness, some hints for an efficient implementation, and preliminary experimental evaluation of our technique.

1 Introduction

Model checking [CGP99] is a formal verification technique applied to check for logical correctness of discrete distributed systems. While it is often used to prove the unreachability of a bad state (such as an assertion violation in a piece of code), with a proper

*The author has been supported by the MEYS project No. CZ.1.07/2.3.00/30.0009 Employment of Newly Graduated Doctors of Science for Scientific Excellence.

†The author has been supported by the MEYS project No. LH11065 Control Synthesis and Formal Verification of Complex Hybrid Systems.

specification formalism, such as the *Linear Temporal Logic* (LTL), it can also check for many interesting liveness properties of systems, such as repeated guaranteed response, eventual stability, live-lock, etc.

Timed automata have been introduced in [AD94] and have become a widely accepted framework for modelling and analysis of time-critical systems. The formalism is built on top of the standard finite automata enriched with a set of real-time clocks and allowing the system actions to be guarded with respect to the clock valuations. In the general case, such a timed system exhibits infinite-state semantics (the clock domains are continuous). Nevertheless, when the guards are limited to comparing clock values with integers only, there exists a bisimilar finite state representation of the original infinite-state real-time system referred to as the region abstraction. The region abstraction builds on top of the observation that concrete real-time clock valuations that are between two consecutive integers are indistinguishable with respect to the valuation of an action guard. Unfortunately, the size of the region-based abstraction grows exponentially with the number of clocks and the largest integer number used. As a result, the region-based abstraction is difficult to be used in practice for the analysis of more than academic toy examples, even though it has its theoretical value.

A practically efficient abstraction of the infinite-state space came with the so called zones [DT98]. Unlike the region-based abstraction, a single state in the zone-based abstraction is no more restricted to represent only those clock values that are between two consecutive integers. Therefore, the zone-based abstraction is much coarser and the number of zones *reachable* from the initial state is significantly smaller. This in turns allows for efficient implementation of verification tools for timed automata, see e.g. UPPAAL [BDL+01].

In this paper we solve the model checking problem of linear time properties over timed automata. To that end we introduce *Clock-Aware Linear Temporal Logic* (CA-LTL), which is a linear time logic built from the standard boolean operators, the standard LTL operator *Until*, and atomic propositions that are boolean combinations of comparisons of clock valuations against integer constants and guards over variables of an underlying timed automaton.

The ability to use clock-valuation constraints as atomic propositions makes the newly introduced logic rather powerful. Note, for example, that in terms of expressibility, it completely covers the fragment of TCTL as used for specification purposes by UPPAAL model checker. The non-trivial expressive power of CA-LTL is also witnessed with a CA-LTL formula $\mathbf{FG}(x \leq 3)$ expressing that the timed automaton under investigation will eventually come to a stable state where it is guaranteed that from that time on the clock variable x will never exceed the value of 3, i.e. a reset of x is going to happen somewhat regularly.

Regarding model checking of CA-LTL we stress that we are aware of the so called *Timed Büchi Automaton Emptiness* problem [Tri09a, Li09, LOD$^+$13]. Timed Büchi Automaton Emptiness could be considered as the solution to the problem of LTL model checking over timed automata provided that the logic used cannot refer to clock valuations. However, for CA-LTL we believe and show later in this paper that the solution of CA-LTL model checking does not reduce to the problem of Timed Büchi Automaton Emptiness.

1.1 Contribution

In this paper we define the syntax and the continuous-time semantics of the*Clock-Aware Linear Temporal Logic* (CA-LTL). We then show how to apply the standard automata-based approach to LTL model checking of Vardi and Wolper [VW86] for a CA-LTL formula and a timed automaton. In particular, we show how to construct a Büchi automaton coming from the CA-LTL specification with a zone-based abstraction of a timed automaton representing the system under verification using the so-called ultraregions. We give the necessary proof of correctness of our construction and list some hints that lead towards an efficient implementation of it. We also report on the practical impact of introducing ultraregions on the size of the zone-base abstracted timed automaton graph.

1.2 Outline

The rest of the paper is organised as follows. We first list the preliminaries and define our new CA-LTL in Section 2. Then, we relate CA-LTL with other logics and explain the motivation behind our approach in Section 3. The technical core of the synchronised product of a Büchi automaton and a zone-based abstracted timed automaton is given in Section 4 including the sketch of the proof of correctness. Section 5 gives some details on the implementation of our construction and lists some experimental measurements we did. In Section 6, we explain how to express all UPPAAL specification properties in CA-LTL. Finally, Section 7 concludes the paper.

2 Preliminaries and Problem Statement

Let X be a finite set of *clocks*. A *simple guard* is an expression of the form $x \sim c$ where $x \in X$, $c \in \mathbb{N}_0$, and $\sim \in \{<, \leq, \geq, >\}$. A conjunction of simple guards is called a *guard*; the empty conjunction is denoted by the boolean constant **tt**. We use $\mathcal{G}(X)$ to denote the set of all guards over a set of clocks X. A *clock valuation over X* is a function $\eta : X \to \mathbb{R}_{\geq 0}$ assigning non-negative real numbers to each clock. We denote the set of all clock valuations over X by $\mathbb{R}^X_{\geq 0}$ and the valuation that assigns 0 to each clock by $\mathbf{0}$. For a guard g and a valuation η, we say that η *satisfies* g, denoted by $\eta \models g$, if g evaluates to true when all x in g are replaced by $\eta(x)$.

We define two operations on clock valuations. Let η be a clock valuation, d a non-negative real number and $R \subseteq X$ a set of clocks to reset. We use $\eta + d$ to denote the clock valuation that adds the delay d to each clock, i.e. $(\eta + d)(x) = \eta(x) + d$ for all $x \in X$. We further use $\eta[R]$ to denote the clock valuation that resets clocks from the set R, i.e. $\eta[R](x) = 0$ if $x \in R$, $\eta[R](x) = \eta(x)$ otherwise.

Definition 2.1. *A* timed automaton *(TA) is a tuple* $A = (L, l_0, X, \Delta, \mathit{Inv})$ *where*

- L *is a finite set of locations,*
- $l_0 \in L$ *is an initial location,*
- X *is a finite set of clocks,*
- $\Delta \subseteq L \times \mathcal{G}(X) \times 2^X \times L$ *is a finite transition relation, and*
- $\mathit{Inv} : L \to \mathcal{G}(X)$ *is an invariant function.*

We use $q \xrightarrow{g,R}_{\Delta} q'$ *to denote* $(q, g, R, q') \in \Delta$.

In the following, we assume that the invariants are upper bounds only, i.e. of the form $x < c$ or $x \leq c$. Note that this is without loss of generality, as lower bound invariants may be always moved to guards of incoming transitions.

The semantics of a timed automaton is given as a labelled transition system.

Definition 2.2. *A* labelled transition system *(LTS) over a set of symbols* Σ *is a triple* $(S, s_0, \to)$*, where*

- S *is a set of states,*
- $s_0 \in S$ *is an initial state, and*

- $\rightarrow \subseteq S \times \Sigma \times S$ *is a transition relation.*

We use $s \xrightarrow{a} s'$ *to denote* $(s, a, s') \in \rightarrow$.

Definition 2.3 (TA semantics). *Let* $A = (L, l_0, X, \Delta, Inv)$ *be a TA. The* semantics *of* A, *denoted by* $[\![A]\!]$, *is a LTS* $(S, s_0, \rightarrow)$ *over the set of symbols* $\{act, \infty\} \cup \mathbb{R}_{\geq 0}$, *where*

- $S = \{(l, \eta) \in L \times \mathbb{R}_{\geq 0}^{X} \mid \eta \models Inv(l)\} \cup \{(l, \infty) \mid Inv(l) = \mathbf{tt}\}$,
- $s_0 = (l_0, \mathbf{0})$,
- *the transition relation* $\rightarrow$ *is specified for all* $(q, \eta), (q', \eta') \in S$ *such that* η *is a clock valuation as follows:*
  - $(q, \eta) \xrightarrow{d} (q', \eta')$ *if* $q = q'$, $d \in \mathbb{R}_{\geq 0}$, *and* $\eta' = \eta + d$,
  - $(q, \eta) \xrightarrow{\infty} (q', \eta')$ *if* $q = q'$ *and* $\eta' = \infty$,
  - $(q, \eta) \xrightarrow{act} (q', \eta')$ *if* $\exists g, R : q \xrightarrow{g,R}_{\Delta} q'$, $\eta \models g$, *and* $\eta' = \eta[R]$.

The first two kinds of transitions are called delay transitions, *the latter are called* action transitions.

In the following, we assume that we only deal with deadlock-free timed automata, i.e. that the only states without outgoing transitions in $[\![A]\!]$ are of the form (l, ∞). A deadlock usually signalises a severe error in the model and its (non-)existence may be ascertained in the standard way.

A *proper run* of $[\![A]\!]$ is an alternating sequence of delay and action transitions that begins with a delay transition and is either infinite or ends with a ∞ delay transition. The length of a proper run $|\pi|$ is the number of action transitions it contains. A proper run is called a Zeno run if it is infinite while the sum of all its delays is finite. Zeno runs usually represent non-realistic behaviour and it is thus desirable to ignore them in TA analysis. However, we postpone the question of dealing with Zeno runs until Section 4.

We now define the syntax and semantics of the clock-aware linear temporal logic. The atomic propositions of this logic are going to be of two kinds—those that consider properties of locations and those that consider properties of clocks. The former ones, which we call location propositions are just arbitrary symbols that are assigned to locations via a labelling function. The latter ones are simple guards over the set of clocks.

Definition 2.4 (CA-LTL syntax). *Let $Ap = Lp \cup G$ where Lp is a set of* location propositions *and G is a set of simple guards. A* clock-aware linear temporal logic *(CA-LTL) formula over Ap is defined as follows:*

$$\varphi ::= l \mid g \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\, \mathbf{U}\, \varphi$$

where $l \in Lp$ and $g \in G$.

We also use the standard derived boolean operators such as $\wedge$ and $\Rightarrow$, and the usual derived temporal operators $\mathbf{F}\,\varphi \equiv \mathbf{tt}\,\mathbf{U}\,\varphi$, $\mathbf{G}\,\varphi \equiv \neg\,\mathbf{F}\neg\varphi$.

We want our semantics of CA-LTL to reason about continuous linear time. We thus need a notion of a (continuous) suffix of a proper run. For a proper run $\pi = (l_0, \eta_0) \xrightarrow{d_0} (l_0, \eta_0 + d_0) \xrightarrow{act} (l_1, \eta_1) \xrightarrow{d_1} (l_1, \eta_1 + d_1) \xrightarrow{act} \cdots$ we define its suffix $\pi^{k,t}$ as follows:

- if $|\pi| > k$ and $t \leq d_k$ then $\pi^{k,t} = (l_k, \eta_k + t) \xrightarrow{d_k - t} (l_k, \eta_k + d_k) \xrightarrow{act} \cdots$,
- if $|\pi| = k$ then $\pi^{k,t} = (l_k, \eta_k + t) \xrightarrow{\infty} (l_k, \infty)$,
- otherwise, $\pi^{k,t}$ is undefined.

Note that the condition $|\pi| = k$ implies that π ends with $\cdots (l_k, \eta_k) \xrightarrow{\infty} (l_k, \infty)$. We further define an ordering on the set of suffixes of π, denoted by $\lhd_\pi$ as follows: $\pi^{i,t} \lhd_\pi \pi^{j,s}$ if both $\pi^{i,t}$ and $\pi^{j,s}$ are defined and either $i < j$ or $i = j$ and $t \leq s$. (The semantics is that $\pi^{i,t}$ is an "earlier" suffix of π than $\pi^{j,s}$.)

Definition 2.5 (CA-LTL semantics). *Let $\mathcal{L} : L \to 2^{Lp}$ be a function that assigns a set of location propositions to each location. The semantics of a CA-LTL formula φ on a proper run $\pi = (l_0, \eta_0) \xrightarrow{d} (l_0, \eta_0 + d_0) \xrightarrow{act} (l_1, \eta_1) \xrightarrow{d1} \cdots$ with a labelling $\mathcal{L}$ is given as follows (the semantics of the boolean operators is the usual one and is omitted here):*

$$\begin{aligned}
\pi \models p \quad &\iff p \in \mathcal{L}(l_0) \\
\pi \models g \quad &\iff \eta_0 \models g \\
\pi \models \varphi\, \mathbf{U}\, \psi \quad &\iff \exists k, t : \pi^{k,t} \textit{ is defined, } \pi^{k,t} \models \psi\textit{, and} \\
&\qquad \forall j, s \textit{ such that } \pi^{j,s} \lhd_\pi \pi^{k,t} : \pi^{j,s} \models \varphi \vee \psi
\end{aligned}$$

For a timed automaton A with a location labelling function $\mathcal{L}$, we say that A with $\mathcal{L}$ satisfies φ, denoted by $(A, \mathcal{L}) \models \varphi$ if for all proper runs π of $[\![A]\!]$, $\pi \models \varphi$.

The goal of this paper is to solve the following problem.

CA-LTL Model Checking Problem. *Given a timed automaton A, a location labelling function $\mathcal{L}$, and a CA-LTL formula φ, decide whether $(A, \mathcal{L}) \models \varphi$.*

3 Related Work and Motivation

There is a plethora of derivatives of linear temporal logics for the specification of properties of real-time systems, timed automata in particular. To name at least some of them, we list TPTL [AH94], MTL [Koy90], MITL [AFH96], RTTL [Ost89], XCTL [HLP90], CLTL [DD07], and LTLC [LT02]. These logics employ various ways of expressing time aspects of underlying systems including one global time clock, time-bounded temporal operators, timing variables with quantifiers, and freeze operators. Some logics are defined with the use of time sampling semantics, which has been shown to be counter-intuitive [AM04]. The key aspect differentiating our CA-LTL from the logics mentioned above is the ability to properly and intuitively reason about clock values in the classical continuous-time semantics while still preserving practical efficiency of the model checking process.

Similar qualities are found in the branching time logic TCTL [BK08] a subset of which is actually supported with UPPAAL tool. Our motivation to introduce CA-LTL was to mimic the branching time TCTL in a linear time setting. We stress that CA-LTL is able to reason about values of clocks in timed automata while still being practically simple enough to allow for an efficient model checking procedure. Note that the inclusion of time-bounded operators, such as the *until* operator of TCTL, would lead to the expressive power of at least MTL, model checking of which is considered computationally infeasible. CA-LTL can thus be seen as a practically motivated extension of LTL, which is powerful enough to express the same properties as can be expressed by the specification language of the world-wide leading timed automata verification tool UPPAAL.[1]

Timed automata can be defined with different types of semantics. The standard continuous-time semantics (as used also for the definition of CA-LTL) is in many cases substituted with the so called sampling semantics. However, it has been shown in [AM04] that cycle detection under the sampling semantics of timed automata with unknown sampling rate is undecidable.

We now use an example of a timed automaton and some CA-LTL formulae to explain the intricacies of our model checking problem.

Example 3.1. *Let us consider a timed automaton as given in Fig. 1 with the labelling function $\mathcal{L}$ assigning to each location its own name only, i.e. $\mathcal{L}(\mathfrak{l}) = \{\mathfrak{l}\}$ for all $\mathfrak{l}$. Let us further consider the*

[1] For more details, see Section 6.

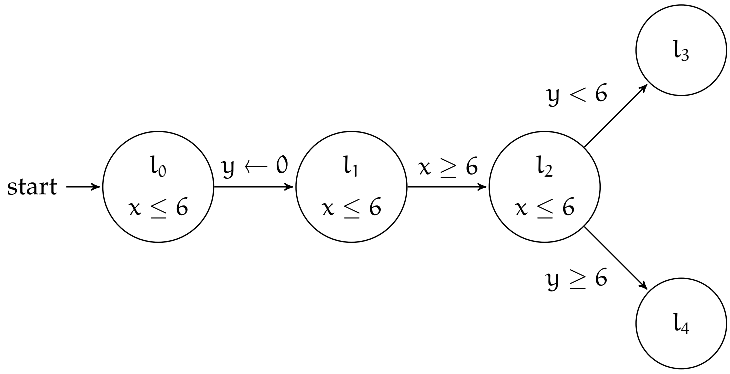

Figure 1: Timed automaton $A_{3.1}$

CA-LTL formulae

$$\varphi = \mathbf{G}(l_1 \Rightarrow ((x \leq 3 \wedge y \leq 3)\, \mathbf{U}\, (x > 3 \wedge y > 3))) \quad \textit{and} \quad \psi = \mathbf{F}\, l_3.$$

Note that while there exists a run satisfying φ and a run satisfying ψ, there is no run satisfying their conjunction, $\varphi \wedge \psi$. The reason is that the runs satisfying φ always perform the reset of y at time 0, while the runs satisfying ψ always perform the reset of y at some other time, to be able to satisfy the guard $y < 6$ together with $x = 6$.

First of all, note that there is no obvious way of combining this TA with a Büchi automaton representing the formula φ (or its negation). The reason is that while staying in l_0, the satisfaction of the guards $x \leq 3$, $y \leq 3$ changes. We could try splitting each location into several ones such that staying in each of these new locations ensures no changes of the guards. However, under the standard TA semantics, such feature is impossible. Indeed, if there were two locations with invariants $x \leq 3$ and $x > 3$, respectively, no transition between them could be enabled at any time. There thus appears to be no direct way of reducing our problem to Timed Büchi Automaton Emptiness.

One way of solving the problem whether a timed automaton satisfies a CA-LTL formula is to evaluate the formula as a standard LTL formula over the automaton's region graph. Suppose that we have the standard region graph construction [ACD90], in which the maximal bounds on each clock also include the bounds appearing in the formula. Then the satisfaction of guards inside a region never changes. This shows that the CA-LTL problem is in the PSPACE complexity class, as both the region graph and the Büchi automaton for the formula may be created on the fly. However, it is well known that the number of regions is impractically large. In the following, we therefore aim to provide a zone-based model checking approach.

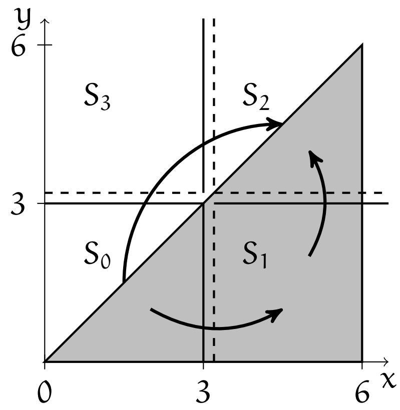

Figure 2: Illustration of the need to consider diagonals separately

Considering the standard zone-based approach [Tri09b], the main issue pointed out above remains—the satisfaction of the guards differs for various parts of a zone. Our first idea is to slice (pre-partition) the zones according to the guards of the formula. In our example, this would mean to slice the zones into one of the four "quadrants" $[0,3] \times [0,3]$, $(3,\infty) \times [0,3]$, $(3,\infty) \times (3,\infty)$, $[0,3] \times (3,\infty)$. These are illustrated in Fig. 2 and named S_0 to S_3. Every sliced zone now respects the guards $x \leq 3$, $y \leq 3$. Note, however, that this partitioning also comes with the need of describing new transitions between the newly defined zone slices. These new transitions correspond to the passage of time within the original zone. Now, consider again Example 1 and the zone that is created after the transition from l_0 to l_1 is taken. The zone is defined as the set of all valuations of clocks ν such that $\nu(x) - \nu(y) \geq 0$ and $\nu(x), \nu(y) \in [0,6]$, also illustrated with the greyed area in Fig. 2.

Let us take the S_0 slice of this zone. The next slice is not uniquely determined. One candidate is the S_1 slice as all valuations with $\nu(x) - \nu(y) > 0$ will reach this slice with the passage of time. However, we also have another candidate, the S_2 slice. This is due to the fact that all valuations with $\nu(x) - \nu(y) = 0$ reach the S_2 slice immediately after leaving the S_0 slice. We cannot take both options with a nondeterministic choice. This would introduce incorrect behaviour, as then there would be a run in the zone graph satisfying the conjunction of formulae $\varphi \wedge \psi$. Therefore, we also need to take diagonals into account. The problem here is very similar to the problem that led to the inclusion of diagonals into standard region graphs. Our slicing areas thus somehow resemble regions, only much larger. Also, their count is only dependent on the number of guards appearing in the CA-LTL formula, and may thus be expected to be reasonable. For their similarity with regions, we call these areas *ultraregions*.

4 Zone-Ultraregion Semantics

In the following, let X be a fixed set of clocks and G a fixed set of simple guards over X. For a clock $x \in X$, we define $\mathcal{I}_x$ to be the coarsest interval partition of $\mathbb{R}_{\geq 0}$ that respects the guards in G, i.e. all values in an interval have to satisfy the same subset of guards of G. Let further B_x denote the set of bounds the clock x is compared against in the guards of G and let $B_{x-y} = \{a - b \mid a \in B_x, b \in B_y\}$. Let then $\mathcal{I}_{x-y} = \{(-\infty, c_0), [c_0, c_0], (c_0, c_1), \ldots, (c_k, \infty)\}$ where $B_{x-y} = \{c_0, \ldots, c_k\}$ and $c_0 < c_1 < \cdots < c_k$. For a valuation $\eta \in \mathbb{R}^X_{\geq 0}$ we use $\mathcal{I}_x(\eta)$ to denote the interval of $\mathcal{I}_x$ that contains the value $\eta(x)$, similarly for $\mathcal{I}_{x-y}(\eta)$. We say that $\mathcal{I}_x(\eta)$ is *unbounded* if it is of the form $[c, \infty)$ or (c, ∞), otherwise we say that it is *bounded*. We now define an equivalence relation with respect to a set of simple guards G on clock valuations.

Definition 4.1 (Ultraregions)**.** *Let X be a set of clocks, G a set of simple guards over X. We define a relation $\simeq_G$ on $\mathbb{R}^X_{\geq 0}$ as follows: $\eta \simeq_G \eta'$ if for all x, $\mathcal{I}_x(\eta) = \mathcal{I}_x(\eta')$, and for all y, z such that $\mathcal{I}_y(\eta)$ and $\mathcal{I}_z(\eta)$ are* bounded, *$\mathcal{I}_{y-z}(\eta) = \mathcal{I}_{y-z}(\eta')$. The equivalence classes of $\simeq_G$ are called the* ultraregions *of G.*

Note that every ultraregion is uniquely identified by a choice of intervals from $\mathcal{I}_x$ and $\mathcal{I}_{x-y}$ for all clocks x, y. Also note that a choice in $\mathcal{I}_{x-y}$ always determines a choice in $\mathcal{I}_{y-x}$.

Example 4.2. *Let $G = \{x \leq 3, x < 6, y \leq 4\}$. Then $\mathcal{I}_x = \{[0,3], (3,6), [6,\infty)\}$, $\mathcal{I}_y = \{[0,4], (4,\infty)\}$, and $\mathcal{I}_{x-y} = \{(-\infty,-1), [-1,-1], (-1,2), [2,2], (2,\infty)\}$.*

The ultraregions of G look as follows:

$$
\begin{aligned}
U_1 &= \{\eta = (x,y) \mid x \in [0,3], y \in [0,4], x - y < -1\} \\
U_2 &= \{\eta = (x,y) \mid x \in [0,3], y \in [0,4], x - y = -1\} \\
U_3 &= \{\eta = (x,y) \mid x \in [0,3], y \in [0,4], x - y \in (-1,2)\} \\
U_4 &= \{\eta = (x,y) \mid x \in [0,3], y \in [0,4], x - y = 2\} \\
U_5 &= \{\eta = (x,y) \mid x \in [0,3], y \in [0,4], x - y > 2\} \\
U_6 &= \{\eta = (x,y) \mid x \in [0,3], y > 4\} \\
U_7 &= \{\eta = (x,y) \mid x \in (3,6), y > 4\}
\end{aligned}
$$

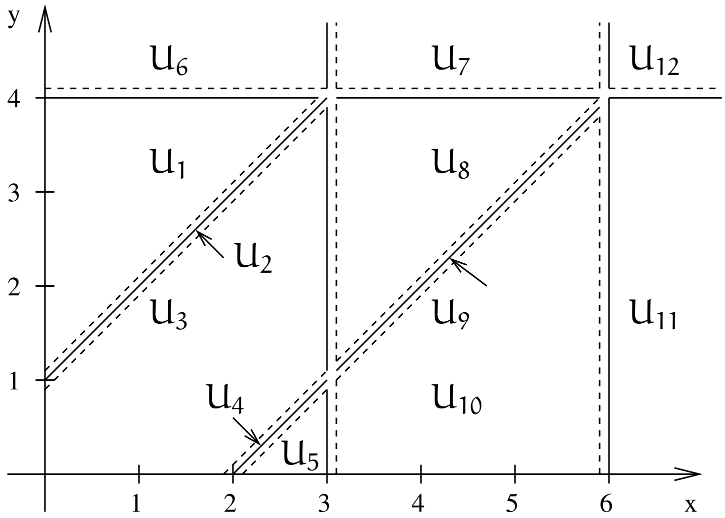

Figure 3: Ultraregions of $G = \{x \leq 3, x < 6, y \leq 4\}$

$$U_8 = \{\eta = (x, y) \mid x \in (3, 6), y \in [0, 4], x - y \in (-1, 2)\}$$
$$U_9 = \{\eta = (x, y) \mid x \in (3, 6), y \in [0, 4], x - y = 2\}$$
$$U_{10} = \{\eta = (x, y) \mid x \in (3, 6), y \in [0, 4], x - y > 2\}$$
$$U_{11} = \{\eta = (x, y) \mid x \geq 6, y \in [0, 4]\}$$
$$U_{12} = \{\eta = (x, y) \mid x \geq 6, y > 4\}$$

These ultraregions are illustrated in Figure 3.

Let $U \neq U'$ be ultraregions. We say that U' is a successor of U if for all $\eta \in U$ there exists $d \in \mathbb{R}_{>0}$ such that $\eta + d \in U'$ and $\forall 0 \leq d' \leq d : \eta + d' \in U \cup U'$.

Lemma 4.3. *An ultraregion has at most one successor.*

Proof. Let U be an ultraregion and let U', U'' be its successors. We show that $U' \cap U'' \neq \emptyset$. As ultraregions are equivalence classes, this is equivalent to $U' = U''$. Let us choose an arbitrary $\eta \in U$. There thus exist d_1 and d_2 satisfying the definition of a successor for U' and U'', respectively. Take $d = \min\{d_1, d_2\}$. W.l.o.g. assume that $d = d_1$. Then $\eta + d \in U'$. As $d \leq d_2$, we also know that $\eta + d \in U \cup U''$. This means that $\eta + d \in U' \cap (U \cup U'')$. Clearly, $\eta + d \notin U$, as $U \neq U'$. Therefore, $\eta + d \in U' \cap U''$. □

This allows us to denote the successor of U by $succ(U)$. If U has no successor, we additionally define $succ(U) = U$. Note that $succ(U) = U$ if and only if all clocks are unbounded in U, i.e. for every $\eta \in U$ and every $d \in \mathbb{R}_{\geq 0}$, $\eta + d \in U$.

Let now $R \subseteq X$ be a set of clocks. The *reset* of U with respect to R, denoted by $U\langle R\rangle$, is defined as follows:

$$U\langle R\rangle = \{U' \mid U' \text{ is an ultraregion } \wedge \exists \eta \in U : \eta[R] \in U'\}$$

Example 4.4. *Continuing with Example 4.2, we may see that e.g. succ*$(U_8) = U_7$, *succ*$(U_{12}) = U_{12}$, $U_{11}\langle x\rangle = \{U_1, U_2, U_3\}$, $U_9\langle x, y\rangle = \{U_3\}$, *and* $U_5\langle x\rangle = \{U_3\}$.

We may now define the zone-ultraregion semantics of a timed automaton. We use the standard notion of clock zones here [Tri09b]. Every zone is described by a set of diagonal constraints of the form $x_i - x_j \prec_{ij} c_{ij}$ where $c_{ij} \in \mathbb{R}$, $\prec_{ij} \in \{<, \leq\}$ for all clocks x_i, $x_j \in X \cup \{x_0\}$, and x_0 is a special clock that has always the value 0. We use these standard operations on zones: intersection $Z \cap Z'$, reset $Z[R] = \{\eta[R] \mid \eta \in Z\}$, and time passing $Z^{\uparrow} = \{\eta + d \mid \eta \in Z, d \in \mathbb{R}_{\geq 0}\}$. The zones may be efficiently represented using difference bound matrices (DBM) [Dil90, BY04]. Although there may be different representations of the same zone, it is a standard result that there exists a unique canonical representation in which the bounds $\prec_{ij} c_{ij}$ are as tight as possible.

In order to keep the number of zones finite, we use the standard *k-extrapolation* construction [Tri09b, BY04, Pet99, Bou04]. Let Z be a zone and let $\prec_{ij} c_{ij}$ be the bounds in its canonical representation. Let $M(x)$ be the highest bound in the guards of TA and the guards from G that compare against x. The extrapolated zone $\mathcal{E}(Z)$ is defined by the set of diagonal constraints $x_i - x_j \prec'_{ij} c'_{ij}$ where

$$\prec'_{ij} c'_{ij} = \begin{cases} < \infty & \text{if } c_{ij} > M(x_i), \\ < -M(x_j) & \text{if } c_{ij} < -M(x_j), \\ \prec_{ij} c_{ij} & \text{otherwise.} \end{cases}$$

Note that the ultraregions are a special case of zones (and the extrapolation does not change them). We may thus also apply the zone operations to ultraregions. However, be aware that the ultraregion reset and the zone reset of an ultraregion are different operations. This is why we use a different notation for $U\langle R\rangle$.

Definition 4.5 (Zone-ultraregion automaton)**.** *Let* $A = (L, l_0, X, \Delta, Inv)$ *be a TA and let* G *be a set of simple guards. The* zone-ultraregion automaton *(ZURA) of* A *with respect to* G *is a labelled transition system whose states are triples* (l, Z, U) *where* $l \in L$, Z *is a clock zone, and* U *is a ultraregion of* G.

The initial state is (l_0, Z_0, U_0) *where* $Z_0 = \{\mathbf{0}\}^{\uparrow} \cap Inv(l_0)$ *and* U_0 *is the ultraregion containing the zero valuation* $\mathbf{0}$. *The transitions are of two kinds:*

- *delay transitions:* $(l, Z, U) \xrightarrow{\delta} (l, Z, succ(U))$ *whenever* $Z \cap succ(U) \neq \emptyset$ *and* $U = succ(U) \implies Z = Z^{\uparrow}$,
- *action transitions:* $(l, Z, U) \xrightarrow{act} (l', \mathcal{E}(Z'), U')$ *whenever* $l \xrightarrow{g,R}_{\Delta} l'$, $U' \in U\langle R\rangle$, $Z' = ((Z \cap U \cap g)[R] \cap U')^{\uparrow} \cap Inv(l')$, *and* $Z' \cap U' \neq \emptyset$.

Example 4.6. *Continuing with Example 3.1, Fig. 4 represents the ZURA of the timed automaton* $A_{3.1}$ *with respect to* $G = \{x \leq 3, y \leq 3\}$.

A combination of a ZURA with a location labelling function $\mathcal{L}$ is interpreted as a Kripke structure [CGP99]. The states and transitions of this Kripke structure are the states and transitions of the ZURA, forgetting the labels of transitions. The state labelling function $\mathcal{L}_K$ is defined as $\mathcal{L}_K(l, Z, U) = \mathcal{L}(l) \cup \{g \in G \mid U \models g\}$. Here, $U \models g$ denotes that all valuations of U satisfy g. Due to the definition of ultraregions, this is equivalent to the existence of a valuation in U satisfying g.

In the next subsection, we are going to prove the following theorem. The theorem gives us a solution to the CA-LTL model checking problem by reducing it to the problem of standard LTL model checking of a Kripke structure.

Theorem 4.7. *Let* A *be a TA, let* A_{ZURA} *be its zone-ultraregion automaton with respect to* G. *Let further* φ *be a CA-LTL formula over* G. *Then* $A \models \varphi$ *iff* $A_{ZURA} \models \varphi$.

We finish this section with a remark about Zeno runs. It might sometimes happen that the model checking algorithm produces a counterexample that is a Zeno run of the original TA. If ignoring such runs is desirable (as it usually is), we may extend the original TA with one special clock z, add a loop on every location with guard $z = 1$ and reset $\{z\}$, and modify the original CA-LTL formula from φ to $(\mathbf{G}\,\mathbf{F}\, z \leq 0 \wedge \mathbf{G}\,\mathbf{F}\, z > 0) \Rightarrow \varphi$.

4.1 Proof of Theorem 4.7

For the remainder of this section, let us assume a fixed TA A, a fixed set of guards G, and a fixed location labelling function $\mathcal{L}$.

We show that the proper runs of $[\![A]\!]$ and the runs of A_{ZURA} are, in some sense, equivalent. In order to do that, we use the notion of a signature of a run. Intuitively, a signature is a sequence of sets of atomic propositions that hold along the given run.

We use the fact that all valuations of a given ultraregion satisfy the same set of guards. For an ultraregion U, we thus use $G(U)$ to denote the set of guards satisfied by the valuations of U.

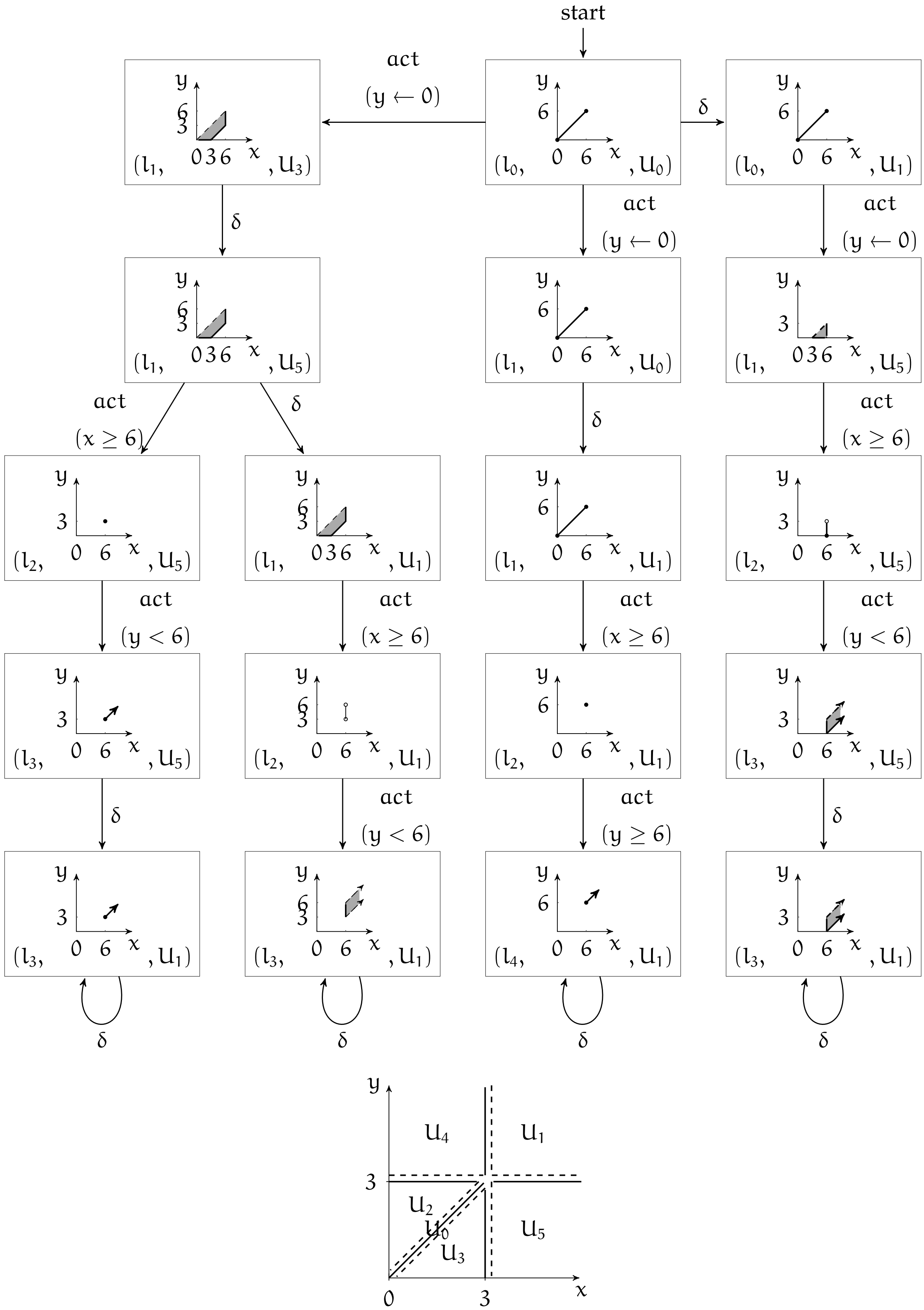

Figure 4: ZURA state space of timed automaton $A_{3.1}$ with respect to $G = \{x \leq 3, y \leq 3\}$

Definition 4.8 (Signature). *Let* G *be a set of simple guards,* Lp *a set of location propositions,* $Ap = G \cup Lp$*, and let* $\pi = (l_0, \eta_0) \xrightarrow{d_0} (l_0, \eta_0 + d_0) \xrightarrow{act} (l_1, \eta_1) \cdots$ *be a proper run of a TA. Let* $U_{j,0}$ *be the ultraregion of* G *containing* η_j *and let* $U_{j,i+1} = succ(U_{j,i})$*. For* $(l_j, \eta_j) \xrightarrow{d_j} (l_j, \eta_j + d_j)$ *in* π*, we define* $w_j \in (2^{Ap})^+$*:*

$$w_j = (\mathcal{L}(l_j) \cup G(U_{j,0})) \cdot (\mathcal{L}(l_j) \cup G(U_{j,1})) \cdots (\mathcal{L}(l_j) \cup G(U_{j,k}))$$

where k *is the least such that* $U_{j,k} = U_{j,k+1}$*. For* $(l_j, \eta_j) \xrightarrow{\infty} (l_j, \infty)$*, we define:*

$$w_j = (\mathcal{L}(l_j) \cup G(U_{j,0})) \cdot (\mathcal{L}(l_j) \cup G(U_{j,1})) \cdots$$

In this case, $w_j \in (2^{Ap})^\omega$*. The signature of* π *with respect to* Ap*, denoted by* $sig_{Ap}(\pi)$*, is defined as the infinite word* $w = w_0 w_1 w_2 \cdots w_j$ *if* π *ends with* (l_j, ∞)*,* $w = w_0 w_1 w_2 \cdots$ *otherwise.*

Our first objective is to show that the runs of A_{ZURA} represent exactly the signatures of all proper runs of A. We then show that the CA-LTL satisfaction on a proper run can be reduced to the classic LTL satisfaction on its signature. These results together imply the statement of Theorem 4.7.

Lemma 4.9. *All reachable states* (l, Z, U) *of* A_{ZURA} *satisfy the following invariants:* $Z \cap U \neq \emptyset$ *and* $Z = Z^\uparrow \cap Inv(l)$*.*

Proof. The lemma follows clearly from the definition of ZURA. □

Lemma 4.10. *Let* $(l, \eta) \xrightarrow{act} (l', \nu) \xrightarrow{d} (l', \nu + d)$ *such that* $\nu \simeq_G \nu + d$*. Let* (l, Z, U) *be a state such that* $\eta \in Z \cap U$*. Then* $(l, Z, U) \xrightarrow{act} (l', Z', U')$ *with* $\nu + d \in Z' \cap U'$*.*

Proof. Let $\eta \in Z \cap U$ and let $l \xrightarrow{g,R}_\Delta l'$ be the timed automaton transition corresponding to $(l, \eta) \xrightarrow{act} (l', \nu)$. This means that $\eta \models g$ and $\nu = \eta[R]$. Let U' be the ultraregion such that $\nu \in U'$. Clearly, $U' \in U\langle R \rangle$. Thus, $\nu \in (Z \cap U \cap g)[R] \cap U'$. As $(l', \nu) \xrightarrow{d} (l', \nu + d)$, we know that $\nu + d \in Inv(l')$. The fact that $\nu + d \in U'$ follows from $\nu \simeq_G \nu + d$. Therefore, $\nu + d \in Z' \cap U'$. □

Lemma 4.11. *Let* $(l, \eta) \xrightarrow{d} (l, \eta + d)$ *such that* $U(\eta + d) = succ(U(\eta))$ *and let* (l, Z, U) *be a state such that* $\eta \in Z \cap U$*. Let either* $succ(U) \neq U$ *or* $Inv(l) = \mathbf{tt}$*. Then* $(l, Z, U) \xrightarrow{\delta} (l, Z, succ(U))$*.*

Proof. The fact that $\eta \in Z$ together with the second invariant of Lemma 4.9 implies that $\eta + d \in Z$. Thus $Z \cap succ(U) \neq \emptyset$. If $succ(U) = U$ and $Inv(l) = \mathbf{tt}$ then the second invariant of Lemma 4.9 implies that $Z = Z^\uparrow$. In any case, $(l, Z, U) \xrightarrow{\delta} (l, Z, succ(U))$. □

Repeated application of the previous lemma gives the following result.

Lemma 4.12. *Let* $(l, \eta) \xrightarrow{d} (l, \eta + d)$ *and let* (l, Z, U) *be a state such that* $\eta \in Z \cap U$. *Then there exists* $i \in \mathbb{N}_0$ *such that* $(l, Z, U) \xrightarrow{\delta} (l, Z, \mathit{succ}(U)) \xrightarrow{\delta} \cdots \xrightarrow{\delta} (l, Z, \mathit{succ}^i(U))$ *with* $\eta + d \in Z \cap \mathit{succ}^i(U)$.

Lemma 4.13. *Let* π *be a proper run of* A *with signature* $sig_{Ap}(\pi)$. *Then there exists an infinite run* $(l_0, Z_0, U_0) \to (l_1, Z_1, U_1) \to \cdots$ *such that* $sig_{Ap}(\pi)(i) = \mathcal{L}(l_i) \cup G(U_i)$, *where* $sig_{Ap}(\pi)(i)$ *denotes the* i*th letter (from* 2^{Ap}*) of* $sig_{Ap}(\pi)$.

Proof. Let $\pi = (l_0, \eta_0) \xrightarrow{d_0} (l_0, \eta_0 + d_0) \xrightarrow{act} (l_1, \eta_1) \cdots$. Let $sig_{Ap}(\pi) = w_0 w_1 w_2 \cdots$ where w_i are as given in Definition 4.8. We inductively define states S_i and finite paths ρ_i of A_{ZURA} and then show how to build the desired run. The intended invariant is that S_i contains (l_i, η_i) and ρ_i connects S_{i-1} to S_i with the following exception: If w_j is the last in $sig_{Ap}(\pi)$ then ρ_j is an infinite path from S_{i-1}. Initially, $S_0 = (l_0, Z_0, U_0)$ and ρ_0 is empty. We now build S_{i+1} and ρ_i for $i \geq 0$.

Assume first that w_i is finite. This means that $(l_i, \eta_i) \xrightarrow{d_i} (l_i, \eta_i + d_i) \xrightarrow{act} (l_{i+1}, \eta_{i+1})$ in π. Applying Lemma 4.12 to $(l_i, \eta_i) \xrightarrow{d_i} (l_i, \eta_i + d_i)$ and S_i gives a run $S_i \xrightarrow{\delta} \cdots \xrightarrow{\delta} (l_i, Z', U')$ such that $\eta_i + d_i \in Z' \cap U'$. Applying Lemma 4.10 to $(l_i, \eta_i + d_i) \xrightarrow{act} (l_{i+1}, \eta_{i+1})$ and (l_i, Z', U') gives $(l_i, Z', U') \xrightarrow{act} (l_{i+1}, Z'', U'')$ such that $\eta_{i+1} \in Z'' \cap U''$. Let $S_{i+1} = (l_{i+1}, Z'', U'')$ and let ρ_i be the path from S_i to S_{i+1} created by the composition of the two paths above.

Assume now that w_i is infinite. This means that $(l_i, \eta_i) \xrightarrow{\infty} (l, \infty)$ and thus $\mathit{Inv}(l_i) = \mathbf{tt}$. Let $d \in \mathbb{R}_{\geq 0}$ be such that $\eta_i + d \in U$ with $U = \mathit{succ}(U)$. Such d has to exist due to the unbounded invariant of l. Applying Lemma 4.12 to $(l_i, \eta_i) \xrightarrow{d} (l_i, \eta_i + d)$ gives a run $S_i \xrightarrow{\delta} \cdots \xrightarrow{\delta} (l_i, Z', U')$ where $U' = \mathit{succ}(U')$. Applying Lemma 4.11 to $(l_i, \eta_i + d) \xrightarrow{0} (l_i, \eta_i + d)$ gives a loop $(l_i, Z', U') \xrightarrow{\delta} (l_i, Z', U')$. Combining the path with the loop gives an infinite path, let us denote it by ρ_i.

The resulting run of A_{ZURA} is $S_0 \rho_i S_1 \rho_1 \cdots$, which ends with ρ_i iff $sig_{Ap}(\pi)$ ends with w_i. It is clear that this run satisfies the statement of the lemma. $\square$

We have thus shown that every proper run of A has its counterpart in A_{ZURA}. However, to show the opposite we lack properties similar to Lemmata 4.10, 4.11, and 4.12. One of the reasons is the zone extrapolation. However, even in the absence of extrapolation, we could only prove a contravariant version of these lemmata, namely that for each transition $(l, Z, U) \xrightarrow{\gamma} (l', Z', U')$ and each $\eta \in Z' \cap U'$ there exists $\xi \in Z \cap U$ and corresponding transitions that lead from (l, ξ) to (l', η). This would allow us to prove that every finite path of A_{ZURA} has a counterpart in A, but that is insufficient, as there is an uncountably infinite number of finite paths in A and we cannot combine them to make one infinite run. We thus use a roundabout way similar to that used in [HSW11] (the differences lie in the existence of delay transitions in ZURA and in the slightly different definition of zones). We use the standard region equivalence to partition the states of A into regions and show that every run of A_{ZURA} corresponds to a run in the region graph. Again, our region graph is a bit nonstandard to account for the $\xrightarrow{\delta}$ transitions.

In the following definition, we use $\lfloor \cdot \rfloor$ to denote the floor function and $\{\cdot\}$ to denote the fractional part function.

Definition 4.14 (Region Equivalence)**.** *Let* $M \in \mathbb{N}_0$*. Two valuations* ν *and* ν' *are region equivalent w.r.t. M, denoted by* $\nu \sim_M \nu'$*, if for all* $x, y \in X$

1. $\nu(x) > M$ *iff* $\nu'(x) > M$,
2. *if* $\nu(x) \leq M$ *then* $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$,
3. *if* $\nu(x) \leq M$ *then* $\{\nu(x)\} = \{\nu'(x)\}$*, and*
4. *if* $\nu(x) \leq M$ *and* $\nu(y) \leq M$ *then* $\{\nu(x)\} \leq \{\nu(y)\}$ *iff* $\{\nu'(x)\} \leq \{\nu'(y)\}$.

The equivalence classes of $\sim_M$ *are called regions.*

In the following, let M be a fixed integer that is greater or equal to all bounds appearing in A and all bounds appearing in G. Clearly, every ultraregion is a union of regions w.r.t. M. Having such M, all the valuations in a given region are indistinguishable by any invariant or any guard appearing in A or G. We may thus extend notation from valuations to regions, such as $r \models g$, $U(r)$, and $G(r)$.

A result from [HSW11] says that $\mathcal{E}(Z) \subseteq \{r \mid r \cap Z \neq \emptyset\}$. This implies $r \cap \mathcal{E}(Z) \neq \emptyset \implies r \cap Z \neq \emptyset$. We shall use this fact in the following.

A region automaton respecting G is a finite transition system with states of the form (l, r) where l is a location and r a region with $r \models \mathit{Inv}(l)$. We have two kinds of transitions: $(l, r) \xrightarrow{\delta} (l, r')$ if there exists $\nu \in r$ and $d \in \mathbb{R}_{>0}$ such that $(l, \nu) \xrightarrow{d} (l, \nu + d)$, $\nu + d \in r'$, $U(r') = \mathit{succ}(U(r))$ and if $U(r) = U'(r)$ then $\mathit{Inv}(l) = \mathbf{tt}$; and $(l, r) \xrightarrow{\mathit{act}} (l, r')$ if there exists $d \in \mathbb{R}_{\geq 0}$, $\nu \in r$, and $\nu' \in r'$ such that $(l, \nu) \xrightarrow{\mathrm{act}} (l, \nu' - d) \xrightarrow{d} (l', \nu')$ and $U(\nu' - d) = U(\nu')$.

The following lemma follows from the fact that $\sim_M$ is a time-abstracting bisimulation [TY01].

Lemma 4.15. *If* $(l, r) \xrightarrow{\delta} (l, r')$ *then for all* $\nu \in r$ *there exists* $d \in \mathbb{R}_{>0}$ *such that* $\nu + d \in r'$ *and* $(l, \nu) \xrightarrow{d} (l, \nu + d)$*. If* $(l, r) \xrightarrow{\mathit{act}} (l', r')$ *then for all* $\nu \in r$ *there exists* $\nu' \in r'$ *and* $d \in \mathbb{R}_{\geq 0}$ *such that* $(l, \nu) \xrightarrow{\mathit{act}} (l, \nu' - d) \xrightarrow{d} (l', \nu')$.

Lemma 4.16. *Let* $\gamma \in \{act, \delta\}$. *Let* (l, Z, U) *be a reachable state of* A_{ZURA} *with* $(l, Z, U) \xrightarrow{\gamma} (l', Z', U')$. *Then for each region* r' *such that* $r' \cap Z' \cap U' \neq \emptyset$ *there exists a region* r *such that* $r \cap Z \cap U \neq \emptyset$ *and* $(l, r) \xrightarrow{\gamma} (l', r')$.

Proof. Let first $\gamma = act$. This means that $l \xrightarrow{g,R}_{\Delta} l'$ and $Z' = \mathcal{E}(Z'')$ where $Z'' = ((Z \cap U \cap g)[R] \cap U')^{\uparrow} \cap Inv(l')$. Let r' be a region with $r' \cap Z' \cap U' \neq \emptyset$. As explained above, this implies that $r' \cap Z'' \cap U' \neq \emptyset$. Let us thus take a valuation $\eta \in r' \cap Z'' \cap U'$. As $\eta \in Z''$, there has to exist $d \in \mathbb{R}_{\geq 0}$ such that $\eta - d \in (Z \cap U \cap g)[R] \cap U'$. This implies that $(\eta - d)(x) = 0$ for all $x \in R$ and that there exists $\xi \in Z \cap U \cap g$ such that $\xi(x) = (\eta - d)(x)$ for all $x \notin R$. Let r be the region that includes ξ. Then $r \cap Z \cap U \neq \emptyset$ and $(l, \xi) \xrightarrow{act} (l', \eta - d) \xrightarrow{d} (l', \eta)$. Thus $(l, r) \xrightarrow{act} (l', r')$.

Let now $\gamma = \delta$. This means that $l = l'$, $Z = Z'$ and $U' = succ(U)$. Clearly, either (l, Z, U) is reachable from the initial state only via $\xrightarrow{\delta}$ transitions or there exists some $(l, Z, \overline{U})$ such that $(l, Z, \overline{U})$ is a result of an $\xrightarrow{act}$ transition and $(l, Z, \overline{U}) \xrightarrow{\delta} \cdots \xrightarrow{\delta} (l, Z, U)$.

In the first case, the statement is obvious.

Let now $(l, Z, \overline{U})$ be as described above. This means that there is some l_p, Z_p, U_p such that $l_p \xrightarrow{g,R}_{\Delta} l$, $Z = \mathcal{E}(Z'')$ where $Z'' = ((Z_p \cap U_p \cap g)[R] \cap \overline{U})^{\uparrow} \cap Inv(l)$. Notice that $Z'' \subseteq (Z'' \cap \overline{U})^{\uparrow}$. Let now r' be a region such that $r' \cap Z \cap U' \neq \emptyset$. Then, as mentioned above, also $r' \cap Z'' \cap U' \neq \emptyset$. Let $\eta \in r' \cap Z'' \cap U'$. Due to the previous observation, there is some $h \in \mathbb{R}_{\geq 0}$ such that $\eta - h \in (Z'' \cap \overline{U})$. We know that $U = succ^i(\overline{U})$ for some i. This means that there exists some d such that $\eta - h + d \in U$. Clearly, $d \leq h$. As zones are convex, also $\eta - h + d \in Z'' \subseteq Z$. Take the region r such that $\eta - h + d \in r$. Then $r \cap Z \cap U \neq \emptyset$ and $(l, \eta - h + d) \xrightarrow{h-d} (l, \eta)$. Thus $(l, r) \xrightarrow{\delta} (l, r')$. ☐

Lemma 4.17. *Let* $(l_0, Z_0, U_0) \xrightarrow{\gamma_0} (l_1, Z_1, U_1) \xrightarrow{\gamma_1} \cdots$ *be a run of* A_{ZURA} *with* $\gamma_i \in \{act, \delta\}$. *Then there exists a run* $(l_0, r_0) \xrightarrow{\gamma_0} (l_1, r_1) \xrightarrow{\gamma_1} \cdots$ *of the region automaton such that* $r_i \subseteq Z_i \cap U_i$.

Proof. The proof of this lemma is similar to the proof of Theorem 7 in [HSW11]. We construct a directed acyclic graph with nodes (i, q_i, r_i) such that $r_i \subseteq Z_i \cap U_i$. There is an edge from (i, q_i, r_i) to $(i+1, q_{i+1}, r_{i+1})$ if $(q_i, r_i) \xrightarrow{\gamma_i} (q_{i+1}, r_{i+1})$. Lemma 4.16 ensures that every node in this graph (except for $(0, l_0, r_0)$) has at least one predecessor. As the graph is infinite with finite branching, it has an infinite path. ☐

Lemma 4.18. *Let* $(l_0, Z_0, U_0) \xrightarrow{\gamma_0} (l_1, Z_1, U_1) \xrightarrow{\gamma_1} \cdots$ *be a run of* A_{ZURA} *with* $\gamma_i \in \{act, \delta\}$. *Then there exists a proper run* π *of* A *such that* $sig_{A_p}(\pi)(i) = \mathcal{L}(l_i) \cup G(U_i)$.

Proof. Take the region automaton run of Lemma 4.17 and create a run of $[\![A]\!]$ by repeated application of Lemma 4.15. The resulting run starts with an *act* transition $(l_0, \eta) \xrightarrow{act} (l_1, \xi)$

with $\eta \in Z_0 \cap U_0$. This means that $\eta(x) = \eta(y)$ for all x, y. We extend the run into a proper run with $(l_0, \mathbf{0}) \xrightarrow{\eta(x)} (l_0, \eta)$. $\square$

We have shown that the proper runs of A and the runs of A_{ZURA} are equivalent in the sense that a signature of a proper run of A is the sequence of labellings on a run of A_{ZURA} and vice versa. What remains is to show that CA-LTL satisfaction may be reduced to standard LTL satisfaction on signatures. Let us recall that the classic satisfaction relation for LTL on an infinite word from $(2^{Ap})^\omega$ is given by $w \models p \in Ap$ if $p \in w(0)$, $w \models \varphi \,\mathbf{U}\, \psi \iff \exists k : w^k \models \psi$ and for all $j < k : w^j \models \varphi$. Here, $w(0)$ is the first letter of the word w and w^k is the kth suffix of w. We omit the standard boolean operators.

We first observe that proper run suffixes and signature suffixes correspond. This simple observation clearly follows from Definition 4.8.

Lemma 4.19. *Let π be a proper run with signature $sig_{Ap}(\pi)$ and let $\pi^{k,t}$ be defined. Then $sig_{Ap}(\pi^{k,t})$ is a suffix of $sig_{Ap}(\pi)$. Further, the set $\{sig_{Ap}(\pi^{j,s}) \mid \pi^{j,s} \lhd \pi^{k,t}\}$ is the set of all suffixes of $sig_{Ap}(\pi)$ that themselves contain $sig_{Ap}(\pi^{k,t})$ as a suffix.*

Lemma 4.20. *Let π be a proper run of a TA, let G be a set of simple guards, Lp a set of location propositions, $Ap = G \cup Lp$. Let φ be a CA-LTL formula over Ap. Then $\pi \models \varphi$ iff $sig_{Ap}(\pi) \models \varphi$ when seen as an LTL formula.*

Proof. The proof is done by induction on the structure of φ. The only interesting part is that $\pi \models \varphi \,\mathbf{U}\, \psi$ as CA-LTL iff $sig_{Ap}(\pi) \models \varphi \,\mathbf{U}\, \psi$ as LTL. However, this is just a corollary to Lemma 4.19. $\square$

This concludes the proof of Theorem 4.7.

5 Implementation

We first show how to compute the successor of a given fixed ultraregion U. Let $\widehat{X}_U \subseteq X$ be the set of all clocks bounded in U, i.e. $\widehat{X}_U = \{x \in X \mid \exists d \in \mathbb{R}_{\geq 0} : \forall \eta \in U : \eta(x) \leq d\}$.

For a clock $x \in X$ we define $U_x \in \mathcal{I}_x$ to be the interval from $\mathcal{I}_x$ such that there exists a valuation $\eta \in U$ with $\eta(x) \in U_x$. For $x, y \in \widehat{X}_U$ we further define U_{x-y} to be the interval from $\mathcal{I}_{x-y}$ such that there exists a valuation $\eta \in U$ with $\eta(x) - \eta(y) \in U_{x-y}$. Note that the existence and uniqueness of U_x and U_{x-y} follow from the definition of ultraregions, and in the latter case, the fact that we only consider differences of bounded clocks. We further use E_x to denote the right endpoint of U_x.

To establish the successor of U, we need to find out which clocks leave U soonest. We thus define the following relation on $\widehat{X}_U$:

$$x \preccurlyeq_U y \iff \forall \eta \in U : \forall d \in \mathbb{R}_{\geq 0} : \eta(x) + d \in U_x \Rightarrow \eta(y) + d \in U_y$$

It is easy to see that $\preccurlyeq_U$ is reflexive, transitive, and that for all $x, y \in \widehat{X}_U$, either $x \preccurlyeq_U y$ or $y \preccurlyeq_U x$. This means that $\preccurlyeq_U$ is a total preorder. We denote the induced equivalence by $\approx_U$. The following lemma gives us a way of computing $\preccurlyeq_U$. Here, we use the notation $a < I$ with the meaning $\forall b \in I : a < b$, similarly for $>$.

Lemma 5.1. *Let* $x, y \in \widehat{X}_U$. *Then* $x \preccurlyeq_U y$ *iff either* $(E_x - E_y) < U_{x-y}$ *or* $(E_x - E_y) \in U_{x-y} \wedge (U_x$ *right-closed* $\implies U_y$ *right-closed*).

Proof. We first prove the 'if' part. Assume first that $(E_x - E_y) < U_{x-y}$. This means that for all $\eta \in U$, $E_x - E_y < \eta(x) - \eta(y)$. Let $\eta \in U$ and $d \in \mathbb{R}_{\geq 0}$ be arbitrary and let $\eta(x) + d \in U_x$. This means that $\eta(x) \leq E_x - d$. Combining the two inequalities gives $E_x - E_y < E_x - d - \eta(y)$. Thus $\eta(y) + d < E_y$, which means that $\eta(y) + d \in U_y$ and therefore $x \preccurlyeq_U y$.

Assume now that $(E_x - E_y) \in U_{x-y}$ and that U_x right-closed implies U_y right-closed. Note that due to the definition of $\mathcal{I}_{x-y}$, $(E_x - E_y) \in U_{x-y}$ implies that U_{x-y} is a singleton interval. Thus, for all $\eta \in U$, $E_x - E_y = \eta(x) - \eta(y)$. Let again $\eta \in U$ and $d \in \mathbb{R}_{\geq 0}$ be arbitrary. Suppose that $\eta(x) + d \in U_x$ while $\eta(y) + d \notin U_y$. This is only possible if $\eta(x) + d = E_x$ while $\eta(y) + d = E_y$. But this means that U_x is right-closed while U_y is right-open, which is a contradiction. Therefore $x \preccurlyeq_U y$.

We now prove the 'only if' part of the lemma. We assume that $x \preccurlyeq_U y$. Suppose that $(E_x - E_y) \in U_{x-y}$ and U_x is right-closed. As already mentioned above, U_{x-y} is a singleton interval in this case. Let us take an arbitrary $\eta \in U$ and let $d = E_x - \eta(x)$. Clearly, $d \geq 0$. As U_x is right-closed, $\eta(x) + d \in U_x$. Due to $x \preccurlyeq_U y$, also $\eta(y) + d \in U_y$ and as $E_x - E_y = \eta(x) - \eta(y)$, we know that $\eta(y) + d = E_y$, which means that the interval U_y is right-closed.

Suppose now that $(E_x - E_y) \notin U_{x-y}$. This can mean that either $(E_x - E_y) < U_{x-y}$ or $(E_x - E_y) > U_{x-y}$. If $(E_x - E_y) > U_{x-y}$ then for all $\eta \in U$, $E_x - E_y > \eta(x) - \eta(y)$. Let $\eta \in U$ be arbitrary and let $h = E_x - E_y - \eta(x) + \eta(y) > 0$. Let further $d = E_y - \eta(y) + h/2$. Then $\eta(y) + d = E_y + h/2 \notin U_y$ while $\eta(x) + d = \eta(x) + E_y - \eta(y) + h/2 = E_x - h + h/2 = E_x - h/2 \in U_x$. By contradiction, $(E_x - E_y) < U_{x_y}$. □

We can now show the construction of a successor. Let $\widetilde{X}_U$ be the set of the smallest clocks with respect to $\preccurlyeq_U$, i.e. $\widetilde{X}_U = \{x \in \widehat{X}_U \mid \forall y \in \widehat{X}_U : x \preccurlyeq_U y\}$. For a bounded interval

$J \in \mathcal{I}_x$, denote by $J^{\uparrow}$ the interval in $\mathcal{I}_x$ such that the right endpoint of J is the left endpoint of $J^{\uparrow}$. For a clock $x \in X$ we then define U'_x as follows:

$$U'_x = \begin{cases} U_x^{\uparrow} & x \in \widetilde{X}_U \\ U_x & x \notin \widetilde{X}_U \end{cases}$$

We then define $U' = \{\eta \mid \forall x \in X : \eta(x) \in U'_x \text{ and } \forall x, y \in X : U'_x \text{ and } U'_y \text{ bounded} \Rightarrow \eta(x) - \eta(y) \in U_{x-y}\}$. We want to show that $U' = succ(U)$. In order to do that, we first need an auxiliary lemma.

Lemma 5.2. *For every* $\eta \in U$ *there exists* $d \in \mathbb{R}_{>0}$ *such that* $\forall x \in \widetilde{X}_U : \eta(x) + d \notin U_x$ *and* $\forall y \in \widehat{X}_U \setminus \widetilde{X}_U : \eta(y) + d \in U_y$.

Proof. We first note that as all clocks in $\widetilde{X}_U$ are equivalent w.r.t. $\approx_U$, we only need to consider one of them. Let $x \in \widetilde{X}_U$ be such a representant.

Let $\eta \in U$ and $y \in \widehat{X}_U \setminus \widetilde{X}_U$ be arbitrary. This means that $y \not\approx_U x$. Due to the previous lemma, either $(E_y - E_x) > U_{y-x}$ or $(E_y - E_x) \in U_{y-x}$ with U_y right-closed and U_x right-open.

If $(E_y - E_x) > U_{y-x}$ then also $(E_y - E_x) > \eta(y) - \eta(x)$. Let $d_y = E_x - \eta(x) + (E_y - E_x - \eta(y) + \eta_(x))/2$. As explained in the previous proof, $\eta(x) + d_y \notin U_x$ while $\eta(y) + d_y \in U_y$. If $(E_y - E_x) \in U_{y-x}$ with U_y right-closed and U_x right-open, let $d_y = E_y - \eta(y)$. Again, $\eta(x) + d_y \notin U_x$ while $\eta(y) + d_y \in U_y$.

Let now $d = \min_{y \in \widehat{X}_U \setminus \widetilde{X}_U} d_y$. As $\eta(x) + d_y \notin U_x$ for all y, this means that $\eta(x) + d \notin U_x$. Furthermore, as $\eta(y) + d_y \in U_y$ for all y and $\eta(y) \in U_y$, this means that $\eta(y) + d \in U_y$ (ultraregions are convex). □

Theorem 5.3. *Let* U *be an ultraregion, let* U' *be defined as above. Then* $U' = succ(U)$.

Proof. We first need to prove that U' is an ultraregion. U' clearly respects the equivalence $\simeq_G$. We only need to show that it is nonempty. Let $\eta \in U$ be arbitrary and let d be given by the previous lemma. Clearly, $\eta + d \in U'$. We also see that $\eta + d \notin U$, thus establishing that U and U' are different.

What remains is to show that $\forall 0 \leq d' \leq d : \eta + d' \in U \cup U'$. Take an arbitrary such d'. For any clock $y \notin \widetilde{X}_U$, $\eta(y) \in U_y$, $\eta(y) + d \in U_y$ and thus also $\eta(y) + d' \in U_y$ (ultraregions are convex). Let us now choose a clock $x \in \widetilde{X}_U$. We know that either $\eta(x) + d' \in U_x$ or $\eta(x) + d' \in U'_x$, as $U'_x = U_x^{\uparrow}$. In the first case, all $z \in \widetilde{X}_U$ satisfy $\eta(z) + d' \in U_z$ as all such clocks are equivalent w.r.t. $\approx_U$ and thus $\eta + d' \in U$. In the second case, all $z \in \widetilde{X}_U$ satisfy $\eta(z) + d' \in U'_z$ for the same reason and thus $\eta + d' \in U'$. □

We now show the construction of the reset operation on ultraregions. Let U be an ultraregion and let $R \subseteq X$ be a set of clocks. As an ultraregion is a special case of a zone, we may apply the standard zone reset operation on U and then tighten the constraints using the standard Floyd-Warshall algorithm approach. The resulting zone can be written as:

$$U[R] = M = \{\eta \mid \forall x \in R : \eta(x) = 0; \forall x \notin R : \eta(x) \in M_x$$
$$\forall x, y \notin R : \eta(x) - \eta(y) \in M_{x-y}\}$$

where $M_x \subseteq U_x$ and $M_{x-y} \subseteq U_{x-y}$ for all x, y. The implied constraints are $\eta(x) - \eta(y) = 0$ for $x, y \in R$ and $\eta(x) - \eta(y) \in M_x$ for $x \notin R$ and $y \in R$. We now have to find the set of all ultraregions that intersect M. Clearly, such set is equal to $U\langle R\rangle$. For $x \notin R$, $y \in R$, let $J_{x-y} = \{J \in \mathcal{I}_{x-y} \mid J \cap M_x \neq \emptyset\}$. Let f be a choice function that assigns to x, y an interval in J_{x-y}, i.e. $f(x, y) \in J_{x-y}$. Let further $O_x \in \mathcal{I}_x$ be the interval containing 0, similarly for O_{x-y}. We then define:

$$U_f = \{\eta \mid \forall x \in R : \eta(x) \in O_x; \forall x \notin R : \eta(x) \in U_x$$
$$\forall x, y \notin R : \eta(x) - \eta(y) \in M_{x-y}$$
$$\forall x, y \in R : \eta(x) - \eta(y) \in O_{x-y}$$
$$\forall x \notin R, y \in R : \eta(x) - \eta(y) \in f(x, y)\}$$

Clearly, every ultraregion intersecting M is of the form U_f for some choice function f as define above. However, as we show in the next example, some U_f do not intersect M.

Example 5.4. *Let us now extend our ultraregion example 4.2 with another clock z with the guard $z \leq 0$. Then $\mathcal{I}_z = \{[0, 0], (0, \infty)\}$, $\mathcal{I}_{x-z} = \{(-\infty, 3), [3, 3], (3, 6), [6, 6], (6, \infty)\}$, and $\mathcal{I}_{y-z} = \{(-\infty, 4), [4, 4], (4, \infty)\}$. Let U be the ultraregion*

$$U = \{(x, y, z) \mid x \in [0, 3], y \in [0, 4], z \in (0, \infty), x - y \in (-1, 2)\}$$

and let $R = \{y, z\}$. Using the zone reset operation and tightening the bounds gives $M = \{(x, y, z) \mid x \in [0, 3], y = z = 0\}$. We then have $J_{x-y} = \{(-1, 2), [2, 2], (2, \infty)\}$ and $J_{x-z} = \{(-\infty, 3), [3, 3]\}$, which gives six possible choice functions and thus six candidate ultraregions. It is easy to verify that two of these candidates, namely those corresponding to the choices $(-1, 2) \in J_{x-y}$, $[3, 3] \in J_{x-z}$ and $[2, 2] \in J_{x-y}$, $[3, 3] \in J_{x-z}$, do not intersect M. The other four candidates together constitute $U\langle R\rangle$.

The implementation of $U\langle R\rangle$ thus works as follows: We first apply the zone reset operation to U and tighten the bounds. We then create the sets of intervals J_{x-y} and iteratively try all choice functions to get all possible candidates. Each candidate is then intersected with M and checked for emptiness. Those who intersect M are then the result.

We have shown the implementation of the ultraregion reset and the ultraregion successor operations. Note that it is possible to pre-compute the ultraregion automaton with respect to a given G. Such a pre-computed automaton will provide an efficient way of obtaining the result of the successor and reset operations for a given ultraregion. Obviously, all other operations required in the construction of the zone-ultraregion automaton are the standard zone operations. These operations can be computed on the fly, thus allowing to employ efficient on-the-fly model checking algorithms.

5.1 Experiments

We have evaluated our method within the parallel and distributed model checker DIVINE [BBH+13]. For the purpose of the evaluation we used several real-time models, namely `2doors.xml`, `bridge.xml`, `fisher.xml`, and `train-gate.xml`, which are distributed with UPPAAL 4.0.13. We have measured the size of ZURA for selected instances of the chosen models with the primary interest in the increase of the state space size with respect to the number of guards used in a specification. Therefore, we work directly with different guard sets instead of CA-LTL formulae. Table 1 shows the growth of zone-ultraregion automaton state space size with the increasing size of the guard set.

We choose guard sets separately for each model as follows, for `2doors.xml` $G_1 = \{x \leq 4\}, G_2 = G_1 \cup \{x \leq 6\}, G_3 = G_2 \cup \{w \leq 0\}$, for `bridge.xml` $G_1 = \{y \leq 10\}, G_2 = G_1 \cup \{y \leq 15\}, G_3 = G_2 \cup \{time \leq 25\}$, for `fisher.xml` $G_1 = \{x \leq 1\}, G_2 = G_1 \cup \{x \leq 2\}, G_3 = G_2 \cup \{x < 2\}$, for `train-gate.xml` $G_1 = \{x \leq 10\}, G_2 = G_1 \cup \{x \leq 5\}, G_3 = G_2 \cup \{x < 15\}$.

6 Expressing UPPAAL Properties in CA-LTL

We now explain how all specification properties of UPPAAL can be expressed in CA-LTL. We refer to the description given in [BDL04]. We first remark that any state formula is a boolean combination of simple guards and location propositions (we consider the value of variables to be the part of a location). We now deal with all kinds of UPPAAL properties.

Table 1: Experimental evaluation of $\mathcal{ZURA}$ state space size

Model	$\|G\| = 0$		$\|G_1\| = 1$		$\|G_2\| = 2$		$\|G_3\| = 3$	
	states	trans.	states	trans.	states	trans.	states	trans.
2doors	189	343	294	508	364	636	482	817
bridge	723	1851	1446	3702	2169	5553	4617	11334
fischer4	1792	4912	12159	29808	16688	38337	32124	72668
fischer5	15142	45262	157623	426256	219435	556705	420875	1063796
fischer6	140716	453328	2174673	6424394	3070446	8536643	5817098	16279518
train-gate3	610	1153	3689	7486	11286	23023	28066	60422
train-gate4	9977	18233	98366	187327	351388	674504	936973	1915545
train-gate5	200776	359031	2479343	4589462	9662204	18112439	27159806	54271266

Reachability Properties The reachability properties in UPPAAL are expressed as $E\Diamond\varphi$ where φ is a state formula. This existential property cannot be expressed as a CA-LTL formula directly, as the semantics of CA-LTL considers all runs. However, we may use the duality of existential and universal quantification and say that a timed automaton *TA* satisfies $E\Diamond\varphi$ iff *TA* does not satisfy $\mathbf{G}\,\neg\varphi$.

Safety Properties The safety properties are expressed as $A\Box\varphi$ or $E\Box\varphi$ where φ is a state formula. The A-version may be expressed directly as $\mathbf{G}\,\varphi$, the E-version indirectly as in the previous case: *TA* satisfies $E\Box\varphi$ iff *TA* does not satisfy $\mathbf{F}\,\neg\varphi$.

Liveness Properties The simple liveness properties are expressed as $A\Diamond\varphi$. These may be expressed directly as $\mathbf{F}\,\varphi$. The *leads to* formula $\varphi \rightsquigarrow \psi$, which has a TCTL equivalent $A\Box(\varphi \Rightarrow A\Diamond\psi)$, can be expressed in CA-LTL as $\mathbf{G}(\varphi \Rightarrow \mathbf{F}\,\psi)$. The reason these formulae are equivalent is similar to the reasoning behind the equivalence of $A\Box(\varphi \Rightarrow A\Diamond\psi)$ and $\mathbf{G}(\varphi \Rightarrow \mathbf{F}\,\psi)$ in standard CTL and LTL.

7 Conclusion and Future Work

Model checking of CA-LTL over timed automata provides system engineers with another powerful formal verification procedure to check for reliability and correctness of time-critical systems. To our best knowledge we are the first to fully describe and implement the process of zone-based LTL model checking over timed automata for a logic that allows clock constraints as atomic propositions. We again recall that the zone-based solutions to the Timed Büchi Automaton Emptiness problem known so far have not provided the solution to the CA-LTL model checking problem as presented in this paper. Our implementation has been done within the parallel and distributed model checker DIVINE which allows us to employ the aggregate power of multiple computational nodes in order to deal with a single model checking task.

We are currently working on several extensions. One of them is to allow atomic propositions used in CA-LTL formulae to be able to refer to the clock value differences. Another extension deals with different zone extrapolation methods and the last one enhances the logic with actions.

References

[ACD90] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking for real-time systems. In *Logic in Computer Science, 1990. LICS'90, Proceedings., Fifth Annual IEEE Symposium on e*, pages 414–425. IEEE, 1990.

[AD94] Rajeev Alur and David L. Dill. A Theory of Timed Automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, January 1996.

[AH94] Rajeev Alur and Thomas A Henzinger. A really temporal logic. *Journal of the ACM (JACM)*, 41(1):181–203, 1994.

[AM04] Rajeev Alur and P. Madhusudan. Decision problems for timed automata: A survey. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *Lecture Notes in Computer Science*, pages 1–24. Springer Berlin Heidelberg, 2004.

[BBH+13] J. Barnat, L. Brim, V. Havel, J. Havlíček, J. Kriho, M. Lenčo, P. Ročkai, V. Štill, and J. Weiser. DiVinE 3.0 – An Explicit-State Model Checker for Multithreaded C & C++ Programs. In *Computer Aided Verification (CAV 2013)*, volume 8044 of *LNCS*, pages 863–868. Springer, 2013.

[BDL+01] Gerd Behrmann, Alexandre David, Kim G. Larsen, Oliver Möller, Paul Pettersson, and Wang Yi. UPPAAL - present and future. In *Proc. of 40*th *IEEE Conference on Decision and Control*. IEEE Computer Society Press, 2001.

[BDL04] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, number 3185 in LNCS, pages 200–236. Springer–Verlag, September 2004.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[Bou04] Patricia Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004.

[BY04] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, pages 87–124. Springer, 2004.

[CGP99] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT press, 1999.

[DD07] Stéphane Demri and Deepak D'Souza. An automata-theoretic approach to constraint LTL. *Information and Computation*, 205(3):380–415, 2007.

[Dil90] David L Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic verification methods for finite state systems*, pages 197–212. Springer, 1990.

[DT98] Conrado Daws and Stavros Tripakis. Model Checking of Real-Time Reachability Properties Using Abstractions. In *Proceedings of the 4th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, TACAS '98, pages 313–329, London, UK, UK, 1998. Springer-Verlag.

[HLP90] Eyal Harel, Orna Lichtenstein, and Amir Pnueli. Explicit clock temporal logic. In *Logic in Computer Science, 1990. LICS'90, Proceedings., Fifth Annual IEEE Symposium on e*, pages 402–413. IEEE, 1990.

[HSW11] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Efficient Emptiness Check for Timed Büchi Automata (Extended version). *CoRR*, abs/1104.1540, 2011.

[Koy90] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):255–299, 1990.

[Li09] Guangyuan Li. Checking Timed Büchi Automata Emptiness Using LU-Abstractions. In *Formal Modeling and Analysis of Timed Systems*, volume 5813 of *LNCS*, pages 228–242. Springer, 2009.

[LOD+13] Alfons Laarman, Mads Chr. Olesen, Andreas Engelbredt Dalsgaard, Kim Guldstrand Larsen, and Jaco van de Pol. Multi-core Emptiness Checking of Timed Büchi Automata Using Inclusion Abstraction. In *Computer Aided Verification (CAV 2013)*, volume 8044 of *Lecture Notes in Computer Science*, pages 968–983. Springer, 2013.

[LT02] Guangyuan Li and Zhisong Tang. Modelling real-time systems with continuous-time temporal logic. In Chris George and Huaikou Miao, editors, *Formal Methods and Software Engineering*, volume 2495 of *Lecture Notes in Computer Science*, pages 231–236. Springer Berlin Heidelberg, 2002.

[Ost89] Jonathan S Ostroff. *Temporal logic for real-time systems*, volume 40. Cambridge Univ Press, 1989.

[Pet99] Paul Pettersson. *Modelling and verification of real-time systems using timed automata: theory and practice*. Citeseer, 1999.

[Tri09a] S. Tripakis. Checking timed Büchi Automata Emptiness on Simulation Graphs. *TOCL*, 10(3), 2009.

[Tri09b] Stavros Tripakis. Checking timed Büchi automata emptiness on simulation graphs. *ACM Transactions on Computational Logic (TOCL)*, 10(3):15, 2009.

[TY01] Stavros Tripakis and Sergio Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, 2001.

[VW86] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings, Symposium on Logic in Computer Science (LICS'86)*, pages 332–344. IEEE Computer Society, 1986.