Removing all silent transitions from timed automata

Cătălin Dima[1] and Ruggero Lanotte[2]

[1] LACL, Université Paris 12, 61 av. du Général de Gaulle, 94010 Créteil Cedex, France
[2] DSCPI, Università dell'Insubria, Via Carloni 78, 22100, Como, Italy

Abstract. We show that all ε-transitions can be removed from timed automata if we allow transitions to be labeled with periodic clock constraints and with periodic clock updates. This utilizes a representation of the reachability relation in timed automata in a generalization of Difference Logic with periodic constraints. We also show that periodic updates are necessary for the removal of ε-transitions.

1 Introduction

Timed automata [1] are a useful tool for modeling real-time systems. A theory of timed automata and timed languages has emerged in the past 15 years, leading more or less to generalizations of the classical results on regular expressions, monadic logics and algebraic characterizations, or to negative results concerning determinization or minimization. The reader is redirected to [2] for a discussion on challenges that remain in order to provide nice generalizations of some important results from classical automata theory.

In this paper we are interested in closing a gap in the theory of timed automata, namely the possibility remove ε-transitions (a.k.a. silent transitions). Previously studied in [6, 10], the problem has received upto now negative results in the classical setting of [1] (see [6]), and in a setting in which "periodic" constraints are allowed to label transitions [10]. Some other results were reported in [9] related to the undecidability of the equivalence problem between ε-free timed automata and general timed automata.

We show that, if we generalize both clock constraints and resets, by allowing periodic clock constraints and periodic updates, then ε-transitions can be removed from a timed automaton. Our periodic updates extend the generalized updates studied in [7, 8].

Our result is based on a technique from [13, 15] for representing reachability relations in timed automata. There, we have proved that the reachability relations constructed in the so-called reset semantics of timed automata are representable by a class of finite automata, the so-called n-automata. Results from [11, 17] could also be used.

We identify here an extension of Difference Logic which is equivalent in expressive power with n-automata, thus identifying also a class of first-order logic properties defining reachability relations. Our extension of Difference Logic contains positive boolean combinations of difference formulas utilizing some generalized form of "ultimate periodicity".

The basic idea in our technique for removing ϵ-transitions is to replace a whole maximal subautomaton $\mathcal{B}$ composed of ϵ-transitions with a finite set of transitions, each one associated with some "entry transition" in $\mathcal{B}$ and an "exit point" from $\mathcal{B}$. The transition is labeled then with a formula which re-encodes the reachability transition defined by $\mathcal{B}$ (with the given entry and exit points) into clock valuation semantics. The re-encoding is necessary as the reachability relation is constructed in terms of reset point semantics. Also the formula involves some generalized forms of updates, specifying that some clocks are to be updated, on the exit point, to some values satisfying some formula in our generalization of Difference Logic.

We also include a proof of the fact that our generalized form of updates are necessary. The proof adapts an example of [10] and does not rely on the assumption that automata work on infinite timed word from [10]. We build a timed automaton recognizing a language for which, for each n, the set of accepted timed words is a "region language", in the sense e.g. of [19, 14]. To recognize such a language with only resets to zero, one would need an unbounded number of clocks.

It should be noted that the region construction, or a zone propagation algorithm would not suffice for removing the ε-transitions. This not only follows from the results of [6, 10], but also as our technique relies essentially on the possibility to express exactly the reachability relation – and it is known that the region construction does not give the possibility for a faithful representation of the reachability construction.

The rest of the paper is organized as follows: the next section recalls the reset time semantics of timed automata and presents our technique for removing ϵ-transitions. The third section gives some intuitions for the technique for removing ε-transitions, and presents our extension of Difference Logic and the generalized form of constraints and updates that are obtained in timed automata after removing ϵ-transitions. The fourth section recalls the results on representing reachability relations with n-automata and gives the proof that our extension of Difference Logic is expressively equivalent with n-automata. The fifth section contains the proof that generalized updates are needed for removing ϵ-transitions. Due to space limitations, the proof of this result can be found in the technical report [16]. We end with a section with conclusions and direction for further work.

2 Timed automata with reset point semantics

Behaviors of timed systems can be modeled by **timed words** over a set of symbols Σ which are finite sequences of nonnegative numbers and symbols from Σ. For example, the sequence $1.3\,a\,1.2\,b$ denotes a behavior in which an *action* a occurs 1.3 time units after the beginning of the observation, and after another 1.1 time units action b occurs. A special case occurs when time elapses after the last action – it is the case of the timed word $3\,a\,1\,b\,1$. We will avoid this for technical reasons, as such behaviors require checking some constraint on a final ε-transition which can never be removed without changing the accepted language of the timed automaton. Hence, we require that each timed word ends with an action name – eventually by inserting a special symbol $\perp$ to signal the end of the timed word. The *untiming* of a timed word is the sequence of actions in it, and denoted $\#(w)$, hence $\#(1.3\,a\,1.2\,b) = ab$. Also the *length* of a timed word, denoted $\ell(w)$, is the sum of time-passage quantities, hence $\ell(1.3\,a\,1.2\,b) = 2.5$. Finally, note that time-passage quantities are additive and ε is the empty timed word, hence $3\,a\,1\,\varepsilon\,1.5\,b = 3\,a\,1\,1.5\,b = 3\,a\,2.5\,b$.

A **timed automaton** [1] is a tuple $\mathcal{A} = (Q, \mathcal{X}, \Sigma, \delta, Q_0, Q_f)$ where Q is a finite set of *locations*, $\mathcal{X}$ is a finite set of *clocks*, Σ is a finite set of *action symbols*, $Q_0, Q_f \subseteq Q$ are sets of *initial*, resp. *final* locations, and δ is a finite set of tuples (*transitions*) $q \xrightarrow{C,z,X} q'$ where $q, q' \in Q$, $X \subseteq \mathcal{X}$, $z \in \Sigma \cup \{\varepsilon\}$ and C is a finite conjunction of *clock constraints* of the form $x \in I$, where $x \in \mathcal{X}$ and $I \subseteq [0, \infty[$ is an interval with integer (or infinite) bounds. A transition $q \xrightarrow{C,\varepsilon,X} q'$ is called an ε*-transition*.

The classical semantics of timed automata works on clock valuations $v : \mathcal{X} \to \mathbb{R}_{\geq 0}$. As usual, for each clock valuation v, subset of clocks $X \subseteq \mathcal{X}$, and real value $u \in \mathbb{R}$, we will denote $v[X := u]$ the clock valuation resulting from updating all clocks in X to u, and $v + u$ the clock valuation resulting by incrementing all clocks with α.

The semantics of a timed automaton is given in terms of a *clock transition system* $\mathcal{C}(\mathcal{A}) = (\mathcal{Q}, \theta, \mathcal{Q}_0, \mathcal{Q}_f)$ where $\mathcal{Q} = Q \times [\mathcal{X} \to \mathbb{R}_{\geq 0}]$, $\mathcal{Q}_0 = Q_0 \times \{\mathbf{0}_{\mathcal{X}}\}$, where $\mathbf{0}_{\mathcal{X}} : \mathcal{X} \to \mathbb{R}_{\geq 0}$ is the zero clock valuation, and $\mathcal{Q}_f = Q_f \times [\mathcal{X} \to \mathbb{R}_{\geq 0}]$ and

$$\theta = \big\{(q, v) \xrightarrow{u} (q, v') \mid v' = v + u\big\} \cup$$
$$\big\{(q, v) \xrightarrow{z} (q', v') \mid \exists q \xrightarrow{C,z,X} q' \in \delta \text{ with } v \models C \text{ and } v' = v[X := 0]\big\}$$

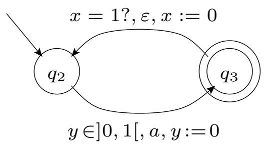
A *trajectory* in $\mathcal{C}(\mathcal{A})$ is a chain $(q^0, v^0) \xrightarrow{\xi_1} (q^1, v^1) \xrightarrow{\xi_2} \ldots \xrightarrow{\xi_k} (q^k, v^k)$ of transitions from θ, with $\xi_i \in \mathbb{R}_{\geq 0} \cup \Sigma \cup \{\varepsilon\}$. The trajectory is *associated with* the timed word $\xi_1 \ldots \xi_k$.

An *accepting trajectory* in $\mathcal{C}(\mathcal{A})$ is a trajectory which starts in $\mathcal{Q}_0$ and ends in $\mathcal{Q}_f$, and *ends with a* Σ*-transition*, that is, a transition labeled with some $a \in \Sigma$. A timed word is *accepted* by a timed automaton if it is associated with an accepting trajectory. This means that any accepted timed word ends with an action symbol and not with a time passage.

The *accepted language* of $\mathcal{A}$ is the set of timed words which are associated with some accepting trajectory in $\mathcal{C}(\mathcal{A})$, and is denoted $L(\mathcal{A})$. Two timed automata are called *equivalent* iff they have the same language.

An example of a timed automaton is provided in Figure 1, which accepts the language $L_{1reg} = \big\{t_1\ a \ldots a\, t_n a \mid \forall 1 \leq i \leq j \leq n, \sum_{i \leq k \leq j} t_k \in]j-i, j-i+1[\,\big\}$.

Fig. 1. An example of a timed automaton.



An alternative semantics can be given to timed automata by speaking rather of the points where the clocks were reset and other points where they are checked, instead of clock values. This is the *reset points semantics*, first used in [5]. The idea is to utilize a variable $\mathbf{t}$ representing current time, and to record, at each moment t and for each clock x_i, the last moment before t when x was reset – denote this by $r(x)$. Then the value of the clock x at t equals $t - r(x)$. Furthermore, time passage in a location amounts to incrementing the "current time" variable, while resetting clock x amounts to the assignment $r[x := t]$.

Formally, if we denote $\mathcal{X}_\mathbf{t} = \mathcal{X} \cup \{\mathbf{t}\}$, the *reset transition system* associated to $\mathcal{A}$ is $\mathcal{T}(\mathcal{A}) = (Q \times [\mathcal{X}_\mathbf{t} \to \mathbb{R}_{\geq 0}], \theta, Q \times \mathbf{0}_{\mathcal{X}_\mathbf{t}}, Q \times [\mathcal{X}_\mathbf{t} : \mathbb{R}_{\geq 0}])$ where

$$\begin{aligned}\theta = &\Big\{(q,r) \xrightarrow{u} (q,r') \mid r' = r[\mathbf{t} := r(\mathbf{t}) + u]\Big\}\\ &\cup \Big\{(q,r) \xrightarrow{z} (q',r') \mid \exists q \xrightarrow{C,z,X} q' \in \delta \text{ such that } r' = r[X := r(\mathbf{t})]\\ &\qquad \text{and if we put } v = r\Big|_{\mathcal{X}} - r(\mathbf{t}) \text{ then } v \models C\Big\}\end{aligned}$$

Here, for a function $f : A \to B$ and subset $C \subseteq A$, $f\Big|_C$ denotes the restriction of f to C.

Transitions of the first type are *time-passage* and those of the second type are *discrete*, and in this case the transition $q \xrightarrow{C,z,X} q'$ is said to be *associated with* the discrete transition $(q,r) \xrightarrow{z} (q',r')$.

Trajectories can be defined then similarly to the clock valuation case, as well as acceptance and accepted language. It is easy to note that the language accepted by a timed automaton, as defined using the clock semantics, is the same as the language accepted by a timed automaton using the reset semantics.

3 Removing a subautomaton with ε-transitions

Suppose we are given two transitions, $\tau = q \xrightarrow{C,z,X} s$ and $\tau' = q' \xrightarrow{C',z',X'} s'$, in a timed automaton $\mathcal{A}$, and some subautomaton $\mathcal{B} = (Q', \mathcal{X}, \emptyset, \delta', \{s\}, \{s'\})$, where $Q' \subseteq Q$ and $\delta' \subseteq \delta$ – hence $\mathcal{B}$ contains only ε-transitions.

We would like to "replace", in $\mathcal{A}$, the subautomaton $\mathcal{B}$, together with the final transition τ', with a single transition $\tau^\mathcal{B}$, labeled with some "generalized" constraint, $\tau^\mathcal{B} = (\overline{s}, C, z', s')$. The transition $\tau^\mathcal{B}$ should take place at the same time where τ' takes place when accepting some timed word. The intuitive construction is depicted in Figure 2.

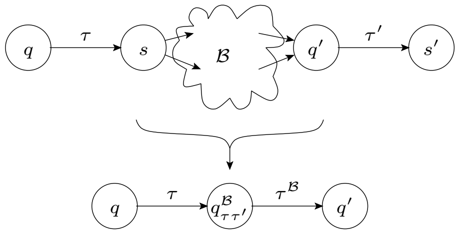


Fig. 2. The scheme of removing one subautomaton composed of only ε-transitions

We seek for a formula $\phi^\mathcal{B}$ on clock valuations, formula which would ensure that each trajectory that takes the two transitions τ and $\tau^\mathcal{B}$ in the modified automaton can be simulated with a trajectory in the original automaton that takes τ, then passes through $\mathcal{B}$, and leaves this subautomaton through τ'. The aim is that this constraint can be defined in some fragment of first-order logic, defining some generalized forms of updates [8].

It appears that the construction of $\phi^\mathcal{B}$ can be done in a simpler way if we focus on the relation between *reset times*, instead of clock values. The reachability relation on reset times defined by τ, τ', and $\mathcal{B}$, denoted $Rs^\mathcal{B}_{\tau\tau'}$, is composed of tuples (r, r') for which there exists some trajectory $\rho = \Big((q_{i-1}, r^{i-1}) \xrightarrow{\zeta_i} (q_i, r^i)\Big)_{1 \leq i \leq N}$, in the reset transition system for $\mathcal{A}$, with $r^0 = r, r^N = r'$ and $\mathcal{T}(\mathcal{A})$, such that the first transition is associated with τ, the last transition is associated with τ', and all intermediary transitions in ρ are associated with ε-transitions in $\mathcal{B}$.

The aim is to build a formula $\psi^{\mathcal{B}}_{\tau\tau'}(\mathbf{r}_1, \ldots, \mathbf{r}_n, \mathbf{t}, \mathbf{r}'_1, \ldots, \mathbf{r}'_n, \mathbf{t}')$ which represents the relation $Rs^{\mathcal{B}}_{\tau\tau'}$, where $\mathbf{r}_i$ would be the variable denoting the last reset time for clock x_i before the moment when τ is taken, $\mathbf{t}$ would be the variable denoting the moment where τ is taken, and $\mathbf{r}'_i$ and $\mathbf{t}'$ would be similarly related with the moment where τ' is taken. (Here we consider that the clocks are numbered $\mathcal{X} = \{x_1, \ldots, x_n\}$.) Using results from [13, 15], we may show that this formula can be constructed in an extension of the Difference Logic: we will show that $\psi^{\mathcal{B}}_{\tau\tau'}$ is a finite disjunction of existential quantifications of conjunctions of formulas of the type $x - y \in D$, where D is an arithmetic expression with interval summation, in the form $D = [a, b[+ k_1\alpha_1 + \ldots + k_p\alpha_p$, with $\alpha_1, \ldots, \alpha_p \in \mathbb{Z}$ being some constants and $k_1, \ldots, k_p$ nonnegative integer variables over which the whole formula may be existentially quantified. In other words, we will be able to write some formula of the type

$$\exists k_1 \ldots \exists k_{n-1} \bigwedge_{i,j} (\mathbf{r}_i - \mathbf{r}_j \in D_{ij}) \wedge (\mathbf{r}'_i - \mathbf{r}_j \in D'_{ij}) \wedge (\mathbf{r}'_i - \mathbf{r}'_j \in D''_{ij}) \wedge$$
$$\bigwedge_i (\mathbf{t} - \mathbf{r}_i \in D_i) \wedge (\mathbf{t} - \mathbf{r}'_i \in D'_i) \wedge (\mathbf{t}' - \mathbf{r}'_i \in D''_i) \wedge (\mathbf{t}' - \mathbf{t} \in D) \quad (1)$$

where $D_{ij}, D'_{ij}, D''_{ij}, D_i, D'_i, D''_i, D$ are "periodic" sets of intervals defined using the variables $k_1, \ldots, k_{n-1}$.

Then we would have to synthesize a formula on *clock values* from this. To do this, we may first observe the following:

- The constraint $\mathbf{r}'_i - \mathbf{r}'_j \in D''_{ij}$ is the same as the constraint $x_i - x_j \in D''_{ij}$, with the clock values referring to the moment when τ' is taken.
- The constraint $\mathbf{r}'_i - \mathbf{t}' \in D''_i$ is the same as the constraint $x_i \in D''_i$, with the clock value of x_i referring to the moment when τ' is taken.
- The constraint $\mathbf{r}_i - \mathbf{r}_j \in D_{ij}$ is the same as the constraint $x_i - x_j \in D_{ij}$, computed at τ', as if the clocks x_i and x_j were not reset during the passage through the automaton $\mathcal{B}$.
- Similarly, the constraint $\mathbf{r}_i - \mathbf{t}' \in D'_i$ is the same as the constraint $x_i \in D'_{ij}$, computed at τ' as if x_i would have been never reset in $\mathcal{B}$.

Modeling the constraint $\mathbf{t}' - \mathbf{t} \in D$ and the rest of the constraints requires the creation of a new clock, x_{time}, which is reset on τ. Then:

- The constraint $\mathbf{t}' - \mathbf{t} \in D$ is the same as the constraint $x_{time} \in D$, checked at τ'.
- The constraint $\mathbf{r}'_i - \mathbf{t} \in D'_i$ is the same as the update $x'_i :\in x_{time} - D'_i$ at $\tau^{\mathcal{B}}$.
- The constraint $\mathbf{r}'_i - \mathbf{r}_j \in D'_{ij}$ is the same as the update $x'_i :\in x_j - D'_{ij}$ at $\tau^{\mathcal{B}}$.

In the next section we prove that the construction of the formulas $\phi^{\mathcal{B}}_{\tau\tau'}$ is always possible. But first, we formalize the above construction, and the extension of Difference Logic which is utilized in our generalized constraints.

Difference logic (DL) is the first-order logic of formulas that bind differences of variables. One possible presentation of DL is through the following syntax:

$$\varphi ::= x - y \in I \mid \varphi \wedge \varphi \mid \exists x \varphi$$

where I is an interval with integer or infinite bounds.

An extension of DL with regular constraints can be defined as follows: a regular constraint is a constraint of the form $\psi : x - y \in I + k_1\alpha_1 + \ldots + k_p\alpha_p$, where k_i are integer-valued variables and α_i are integer constants. Then formulas in this "regular" extension are of the type

$$\varphi ::= \psi \mid \exists k_1 \ldots \exists k_p \varphi \mid \varphi \wedge \varphi \mid \neg\varphi$$

This logic is too strong for our aim: some formulas in this logic cannot specift the reachability relation of some timed automaton. For example, the formula

$$\phi_\exists : \exists k.(x - y = k) \wedge (y - z = k)$$

cannot represent any reachability relation. Intuitively, if $\phi_\exists$ represented the reachability relation, the timed automaton would have to accept $L_{nonreg} = \{a\,n\,b\,n\,c \mid n \in \mathbb{N}\}$, which is clearly not a timed regular language, a simple pumping argument rejecting it.

The logic that we seek for is the **ordered regular difference logic** (ORDL for short), which is the fragment of the above extension of DL defined as follows:

$$\begin{aligned} &\varphi ::= \varphi \vee \varphi \mid [\eta]\psi \\ &\psi ::= \exists k_1 \ldots \exists k_{n-1} \bigwedge\nolimits_{1 \le i < j \le n} \Big(x_j - x_i \in I_{ij} + \sum\nolimits_{l=i}^{j-1} k_l \alpha_l \Big) \end{aligned} \tag{2}$$

(i.e. with n real-valued variables and $n-1$ integer-valued variables), where:

1. $\eta : \{1, \ldots, n\} \to \{1, \ldots, n\}$ is a bijection and $[\eta]\varphi$ is the result of changing, in formula φ, each index i labeling a variable, a constant or an interval into $\eta(i)$. That is, we change x_i into $x_{\eta(i)}$, k_i into $k_{\eta(i)}$, α_i into $\alpha_{\eta(i)}$ and I_{ij} into $I_{\eta(i)\eta(j)}$.
2. $I_{ij} \subseteq [0, \infty[$ is a *bounded* interval with nonnegative integer bounds and α_i is a nonnegative integer constant.

A further restriction applies to the semantics of ORDL formulas, as quantifications are supposed to be only over *nonnegative integers*.

The following property shows that formulas in DL can be expressed in ORDL:

Proposition 1. *For each formula ϕ in DL there exists a formula $\overline{\phi}$ in ORDL which is equivalent with ϕ.*

Proof. The formulas which may pose problems are formulas which contain unbounded intervals. The idea is then to decompose such formulas into a "regular" set of regions.

The formalization is the following: first, each interval which contains negative numbers is split in two, that is, we replace each constraint of the form $x - y \in [-\alpha, \beta]$ with $x - y \in [0, \beta] \vee y - x \in]0, \alpha]$.

Further, each *finite and non-point* interval occurring in ϕ is decomposed into unit-length intervals – that is, each subformula of the form $x - y \in I$ with $\inf I = a$ and $\sup I = b$ is decomposed into a disjunction of the form $\bigvee_{l=a}^{b-1} x - y \in I \cap [l, l+1]$.

Then we bring the formula into DNF, and transform each conjunct into *normal form*, as it is done for Difference-Bound Matrices (DBM) [4, 13, 15]. After these steps, we end with formulas which are obtained, by bijective renamings of indices, from formulas of the type

$$\phi' = \bigwedge_{1 \leq i < j \leq n} x_j - x_i \in I_{ij} \tag{3}$$

where some of I_{ij} are either *infinite* intervals, *unit length* intervals, or *point* intervals.

Remark 1. Recall that a conjunct of the form 3 is in DBM normal form if and only if for each triple of indices i, j, k, $I_{ij} \subseteq I_{ik} + I_{kj}$ [13, 12]. Note then that for all $i < j < k < l$, if I_{il} is finite then I_{jk} is finite too.

Denote $\beta_i = \inf I_{i,i+1}$, and $\alpha_i = 1$ if $I_{i,i+1}$ is infinite and $\alpha_i = 0$ otherwise. Also, for each $i < j$, put $J_{ij} = I_{ij} \cap \Big[0, \sum_{l=i}^{j-1} (\beta_l + 1)\Big]$.

Then ϕ' is equivalent with the formula

$$\phi'' : \exists k_1, \ldots, \exists k_{n-1} \bigwedge_{1 \leq i < j \leq n} \Big(x_j - x_i \in J_{ij} + \sum_{l=i}^{j-1} \alpha_l k_l \Big)$$

To see that, take a tuple $\overline{x}$ that satisfies ϕ', and denote $k_i = \lfloor \overline{x}_{i+1} - \overline{x}_i \rfloor - \beta_i$. Denote $\overline{y}$ the tuple defined by $\overline{y}_i = \overline{x}_i - \sum_{l=1}^{i-1} k_l$ for all $i \leq n$. Then $\overline{y}$ satisfies the formula $\phi'' : \bigwedge_{1 \leq i < j \leq n} \overline{y}_j - \overline{y}_i \in J_{ij}$ □

In the sequel, for a renaming of variables defined by a bijection $\eta : \{1, \ldots, n\} \to \{1, \ldots, n\}$, and two real-valued variables x_i, x_j, we abuse notation by denoting $\eta(x_i) < \eta(x_j)$ instead of $\eta(i) < \eta(j)$.

The class of timed automata which utilize as constraints and updates (in the sense of [8]) formulas from ORDL is the following:

Definition 1. *A* ***timed automaton with generalized constraints/updates*** *is a tuple $\mathcal{A} = (Q, \mathcal{X}, \Sigma, \delta, \delta_0, \delta_f)$ where Q, $\mathcal{X}$, Σ, bear the same names and properties from timed automata, with $\mathcal{X} = \{x_1, \ldots, x_n\}$ and*

1. *δ is a finite set of tuples (*transitions*) $q \xrightarrow{[\eta]\psi, a} r$ where ψ is a formula of the form 2 in ORDL whose real-valued variables are in $\mathcal{X} \cup \mathcal{X}'$, with $\mathcal{X}'$ being a copy of $\mathcal{X}$. Moreover, η is a variable reordering such that $\eta(x) < \eta(x')$ for all $x \in \mathcal{X}$.*
2. *$\delta_0 \subseteq \delta$ is a set of* initial transitions *and $\delta_f \subseteq \delta$ a set of* final transitions.
3. *Initial transitions carry only generalized constraint of the form $\bigwedge_{1 \leq i \leq n} x_i = 0 \wedge x'_i := 0$.*
4. *Initial and final transitions are labeled with non-ε symbols.*

Note that the first condition implies that, in the formulas labeling transitions, if a conjunct of the form $x'_i - x_i \in D_{ii'}$ occurs, then $D_{ii'} \subseteq [0, \infty[$.

The semantics of this extension of timed automata is then given by the clock transition system $\mathcal{C}(\mathcal{A}) = (\mathcal{Q}, \theta, \mathcal{Q}_0, \mathcal{Q}_f)$ where $\mathcal{Q} = Q \times [\mathcal{X} \to \mathbb{R}_{\geq 0}]$, $\mathcal{Q}_0 = Q_0 \times \{\mathbf{0}_{\mathcal{X}}\}$, $\mathcal{Q}_f = Q_f \times [\mathcal{X} \to \mathbb{R}_{\geq 0}]$ and

$$\theta = \big\{ (q, v) \xrightarrow{t} (q, v') \mid v'_i = v_i + t,\ \forall i \in [1 \ldots n] \big\}$$
$$\cup \big\{ (q, v) \xrightarrow{a} (q', v') \mid \exists q \xrightarrow{\varphi, a} q' \in \delta \text{ with } (v, v') \models \varphi, \text{ where } (v, v') \text{ is defined as}$$
$$(v, v') : \mathcal{X} \cup \mathcal{X}' \to \mathbb{R}_{\geq 0}, (v, v')(x) = v(x), (v, v')(x') = v'(x) \big\}$$

Trajectories are then defined similarly to classical timed automata, while acceptance requires a trajectory to *start with an initial transition* and *end with a final transition*.

Note that, at each discrete transition in $\mathcal{C}(\mathcal{A})$, $(v, v') \models \varphi$ simulates the generalized updating part in the following sense: if we consider that a trajectory is constructed "from left to right", and we have constructed the trajectory upto some state (q, v), then the extension of this trajectory past transition $q \xrightarrow{\varphi,z} q'$ is constructed by choosing some values $v' = (v'_1, \ldots, v'_n)$, such that $(v, v') \models \varphi$, and then assigning these values to the respective clocks in $\mathcal{X}$. Of course, if no tuple (v, v') can be constructed which satisfies φ, then the trajectory cannot be extended past the transition $q \xrightarrow{\varphi,z} q'$.

We end this section with the formalization of the construction from Figure 2. Given a timed automaton with generalized constraints/updates $\mathcal{A} = (Q, \mathcal{X}, \Sigma, \delta, Q_0, Q_f)$, suppose that there exists a subautomaton $\mathcal{B} = (Q', \mathcal{X}, \Sigma, \delta', \{\tau\}, \{\tau'\})$, where $Q' \subseteq Q$ and $\delta' \subseteq \delta$ is composed only of ε-transitions. Fix further some state in $\mathcal{B}$, $s \in Q'$ and denote T_s the set of non-ε-transitions of $\mathcal{A}$ that enter s, i.e.

$$T_s = \left\{ q \xrightarrow{\varphi,a} s \in \delta \mid a \in \Sigma \right\}$$

Also take a state in $\mathcal{B}$, $q' \in Q'$ and some non-ε-transition leaving q', $\tau' = q' \xrightarrow{\varphi',b} s'$.

Suppose further that, for each $\tau \in T_s$, there exists a formula in ORDL $\psi^{\mathcal{B}}_{\tau\tau'} = \psi^{\mathcal{B}}_{\tau\tau'}(\mathbf{r}_1, \ldots, \mathbf{r}_n, \mathbf{t}, \mathbf{r}'_1, \ldots, \mathbf{r}'_n, \mathbf{t}')$ which represents the reachability relation $Rs^{\mathcal{B}}_{\tau\tau'}$. We will call this the **ORDL assumption**.

Then, in each constraint $\psi^{\mathcal{B}}_{\tau\tau'}$, apply the following change of variables:

$$\mathbf{r}_i \mapsto \mathbf{t}' - \mathbf{x}_i,\ \mathbf{r}'_i \mapsto \mathbf{t}' - \mathbf{x}'_i,\ \mathbf{t} \mapsto \mathbf{t}' - x'_{time}$$

Hence, the resulting constraint is built as follows:

$$\begin{aligned} &\phi^{\mathcal{B}}_{\tau\tau'}(\mathbf{x}_1, \ldots, \mathbf{x}_n, x_{time}, \mathbf{x}'_1, \ldots, \mathbf{x}'_n, x'_{time}) = \\ &\qquad \psi^{\mathcal{B}}_{\tau\tau'}(\mathbf{t}' - \mathbf{x}_1, \ldots, \mathbf{t}' - \mathbf{x}_n, \mathbf{t}' - x'_{time}, \mathbf{t}' - \mathbf{x}'_1, \ldots, \mathbf{t}' - \mathbf{x}'_n, \mathbf{t}') \end{aligned} \tag{4}$$

Remark 2. Note that in $\phi^{\mathcal{B}}_{\tau\tau'}$ the variable $\mathbf{t}'$ does not occur. This is because all constraints are difference constraints, and therefore $\mathbf{t}'$ gets canceled everywhere.

Consider then the timed automaton $\overline{\mathcal{A}} = (Q \cup \{q^{\mathcal{B}}_{\tau\tau'}\}, \mathcal{X}, \Sigma, \overline{\delta}, Q_0, Q_f)$, in which (recall that τ' ends in s'):

$$\begin{aligned} \overline{\delta} = \big(\delta \setminus T_s\big) &\cup \left\{ q \xrightarrow{\varphi \wedge x'_{time}:=0,a} s^{\mathcal{B}}_{\tau\tau'}, s^{\mathcal{B}}_{\tau\tau'} \xrightarrow{\phi^{\mathcal{B}}_{\tau\tau'},b} s' \mid \tau = q \xrightarrow{\varphi,a} s \in T_s, \tau' \notin T_s \right\} \\ &\cup \left\{ q \xrightarrow{\varphi \wedge x'_{time}:=0,a} s^{\mathcal{B}}_{\tau\tau'}, s^{\mathcal{B}}_{\tau\tau'} \xrightarrow{\phi^{\mathcal{B}}_{\tau\tau'},b} s^{\mathcal{B}}_{\tau\tau'} \mid \tau = q \xrightarrow{\varphi,a} s \in T_s, \tau' \in T_s \right\} \end{aligned}$$

Lemma 1. *$\overline{\mathcal{A}}$ is equivalent with $\mathcal{A}$.*

The proof of this lemma follows straightforwardly once we prove the ORDL assumption.

Note that this lemma removes one "entry point" in the subautomaton $\mathcal{B}$. Therefore, if $\mathcal{B}$ is *maximal*, in the sense that all transitions connecting $\mathcal{B}$ with the rest of $\mathcal{A}$ are non-ε

transitions, then it can be applied iteratively to all the "entry points" in the subautomaton $\mathcal{B}$, until $\mathcal{B}$ becomes unreachable from the input states of the resulting timed automaton $\overline{\mathcal{A}}$. And further, iteratively, to any other maximal subautomaton composed of only ε-transitions. Hence,

Theorem 1. *Any timed automaton with generalized constraints/updates is equivalent with a timed automaton with generalized constraints/updates and with no ε-transition.*

An example of the application of this procedure for the timed automaton from [10] is given in the technical report [16].

In the following section we prove that the ORDL assumption in the preamble of Lemma 1 holds. To this end, we recall some of the results proved previously on the representation of reachability relations.

4 Representing reachability relations on reset times in ORDL

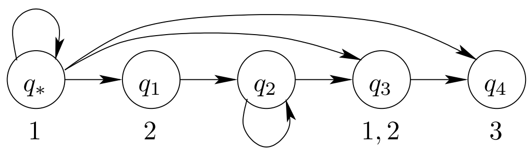
In [13, 15], we have given a characterization of relations of the type $Rs_{\tau\tau'}$. We recall here briefly these characterizations, and then prove the ORDL assumption.

Definition 2. *An n-**automaton** is a tuple $\mathcal{N} = (Q, \delta, Q_*, Q_1, \ldots, Q_n)$ in which Q is a finite set of* states*, $\delta \subseteq Q \times Q$ is a set of* transitions*, and for each $1 \leq i \leq n$, Q_i is the* accepting component for index i*; its elements are called* accepting states for index i*. Finally, $Q_* \subseteq Q$ is the set of* initial states *and we assume that for all $q_* \in Q_*$, $(q_*, q_*) \in \delta$ and $(q_*, q) \in \delta$ for each $q \in Q_i$ and each $1 \leq i \leq n$.*

An n-automaton accepts a tuple $\overline{a} = (a_1, \ldots, a_n) \in \mathbb{N}^n$ if there exists some run starting in Q_* for which a_i is some index, along the run, where the run passes through some state in Q_i. Or, in other words, we put all the components of $\overline{a}$ on an axis and let the automaton "parse" this axis, unit after unit, signaling then it passes through some set Q_i, for some $1 \leq i \leq n$.

More formally, **run** is a sequence of states $\rho = (q_j)_{j\in\{0,\ldots,k\}}$ connected by transitions, i.e., $(q_{j-1}, q_j) \in \delta$ for all $j \in \{1, \ldots, k\}$, and with $q_0 \in Q_*$. The run is **accepting** if for each $i \in \{1, \ldots, n\}$ there exists some $j \in \{1, \ldots, k\}$ such that $q_j \in Q_i$. Given $\overline{a} = (a_1, \ldots a_n) \in \mathbb{N}^n$ we say that $\overline{a}$ is **accepted by** ρ iff $q_{a_i} \in Q_i$ for all $i \in \{1, \ldots, n\}$. The **accepted language** of $\mathcal{N}$ is the set of points in $\mathbb{N}^n$ accepted by $\mathcal{N}$.

For example, consider the following 3-automaton $\mathcal{N}_0$



in which $Q_1 = \{q_*, q_3\}$, $Q_2 = \{q_1, q_3\}$, $Q_3 = \{q_4\}$ as suggested by the indices below the respective states. This automaton accepts the 3-tuple $(1, 4, 5)$: the accepting run for it is $q_* \to q_* \to q_1 \to q_2 \to q_3 \to q_4$. The same run accepts $(4, 1, 5)$ too.

These automata represent only tuples of integers, with integers represented in unary. They can be converted to representing sets of real numbers which occur from reachability relations in timed automata with the aid of *regions*.

An n-region is a nonempty subset R of $\mathbb{R}^n_{\geq 0}$ which can be represented with the aid of a constraint of the form $\bigwedge_{1\leq i\leq n} x_i - x_j \in I_{ij}$, where I_{ij} is either a point interval or an open unit interval $]a, a+1[$ with $a \in \mathbb{Z}$. We will abuse notation and utilize R both for the region and for its representing constraint. We will denote in the sequel the set of regions in $\mathbb{R}^n_{\geq 0}$ by Reg_n.

Usually, extra constraints of the form $x_i - x_0 \in I_i$ occur as the part of the conjunct, with the convention that $x_0 = 0$. In this section we will not consider this type of regions, as we are interested in representing only difference constraints and not one-variable constraints – recall that the reset-time semantics of transitions in a timed automaton does not involve one-variable constraints.

We will recall our representation of (some) sets of real numbers by n-automata on an example, and then give the general framework. Consider the region

$$R : x = y \wedge x' - y' \in]2,3[\wedge y' - x \in]1,2[\wedge x' - x \in]3,4[$$

We will represent this region with any of the points $d^R = (x = k, y = k, x' = k+4, y' = k+2)$ with $k \in \mathbb{N}$ – these are the integer points on one of the edges of the polyhedron $R \cap (x = 0)$,

But this edge is also an edge in 15 other polyhedra, so, when representing R, the set of points d needs to be accompanied by some extra information, identifying R among all polyhedra which have d as an edge. This information is a matrix whose elements are relational symbols in the set $\{<, =, >\}$. The idea is that we associate to each pair of variables x_1, x_2, the relational symbol which relates the difference $x_1 - x_2$ with the difference $d_1 - d_2$ between the respective components of d.

In our case, in R we have $x' - y' \in]2,3[$ and in d we have $d_{x'} - d_{y'} = 2$, which means that $x' - y' > d_{x'} - d_{y'} = 2$, which will be the relational symbol corresponding to row x' and column y' in the matrix associated with d is $>$.

Hence, one of our representations of the region R is the tuple (d, M) with

$$d = (x = k, y = k, \\ x' = k+4, y' = k+2),\ k \in \mathbb{N}, \qquad M = \begin{pmatrix} & | & x & y & x' & y' \\ \hline x & | & `=` & `=` & `>` & `>` \\ y & | & `=` & `=` & `>` & `>` \\ x' & | & `<` & `<` & `=` & `>` \\ y' & | & `<` & `<` & `<` & `=` \end{pmatrix}$$

In general, we represent a region $R = \bigwedge_{1\leq i\leq n} x_i - x_j \in I_{ij}$ with a pair (d, M) with:

1. $d = (d_1, \ldots, d_n) \in \mathbb{R}^n_{\geq 0}$ is one of the vertices of the polyhedron $cl(R) \cap \mathbb{R}^n_{\geq 0}$ – here $cl(Z)$ represents the closure of a set of reals. Note that d is the "first" member of the regular set of points $d = (d_1 + k, \ldots, d_n + k)$ with $k \in \mathbb{N}$.
2. M is the matrix of relational symbols in $\{<, =, >\}$, for which
   - M_{ij} is ' $<$ ' if $x_i - x_j < d_i - d_j$,
   - M_{ij} is ' $=$ ' if $x_i - x_j = d_i - d_j$,
   - M_{ij} is ' $>$ ' if $x_i - x_j > d_i - d_j$.

It should be noted that not all matrices of relational symbols are consistent – see [13, 15] for details. In the sequel we denote Γ_n the set of matrices of relational symbols (which introduce consistent constraints).

Given a pair (d, M) like above, the reconstruction of the region represented by (d, M) is straightforward:

$$R_{ij} = \begin{cases} \{d_{ij}\} & \text{iff } M_{ij} \text{ is `='} \\ ]d_{ij}-1, d_{ij}[ & \text{iff } M_{ij} \text{ is `<'} \\ ]d_{ij}, d_{ij}+1[& \text{iff } M_{ij} \text{ is `>'}. \end{cases} \tag{5}$$

We denote $[d, M]$ the region R constructed as above, and if D is a set of n-dimensional points with integer coordinates, we denote $[D, M] = \bigcup\{[d, M] \mid d \in D\}$.

So what remains to do is to extend n-automata such that they accept tuples of the form (d, M) consisting of a point in $\mathbb{R}^n_{\geq 0}$ and a matrix of relational symbols. This gives the following definition:

Definition 3. *An n-**region automaton** is a tuple $\mathcal{R} = (Q, \delta, Q_*, Q_1, \dots Q_n, \lambda)$ where Q, δ, Q_* and $Q_1, \dots, Q_n$ bear the same meaning and properties as in n-automata, while $\lambda : Q \to \Gamma_n$ is a labeling function, associating with each state a matrix of relational symbols. Additionally, it is required that if $(q, q') \in \delta$ then $\lambda(q) = \lambda(q')$, and that, for all $q_* \in Q_*$, $(q_*, q_*) \in \delta$.*

The notions of run and acceptance are defined as in n-automata. Observe that, by definition, all the states in a run must be labeled with the same n-relation. An accepting run $\rho = (q_j)_{0 \leq j \leq k}$ **accepts** a pair (d, M) consisting of a tuple $d \in \mathbb{R}^n_{\geq 0}$ and a matrix of relational symbols $M \in \Gamma_n$ if d is accepted by the underlying n-automaton and $\lambda(q_j) = M$ for all $0 \leq j \leq k$

We then say that an n-region automaton $\mathcal{R}$ **represents** some relation on real numbers $R \subseteq \mathbb{R}^n_{\geq 0}$ if

$$R = \bigcup \big\{ [d, M] \mid (d, M) \text{ accepted by } \mathcal{R} \big\}$$

We also denote $\mathsf{Reg}(\mathcal{R})$ the union of the regions represented by the n-region automaton.

In [13, 15], we have proved the following property:

Theorem 2. *For any timed automaton $\mathcal{A}$, any pair of transitions τ, τ' and any sub-automaton $\mathcal{B}$, the reachability relation $Rs^{\mathcal{B}}_{\tau\tau'}$ is representable with a $2n+2$-region automaton, where n is the number of clocks in the given timed automaton.*

The following result ensures that the ORDL assumption in the preamble of Lemma 1 holds, hence completing the proof of Theorem 1:

Theorem 3. *Any relation which is representable with some n-region automaton is expressible in the logic ORDL, and the reverse also holds.*

Proof. For the direct proof, take $\mathcal{R} = (Q, \delta, Q_*, Q_1, \dots Q_n, \lambda)$. We will show that for each matrix of relational symbols $M \in \Gamma_n$ and each reordering η of the first n integers – i.e. bijection $\eta : \{1, \dots, n\} \to \{1, \dots, n\}$ – the set of regions represented by accepting runs labeled M and passing through the accepting indices in the order given by η can be represented in ORDL.

First, some notation: for a bijection η as above and a tuple of integers $d \in \mathbb{N}^n$, by abusing notation, we denote $\eta(d)$ the tuple $\eta(d) = (d_{\eta(1)}, \dots, d_{\eta(n)})$. On the other hand, we say that d *respects* η if whenever $d_i < d_j$ we must also have $\eta(i) < \eta(j)$.

Denote then

$$D_{\eta,M} = \big\{ d \in \mathbb{N}^m \mid (\varphi(d), M) \in L(\mathcal{R}) \text{ and } d \text{ respects } \eta \big\}$$

Remark 3. We may prove that there exist some integer constants $\alpha_i, \beta_i \in \mathbb{N}$ for all $0 \le i \le n-1$ such that

$$D_{\eta,M} = D_0 \cup D^{reg}_{\eta,M} \text{ where } D_0 \text{ is a finite set of regions and} \tag{6}$$

$$D^{reg}_{\eta,M} = \big\{ (k_0, k_0 + k_1\alpha_1 + \beta_1, \ldots, k_0 + k_1\alpha_1 + \beta_1 + \ldots + k_{n-1}\alpha_{n-1} + \beta_{n-1}) \mid k_0, k_1, \ldots, k_{n-1} \in \mathbb{N} \big\} \tag{7}$$

Observe that this means that $\forall i \exists \alpha_i \in \mathbb{N} \exists \beta_i \in \mathbb{N} \forall d \in D^{reg}_{\eta,M} \exists k_i \in \mathbb{N}$ with $d_i - d_{i-1} = k_i\alpha_i + \beta_i$.

This remark follows as a corollary of the well-known fact that the set of lengths of words accepted by a finite automaton is an ultimately periodic sequence. Note first that, by definition, $(q_*, q_*) \in \delta$ for any $q_* \in Q_*$, and $(q_*, q) \in \delta$ for any $q \in Q$, and therefore the lengths of words connecting states in Q_* with any other states form a periodic sequence with period 1 – this is the reason why, in $D^{reg}_{\eta,M}$, $d_1 = k_0$ for some $k_0 \in \mathbb{N}$.

Then we apply this ultimately-periodic result to the finite automaton $\mathcal{R}_1$ which copies the transition relation of $\mathcal{R}$ but with initial states Q_1 and final states Q_2, and get constants α_1 and β_1 such that, in $D^{reg}_{\eta,M}$, $d_2 - d_1 = k_1\alpha_1 + \beta_1$ for some $k_1 \in \mathbb{N}$. Then we apply this result further to $\mathcal{R}_2$ which is the finite automaton resulting from $\mathcal{R}$ with initial states Q_2 and final states Q_3, to get α_2 and β_2, and so on.

Let us then further observe that, in 7 above, $[D_0, M]$ can be easily expressed in DL, since it's a finite disjunction of constraints of the form $\bigwedge_{1 \le i < j \le n} (x_j - x_i \in R_{ij})$, where R_{ij} are the intervals defined in Identity 5 above.

Denote $D^{reg}_{\eta,M}$ the "regular" part of $D_{\eta,M}$ from 7. Then $D^{reg}_{\eta,M}$ can be expressed in ORDL as follows: denote first d_{reg} the tuple

$$d_{reg} = (\beta_0, \beta_0 + \beta_1, \ldots, \beta_0 + \beta_1 + \ldots + \beta_{n-1})$$

Denote I_{ij} the intervals defined as in Identity 5 by the tuple d_{reg}, i.e. $I_{ij} = \{d^{ij}_{reg}\}$ if M_{ij} is '$=$', $I_{ij} =]d^{ij}_{reg} - 1, d^{ij}_{reg}[$ if M_{ij} is '$<$' and $I_{ij} =]d^{ij}_{reg}, d^{ij}_{reg} + 1[$ if M_{ij} is '$>$'.

Then the following constraint in ORDL represents $D^{reg}_{\varphi,M}$:

$$\Phi : \exists k_1 \exists k_2 \ldots \exists k_{n-1} \bigwedge_{1 \le i < j \le n} \Big(x_j - x_i \in I_{ij} + \sum_{l=i}^{j} k_l \alpha_l \Big) \tag{8}$$

It remains to apply the change of variables η to formula Φ. Since $\mathsf{Reg}(\mathcal{R})$ is a finite union of sets of the form $[D_{\eta,M}, M]$, this ends the proof of the direct implication.

For the reverse implication, consider some formula Φ in the form 8 as above. Consider some decomposition into regions of the n-dimensional set P_0 defined by the formula:

$$\Phi_0 : \bigwedge_{1 \le i < j \le n-1} \big(x_j - x_i \in I_{ij} \big)$$

Observe that, if we denote $\phi_R = \bigwedge_{1\leq i<j\leq n-1}(x_j - x_i) \in J_{ij}^R$ the formula defining some region R, then Φ is equivalent with the following formula:

$$\bigvee_{R\subseteq P_0} \exists k_1 \ldots \exists k_{n-1} \bigwedge_{1\leq i<j\leq n-1} \Big(x_j - x_i \in J_{ij}^R + \sum_{l=i}^{j} k_l \alpha_l \Big)$$

Since by definition, in any ORDL formula each I_{ij} is bounded, hence there exist only finitely many distinct regions $R \subseteq P_0$, the above formula is an ORDL formula.

Therefore we only need to treat the case where Φ_0 defines an n-dimensional region. So then for each representation (d, M) of the region Φ_0 we need to build an n-automaton for the set of tuples

$$D = \big\{ (d_1 + k_0, d_2 + k_0 + k_1\alpha_1, \ldots, d_n + k_0 + \sum\nolimits_{l=1}^{n-1} k_l\alpha_l) \mid k_0, \ldots, k_{n-1} \in \mathbb{N} \big\}$$

which is intuitively drawn in Figure 3.

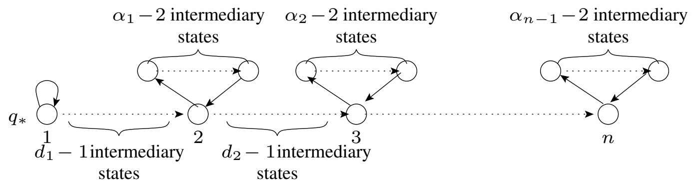


Fig. 3. Representing the set of tuples D with an n-automaton.

In Figure 3, the number of *arrows* between a state labeled i and a state labeled $i + 1$ must equal d_i. In some cases we might have $d_i = 0$ – then we put an ε-transition between the state labeled i and the state labeled $i+1$. In [13, 15, 12] we have shown that n-automata with ε-transitions are equivalent with n-automata without ε-transitions. On the other hand, if $\alpha_i = 0$ then we do not draw any circuit on the state labeled $i-1$. This ends the proof of the reverse implication of Theorem 3. □

5 Impossibility of removing ε-transitions without generalized updates

In this section we prove that the ε-transitions cannot be removed unless utilizing the generalized constraints and updates we presented in the third section. To this end, we adapt an example of Choffrut & Goldwurm form [10], where it is proved that ε-transitions cannot be removed by using “periodic” constraints. The proof in [10] relies essentially on the hypothesis that timed automata of [10] work on infinite timed words. We show here that this assumption is not necessary, as with ε-transitions we may define a “very thin” language (in the sense of [3]), which cannot be defined without any ε-transition.

The working example is from Figure 1; recall that the language of that automaton is $L_{1reg} = \big\{ t_1\ a \ldots a\, t_n a \mid \forall 1 \leq i \leq j \leq n, \sum_{i\leq k\leq j} t_k \in]j-i, j-i+1[\, \big\}$.

Let us denote, for each timed word $w = t_1\ a \dots a\, t_n a \in L_{1reg}$, by p_w the point $p_w = (t_1, t_1 + t_2, \dots, t_1 + t_2 + \dots + t_n) \in \mathbb{R}^n_{\geq 0}$. Also denote R_w the n-dimensional region to which p_w belongs.

Remark 4. For any $n \in \mathbb{N}$ and any timed word $w \in L_{1reg}$, R_w is characterized[1] by the following constraint:

$$\phi : x_j - x_i \in]j-i-1, j-i[\text{ with } x_0 = 0$$

In other words, for each $n \in \mathbb{N}$, $L_{1reg} \cap \{w \mid \ell(w) = n\}$ consists of timed words w whose associated n-dimensional points p_w all lie in a single region R_w.

Given a timed automaton with generalized constraints/updates, we say that a transition τ *resets all clocks to 0* if, for any state transition in $\mathcal{C}(\mathcal{A})$, $(q, v) \to (q', v')$, which is associated with τ, we have that if $v'(x) \neq v(x)$ then $v'(x) = 0$.

Proposition 2. *The language L_{1reg} cannot be accepted by any timed automaton with generalized constraints, with all resets to 0 and without ε-transitions.*

Proof. The proof idea is that, in any timed automaton (even with generalized constraints), if all clock updates *are resets to zero* and occur only at the times where the action symbols occur, then there exist two timed words $w_1, w_2 \in L(\mathcal{A})$ with the same untiming a^N, and such that the regions R_{w_1} and R_{w_2} are distinct. It would then follow that $R^n_{L(\mathcal{A})}$ cannot be a single region, which is the case for L_{1reg}.

The main ingredient in this proof is the *normal form* of Difference-Bound Matrices (DBM) [4, 13, 15], which is a procedure for checking emptiness for constraints written in DL. We utilize Proposition 6 from [15], which basically says that, given a formula which results from this consistency checking procedure, and if we take any value in some of the intervals occurring in the DBM, then this value can be "completed" to a solution of the constraint. Then, the assumption that L_{1reg} is accepted only with resets to 0 would imply that there exists some run in the timed automaton which gives rise to a constraint in normal form in which some of the intervals occurring in the matrix is an interval of length at least two – which will imply that two timed words with distinct regions R_{w_1} and R_{w_2} are accepted by that run.

The complete proof can be found in the technical report [16].

6 Conclusions

We have proved that ε-transitions can be removed timed automata if one allows a generalized form of constraints which include also generalized updates to clock values. The result relies on ORDL, a generalization of Difference Logic which captures exactly the reachability relations defined in timed automata.

As our extension of Difference Logic is quite restrictive in syntax, we would be interested in results on checking whether "periodic" DL formulas are expressible in ORDL. Another path for research could be to generalize forward and backward propagation algorithms to timed automata using generalized constraints/updates.

[1] Here we employ the usual notion of region, involving also the constraint $x_0 = 0$.

References

1. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
2. E. Asarin. Challenges in timed languages. *Bulletin of EATCS*, 83, 2004.
3. E. Asarin and A. Degorre. Volume and entropy of regular timed languages. HAL Archive no. *hal-00369812*, 2009.
4. R. Bellmann. *Dynamic Programming*. Princeton University Press, 1957.
5. J. Bengtsson, B. Jonsson, J. Lilius, and Wang Yi. Partial order reductions for timed systems. In *Proceedings of CONCUR'98*, volume 1466 of *LNCS*, pages 485–500, 1998.
6. B. Bérard, V. Diekert, P. Gastin, and A. Petit. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36:145–182, 1998.
7. P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Are timed automata updatable? In *Proceedings of CAV'2000*, volume 1855 of *LNCS*, pages 464–479, 2000.
8. P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Expressiveness of updatable timed automata. In *Proceedings of MFCS'2000*, volume 1893 of *LNCS*, pages 232–242, 2000.
9. P. Bouyer, S. Haddad, and P.-A. Reynier. Undecidability results for timed automata with silent transitions. *Fundamenta Informaticae*, 92:1–25, 2009.
10. Ch. Choffrut and M. Goldwurm. Timed automata with periodic clock constraints. *Journal of Automata, Languages and Combinatorics*, 5:371–404, 2000.
11. H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In *Proceedings of CONCUR'99*, volume 1664 of *LNCS*, pages 242–257, 1999.
12. C. Dima. *An algebraic theory of real-time formal languages*. PhD thesis, Université Joseph Fourier Grenoble, France, 2001.
13. C. Dima. Computing reachability relations in timed automata. In *Proceedings of the 17th IEEE Symposium on Logic in Computer Science (LICS'02)*, pages 177–186, 2002.
14. C. Dima. A nonarchimedian discretization for timed languages. In *Proceedings of FORMATS'03*, volume 2791 of *LNCS*, pages 161–181, 2003.
15. C. Dima. A class of automata for computing reachability relations in timed systems. In E. Clarke and F. L. ¸Tiplea, editors, *Proceedings of the NATO Advanced Research Workshop on Verification of Infinite State Systems with Applications to Security (VISSAS'05)*, NATO ARW Series. 2005. to appear.
16. C. Dima and R. Lanotte. Removing all silent transitions from timed automata. Technical Report TR-LACL-2009-11, LACL, 2009.
17. Y. Jurski. *Expression de la relation binaire d'accessibilité pour les automates à compteurs plats et les automates temporisés*. PhD thesis, École Normale Supérieure de Cachan, France, 1999.
18. M. Mahfoudh, P. Niebert, E. Asarin, and O. Maler. A satisfiability checker for difference logic. In *Proceedings of SAT 2002*, pages 222–230, 2002.
19. J. Ouaknine and J. Worrell. Revisiting digitization, robustness, and decidability for timed automata. In *Proceedings of LICS'03*, pages 198–207. IEEE Computer Society Press, 2003.