Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 345 (2005) 27–59
Theoretical Computer Science
www.elsevier.com/locate/tcs

A partial order semantics approach to the clock explosion problem of timed automata☆

D. Lugiez*, P. Niebert, S. Zennou

Laboratoire d'Informatique Fondamentale (LIF) de Marseille, Université de Provence – CMI, 39, rue Joliot-Curie / F-13453 Marseille Cedex 13, France

Abstract

We present a new approach to the symbolic model checking of timed automata based on a partial order semantics. It relies on *event zones* that use vectors of event occurrences instead of *clock zones* that use vectors of clock values grouped in polyhedral clock constraints. We provide a description of the different congruences that arise when we consider an independence relation in a timed framework. We introduce a new abstraction, called *catchup* equivalence which is defined on event zones and which can be seen as an implementation of one of the (more abstract) previous congruences. This formal language approach helps clarifying what the issues are and which properties abstractions should have. The catchup equivalence yields an algorithm to check emptiness which has the same complexity bound in the worst case as the algorithm to test emptiness in the classical semantics of timed automata. Our approach works for the class of timed automata proposed by Alur–Dill, except for state invariants (an extension including state invariants is discussed informally). First experiments show that the approach is promising and may yield very significant improvements.

Keywords: Algorithms; Verification; Timed automata; Partial order

☆ This work was supported by the IST project AMETIST, contract IST-2001-35304. http://ametist.cs.utwente.nl.

* Corresponding author.
E-mail addresses: lugiez@cmi.univ-mrs.fr (D. Lugiez), niebert@cmi.univ-mrs.fr (P. Niebert), zennou@cmi.univ-mrs.fr (S. Zennou).

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.07.023

1. Introduction

Timed automata [3] are a powerful tool for the modeling and the analysis of timed systems. They extend classical automata by *clocks*, continuous variables "measuring" the flow of time. A state of a timed automaton is a combination of its discrete control location and the *clock values* taken from the real domain. While the resulting state space is infinite, *clock constraints* have been introduced to reduce the state spaces to a finite set of equivalence classes, thus yielding a finite (although often huge) symbolic state graph on which reachability and some other verification problems can be resolved.

While the theory, algorithms [17,18] and tools [5,26] for timed automata represent a considerable achievement (and indeed impressive industrial applications have been treated), the combinatorial explosion particular to this kind of modeling and analysis—sometimes referred to as "clock explosion" [1] (at the same time similar to and different from classical "state explosion")—remains a challenge for research and practice. Despite the theoretical limits (for a PSPACE complete problem), great effort has been invested into the optimization of the symbolic approach (see e.g. [4,6,12,13]).

Among the attempts to improve the efficiency of analysis algorithms, one line of research has tried to transfer "partial order reduction methods", a set of techniques known to give good reductions (and thus allowing to handle bigger problems) for discrete systems [16,20,22,23], to the timed setting. Partial order methods basically try to avoid redundant research by exploiting knowledge about the structure of the reachability graph, in particular *independence* of pairs of transitions of loosely related parts of a complex system. Such pairs a and b commute, i.e. a state s allowing a sequence ab of transitions to state s' also allows ba and this sequence also leads to the same state s'.

However, this kind of commutation is easily lost in classical symbolic analysis algorithms for timed automata, which represent sets of possible clock values by symbolic states: consider two "independent" actions a resetting clock $X := 0$, and b resetting clock $Y := 0$. Executing a first and then b means that afterwards (time may have elapsed) $X \geqslant Y$ whereas executing b first and then a implies that afterwards $X \leqslant Y$. The result of this is that in the algorithms used in tools like UppAal and Kronos, ab and ba lead to *different*, in fact incomparable symbolic states.

1.1. Preceding work and state of the art

In previous work, we find two main approaches for partial order methods in timed systems.

The first kind [10,21] analyzes clock constraints and clock reset between two actions to know in which cases these actions commute. Partial orders are then applied on these (few) cases.

The second kind relaxes constraints added between actions when they occur. In the classical semantics, actions that occur are totally ordered according to their order of occurrence. In these semantics, actions that occur are ordered only if they are causally related in a

[1] Personal communication by Thomas Henzinger: this term was introduced by him informally in presentations, not in writing. But it has become folklore in the timed automata community.

network, like a time Petri net [25], a network of timed automata [8] or a TEL structure [7]. These relaxed semantics reestablish commutations between actions that are done by distinct processes and then reduce the generated state space. For all these methods an abstraction has to be done on the symbolic state space to identify states in order to ensure finiteness. In addition, [25] combines this relaxed semantics with the combination of the stubborn set method [24], a partial order reduction method that explores for untimed systems that explore at each discrete state a subset of transitions.

Whereas [25] and [7] maintain a matrix of constraints between possible time of transition occurrences to know if a transition enabled can be fired before the other enabled ones, [8] do not check for these conditions: they assume that each automaton in the network has its own local time and this time is only synchronized in a common transition. This implies that every pair of actions that commute in an untimed framework do commute in the timed context. Consequently, a partial order reduction for discrete systems like ample set [22] can be directly applied to make space savings. The major restriction of [8] is that the automata in the network may only use local clocks, not shared or global clocks. Secondly, the index of the abstraction used produces significantly more symbolic states than that of classical zone automata and that the benefit of partial order reductions is often insufficient to compensate the blowup by the weaker abstraction. [2] The POSET approach [7] is based on safe Time Petri nets and models zones based on the generation times of yet unconsumed tokens and thereby avoids relating these times for independent transitions. The Time Petri nets in question can be understood as a subclass of timed automata avoiding certain difficulties of the Alur–Dill framework. The POSET approach has turned out to be very successful for circuit applications.

1.2. This work

Our work falls into the second class of partial order approaches to timed automata, but with a shift in goals: rather than aiming at the transfer of partial order reductions our aim is to reduce the number of explored symbolic states due to a more abstract semantics.

Our first contribution is a new framework for symbolic state exploration of timed automata based on *event zones*. Event zones consider sets of vectors of time stamps rather than clock values. We give conditions for the independence of transitions that include their use of clocks (conditions, updates). Event zones can be understood as a common generalization of the POSET approach [7] (in leaving the restricted class of safe Time Petri nets) and the "local time" approach [8] (in allowing shared clocks). We cover the full class of Alur–Dill timed automata except for state invariants. However, we give an informal discussion on an extension with such state invariants in Section 7.

Event zones allow us to define a symbolic automaton for the language of feasible executions of a timed automaton (up to commutation). However, such symbolic automata are unavoidably infinite (see Proposition 12). Our second contribution consists of a language theoretic framework for emptiness checking of these symbolic automata despite the fact that they have infinitely many states. We do so by introducing a number of preorders and equivalences related to the Myhill–Nerode right congruence of classical automata. One such

[2] Personal communication by Bengt Johnnson.

preorder, which we call "catchup simulation", is proven to be of finite index and to preserve certain paths that are on the whole sufficient to cut branches of the symbolic automaton while preserving non-emptiness. Then we use this framework for an emptiness checking algorithm, which we prove correct and which we have actually implemented and compared to the classical clock zone approach, with very satisfying results.

An important aspect of our abstraction is that it is closely related to standard clock zone abstractions, preserving the upper bounds on the symbolic state space. More importantly, experiments exhibit reductions of modest or strong degree compared to the classical approach. In no case an increase of the number of symbolic states occurred.

The structure of the paper is as follows: in Section 2 we introduce the basic notions of timed automata, notably timed words and the languages abstracted from time stamps. In Section 3, we introduce independence of transitions and Mazurkiewicz traces, as well as a relaxed semantics for timed automata where time needs to advance only between dependent actions. This semantics is related to the classical semantics of Section 2 and we show that the emptiness problems of the classical semantics and the partial order semantics are equivalent. In Section 4, we define a symbolic automaton that accepts the language of the relaxed semantics. This automaton relies on the concept of event zones that represent the time constraints that must be satisfied by words up to dependency relation (these event zones are represented by the classical difference bounded matrices). Section 5 revisits the problem from a language theoretic point of view which explains the difficulties of previous approaches by showing that the existence of a finite automaton is not guaranteed. Finally we introduce a simulation between event zones that has a finite index. In Section 6, we use this relation in a algorithm that solves the emptiness problem of timed automata and we give experiments that show that this algorithm behaves well in practice. In Section 7, we discuss future work, in particular concerning the extension by state invariants. For readability, some long proofs are placed in the Appendix.

2. Basics

In this section, we introduce basic notions of timed words, timed languages, as well as their finite representation by timed automata [3]. Finally, the intrinsic combinatoric explosion of the state space needed for verifying this model is introduced.

For an alphabet Σ of actions denoted by $a, b, c \ldots, \Sigma^*$ is the set of finite sequences $a_1 \ldots a_n$ called words, with ε the empty word. The length n of a word $a_1 \ldots a_n$ is denoted by $|a_1 \ldots a_n|$. A *timed word* is a sequence $(a_1, \tau_1) \ldots (a_n, \tau_n)$ of elements in $(\Sigma \times \mathbb{R}^+)^*$, with $\mathbb{R}^+$ the set of non-negative reals, the τ_i's are *time stamps*. For convenience, we set $\tau_0 = 0$ to be an additional time stamp for the beginning. In the literature, timed words are also represented as pairs (w, τ) with $\tau : \{1, \ldots, |w|\} \rightarrow \mathbb{R}^+$ a function assigning a time stamp to each position in the word $w = a_1 \ldots a_n$. A timed word is *normal* if $\tau_i \leqslant \tau_j$ for $i \leqslant j$ as in $(a, 3.2)(c, 4.5)(b, 6.3)$ whereas $(a, 3.2)(c, 2.5)(b, 6.3)$ is not normal. Normal timed words represent temporally ordered sequences of events and serve as standard semantics of timed automata in the literature. Concatenation of normal timed words is only a partial function and the set of normal timed words is thus a *partial monoid* only.

In timed systems, events can occur only if certain time constraints are satisfied. In timed automata, a finite set of real valued[3] variables X, called *clocks*, are used to express the time constraints between an event that resets a clock and another event that refers to the clock value at the time of its occurrence. The clock constraints permitted here are conjunctions of *atomic clock constraints*, comparisons between a clock and a numerical constant. To preserve decidability, constants are assumed to be positive rationals and for simplicity in $\mathbb{N}$, the set of natural numbers. For a set of clocks X, the set $\Phi(X)$ of clock constraints ϕ is formally defined by the following grammar:

$$\phi := \text{true} | x \bowtie c \, | \phi_1 \wedge \phi_2,$$

where x is a clock in X, $\bowtie \in \{<, \leqslant, >, \geqslant\}$ and c is a constant in $\mathbb{N}$ (true is for transitions without conditions). Another way of looking at clock constraints is sets of atomic constraints that must all be satisfied.

A *clock valuation* $v : X \to \mathbb{R}$ is a function that assigns a real number to each clock. We denote by $v + \tau$ the clock valuation that translates all clock $x \in X$ synchronously by τ such that $(v+\tau)(x) = v(x) + \tau$. For a subset C of clocks, $v[C \leftarrow 0]$ denotes the clock valuation with $v[C \leftarrow 0](x) = 0$ if $x \in C$ and $v[C \leftarrow 0](x) = v(x)$ if $x \notin C$, i.e. the valuation where the clocks in C are reset to 0. The satisfaction of the clock constraint ϕ by the clock valuation v, i.e. the fact that all atomic constraints are satisfied when substituting $v(x)$ for x, is denoted by $v \vDash \phi$.

Given an alphabet Σ and a set of clocks X, a *timed automaton* is a quintuple $\mathcal{A} = (\Sigma, S, s_0, \to, F)$ where S is a finite set of locations, $s_0 \in S$ is the initial location, $F \subseteq S$ is the set of final locations and $\to \subseteq S \times [\Sigma \times \Phi(X) \times 2^X] \times S$ is a set of transitions. For a transition $(s, a, \phi, C, s') \in \to$ we write $s \overset{(a,\phi,C)}{\to} s'$ and call a the *label* of the transition. Fig. 1 describes several timed automata.

For our formal development, we introduce three distinct notions of sequences of execution: *paths* (ignoring time constraints), *runs* (paths with time stamps respecting the time constraints), *normal runs* (furthermore the time stamps respect the progress of time):

A *path* in $\mathcal{A}$ is a finite sequence $s_0 \overset{(a_1,\phi_1,C_1)}{\to} s_1 \ldots \overset{(a_n,\phi_n,C_n)}{\to} s_n$ of consecutive transitions $s_{i-1} \overset{(a_i,\phi_i,C_i)}{\to}, s_i$. The word $a_1 \ldots a_n$ of transition labels is called the path labeling. If s_n is in F, the path is said to be accepted. The set of labelings of accepted paths is called the *untimed language* of $\mathcal{A}$ and denoted $L(\mathcal{A})$.

A state or *configuration* (s, v) of a timed automaton consists of the current location s and the clock values, represented by a clock valuation v.

A *run* of a timed automaton is a path extended by time stamps for the transition occurrences satisfying clock constraints and resets:

$(s_0, v_0) \overset{(a_1,\phi_1,C_1),\tau_1}{\longrightarrow} (s_1, v_1) \ldots \overset{(a_n,\phi_n,C_n),\tau_n}{\longrightarrow} (s_n, v_n)$ where $(a_1, \tau_1) \ldots (a_n, \tau_n)$ is a timed word and $(v_i)_{0 \leqslant i \leqslant n}$ are clock valuations defined by

$$v_0(x) = 0 \quad \text{for all } x \in \mathcal{C},\ \tau_0 = 0$$
$$v_{i-1} + (\tau_i - \tau_{i-1}) \vDash \phi_i,$$

[3] In the classical case (for normal timed words) *positive real values* would suffice, see Remark 1 for explanation.

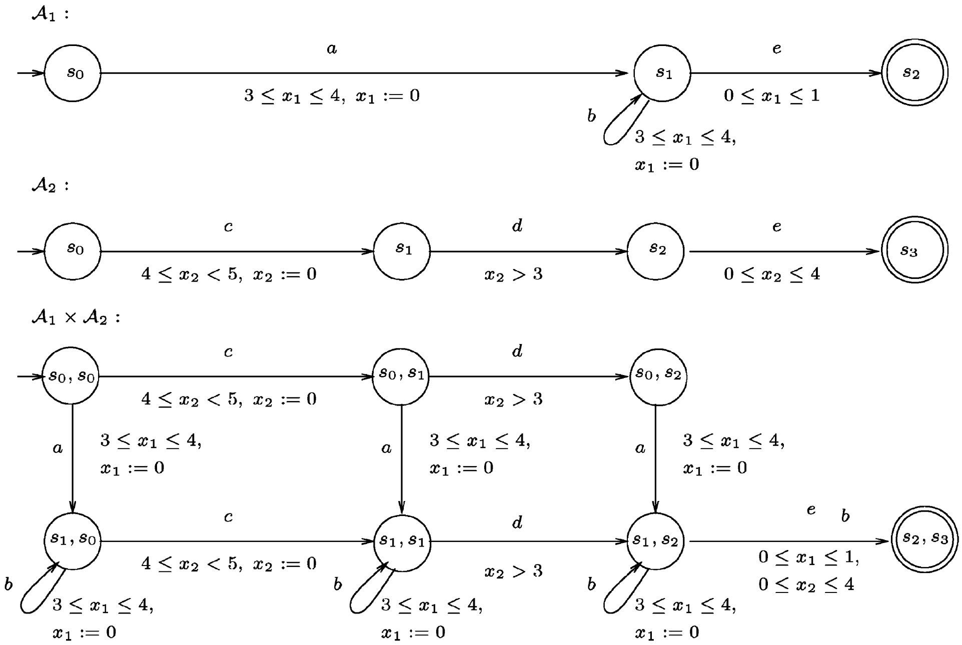

Fig. 1. A system of two timed automata $\mathcal{A}_1$, $\mathcal{A}_2$ and its semantics as product.

$$v_i = (v_{i-1} + (\tau_i - \tau_{i-1}))[C_i \leftarrow 0].$$

The timed word $(a_1, \tau_1) \ldots (a_n, \tau_n)$ is the *timed labeling* of the run. The run is *accepted by* $\mathcal{A}$ if $s_n \in F$.

A *normal run* is a run such that its timed labeling $(a_1, \tau_1) \ldots (a_n, \tau_n)$ is a normal timed word i.e. $\tau_1 \leqslant \tau_2 \leqslant \cdots \leqslant \tau_n$.

Remark 1. It is straightforward to see that for normal runs the valuations always produce positive values: clocks are either reset to 0 or the translations $v + (\tau_i - \tau_{i-1})$ increase the values since $\tau_i \geqslant \tau_{i-1}$. In non-normal runs, this need not be the case.

The *timed language* $L_T(\mathcal{A})$ of $\mathcal{A}$ is the set of *normal* timed words that are labelings of (normal) runs accepted by $\mathcal{A}$. The path labeling $a_1 \ldots a_n$ is said to be *realizable* if for some time stamps τ_i the normal timed word $(a_1, \tau_1) \ldots (a_n, \tau_n)$ (then called the normal realization of $a_1 \ldots a_n$) is the labeling of a normal run. The language of realizable words that are the labeling of an *accepted run* is denoted by $L_N(\mathcal{A})$. For instance, in the product automaton of Fig. 1 $(a, 3.2)(c, 4)(b, 6.2) \in L_T(\mathcal{A})$ is a normal realization of the path labeling *acb*, hence $acb \in L_N(\mathcal{A})$.

A timed automaton is *action deterministic* if for two transitions $s \overset{(a,\phi_1,C_1)}{\rightarrow} s_1$ and $s \overset{(a,\phi_2,C_2)}{\rightarrow} s_2$, we have that $\phi_1 = \phi_2$, $C_1 = C_2$ and $s_1 = s_2$. Similarly, we call the timed automaton *constraint consistent* if actions determine uniquely clock constraints and resets, i.e. for each pair of transitions (s_1, a, ϕ, C, s_2) and $(s_1', a, \phi', C', s_2')$ with the same action, we have $\phi = \phi'$ and $C = C'$. In that case, given an action a, the unique clock constraint and reset are denoted by ϕ_a and C_a, respectively. In this paper, we will *only consider timed automata that are action deterministic and constraint consistent* and the next proposition states that this assumption is not a restriction w.r.t. deciding whether $L_T(\mathcal{A})$ is empty or not.

Proposition 2. *Let $\mathcal{A} = (\Sigma, S, s_0, \rightarrow, F)$ be a timed automaton. There exists a deterministic timed automaton $\mathcal{A} = (\Sigma', S, s_0, \rightarrow', F)$ such that $L_T(\mathcal{A}) = \emptyset$ iff $L_T(\mathcal{A}') = \emptyset$. Furthermore, there is a morphism $\sigma : \Sigma' \rightarrow \Sigma$ such that $(a_1, \tau_1) \dots (a_n, \tau_n) \in L_T(\mathcal{A}')$ implies $(\sigma(a_1), \tau_1) \dots (\sigma(a_n), \tau_n) \in L_T(\mathcal{A})$.*

Proof. For each $a \in \Sigma$ let Δ_a be the sequence of tuples (s, a, ϕ, C, s') such that $(s, a, \phi, C, s') \in \Delta_a$ iff $\exists (s, a, \phi', C', s'') \in \Delta_a$ or $\exists (s'', a, \phi', C', s''') \in \Delta_a$ with $\phi \neq \phi'$ or $C \neq C'$, i.e. Δ_a contains all the transitions that exhibit a non-deterministic or a constraint non-consistency behavior. Let n_a be the number of elements of the sequence.

Let $\Sigma_a = \{a_1, \dots, a_{n_a}\}$ and let $\rightarrow'$ be the relation obtained by replacing the ith element $(s, a, \phi, C, s') \in \Delta_a$ with (s, a_i, ϕ, C, s') (for all $a \in \Sigma$). The morphism σ is defined by $\sigma(a_i) = a$.

A straightforward induction on the length of runs in $\mathcal{A}$ and $\mathcal{A}'$ proves that $L_T(\mathcal{A}) = \emptyset$ iff $L_T(\mathcal{A}') = \emptyset$ and that σ satisfies the second property. □

The main issue in the analysis of timed automata is the *emptiness problem*, i.e. to decide whether $L_T(\mathcal{A})$ is empty and if not extract a witness run. Since the number of configurations is infinite, the classical way to solve this problem is to construct a finite quotient of the set of clock valuations [3] thus obtaining a finite automaton for an untimed abstraction of $L_T(\mathcal{A})$ such as $L_N(\mathcal{A})$ which has an equivalent emptiness problem. Technically, the states of this automaton are couples (s, Z) where s is a location of the original timed automaton and Z is a "zone", a symbolic representation of an equivalence class of clock valuations. However, despite substantial progress in the representation of symbolic states [6,13,15], the number of zones is unavoidably exponential in the number of clocks and the resulting combinatorial explosion remains a main challenge for the applicability of timed automata.

3. Independence for timed automata

In this work we focus on an aspect of this combinatorial explosion that results from the analysis of *concurrent* timed systems, typically *networks of timed automata*. Such networks are the basis of timed automata tools [5,26] and consist of individual components that either execute transitions independently or synchronize with other components according to some communication mechanism (by rendezvous, shared variables, etc.). Each automaton may have local clocks, but it may also share global clocks with other automata. Without formally defining such networks, Fig. 1 actually shows a timed automaton as a product of two component automata. This product construction is the source of a lot of redundancy in the resulting timed automaton, notably exposing pairs of transitions like the ones labeled a and c that might be explored in any order leading to the same state. On the level of untimed languages, this results in the closure of $L(\mathcal{A})$ under exchanges of independent transitions: if $uacv \in L(\mathcal{A})$ then also $ucav \in L(\mathcal{A})$. The same, however, is not true for $L_T(\mathcal{A})$ or for $L_N(\mathcal{A})$. As a result, paths that are equivalent for $L(\mathcal{A})$ need not be equivalent for $L_N(\mathcal{A})$ thus leading to incomparable symbolic states (s, Z_1), (s, Z_2).

This section aims to reestablish these commutations in the more general context of Mazurkiewicz trace theory [14] that we extend to the time setting.

To model concurrency, we use an independence relation between actions such that actions are independent when the order of their occurrence is irrelevant. Formally, an *independence relation* I for an (action deterministic and constraint consistent) timed automaton $\mathcal{A} = (\Sigma, S, s_0, \rightarrow, F)$ is a symmetric and irreflexive relation $I \subseteq \Sigma \times \Sigma$ such that the following two properties hold for any two $a, b \in \Sigma$ with $a\ I\ b$:

(i) $s \overset{(a,\phi_a,C_a)}{\rightarrow} s_1 \overset{(b,\phi_b,C_b)}{\rightarrow} s_2$ implies $s \overset{(b,\phi_b,C_b)}{\rightarrow} s_1' \overset{(a,\phi_a,C_a)}{\rightarrow} s_2$ for some location s_1'

(ii) $C_a \cap C_b = \emptyset$ and no clock x in C_b belongs to an atomic clock constraint $x \bowtie c$ of ϕ_a and conversely no clock x in C_a belongs to an atomic clock constraint $x \bowtie c$ of ϕ_b.

We also use the dependence relation $D = \Sigma \times \Sigma - I$, which is reflexive and symmetric.

Intuitively, condition (ii) arises from the view of clocks as shared variables in concurrent programming: an action resetting a clock is writing it whereas an action with a clock constraint on this clock is reading it. The restriction states that two actions are dependent if both are writing the same variable or one is writing a variable the other one is reading it.

Since $I = \emptyset$ trivially meets (i) and (ii) such a relation always exists. Computing a good (the larger, the better) I meeting (i) and (ii) is a matter of static analysis and is typically done on the level of a network *before* constructing the product timed automaton: sufficient criteria for (i) may require that two transitions originate from distinct components and do not have conflicts around shared variables and do not synchronize on the same channels. For instance, $I = \{(a, c), (c, a), (b, c), (c, b), (a, d), (d, a), (b, d), (d, b)\}$ is an independence relation for the timed automaton of Fig. 1.

The *Mazurkiewicz trace equivalence* associated to the independence relation I is the least congruence $\simeq$ over Σ^* such that $ab \simeq ba$ for any pair of independent actions $a\ I\ b$. A *trace* $[u]$ is the congruence class of a word $u \in \Sigma^*$.

By definition, two words are equivalent with respect to $\simeq$ if they can be obtained from each other by a finite number of exchanges of adjacent independent actions. For e.g., $abc \simeq acb \simeq cab$ with I as defined above for Fig. 1 but $abc \not\simeq bac$ (a and b are dependent). In other words, this permutation of actions between two equivalent words lets the relative order of occurrences of dependent actions unchanged, formally:

Lemma 3. *Let I be a independence relation, $\simeq$ the induced Mazurkiewicz trace equivalence and $a_1 \ldots a_n \simeq b_1 \ldots b_n$ be two equivalent words. There exists a uniquely determined permutation $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ such that $a_i = b_{\pi(i)}$ and for $a_i\ D\ a_j$ we have $i < j$ iff $\pi(i) < \pi(j)$.*

Conversely, let $a_1 \ldots a_n$ be a word and $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ be a permutation of indices such that for each pair i, j $a_i\ D\ a_j$ we have $i < j$ iff $\pi(i) < \pi(j)$. Then $a_{\pi(1)} \ldots a_{\pi(n)} \simeq a_1 \ldots a_n$.

Proof. By induction on the number of exchanges. $\square$

For convenience in applications to timed words, we assume π to be extended to 0 with $\pi(0) = 0$.

The untimed language $L(\mathcal{A})$ of a timed automaton $\mathcal{A}$ is closed under the equivalence $\simeq$ and this is the theoretical foundation of many partial order reduction approaches. For instance, reductions that preserve at least one representative for each equivalence class do preserve non-emptiness of the untimed languages. Moreover the equivalence relation extends to runs when disregarding normality constraints:

Lemma 4. *Let $(a_1, \tau_1) \dots (a_n, \tau_n)$ be the timed labeling of a run, $\pi : \{1, \dots, n\} \to \{1, \dots, n\}$ be a permutation with $a_1 \dots a_n \simeq a_{\pi(1)} \dots a_{\pi(n)}$. Then $(a_{\pi(1)}, \tau_{\pi(1)}) \dots (a_{\pi(n)}, \tau_{\pi(n)})$ is also a timed labeling of a run.*

Proof. The proof is by induction of the number of exchanges in π, it is sufficient to consider the case of a single exchange. Let

$$(a_1, \tau_1) \dots (a_k, \tau_k)(a, \tau_{k+1})(b, \tau_{k+2})(a_{k+3}, \tau_{k+3}) \dots (a_n, \tau_n),$$

be the time labeling where $a\ I\ b$ and let $r = (s_0, v_0) \dots (s_n, v_n)$ be the corresponding run. Assume that $s_k \stackrel{a,\phi_a,C_a}{\to} s_{k+1} \stackrel{b,\phi_b,C_b}{\to} s_{k+2}$.

We prove the existence of a unique run $r' = (s'_0, v'_0) \dots (s'_n, v'_n)$ with timed labeling

$$(a_1, \tau_1) \dots (a_k, \tau_k)(b, \tau_{k+2})(a, \tau_{k+1})(a_{k+3}, \tau_{k+3}) \dots (a_n, \tau_n)$$

such that $s'_i = s_i$ for $i \neq k+1$ and $v'_i = v_i$ for $i \notin \{k+1, k+2\}$.

By property (i) of I, $s_k \stackrel{b,\phi_b,C_b}{\to} s'_{k+1} \stackrel{a,\phi_a,C_a}{\to} s_{k+2}$ and all other transitions are unchanged hence $s'_i = s_i$ for $i \neq k+1$.

The sequence r' is a run if the time valuations v'_i satisfy the constraints. We consider two cases:

(1) $i \leqslant k$ or $i > k+3$. The result holds since r is a run:

(2) $i = k+1$, $i = k+2$ and $i = k+3$.

Since r is a run, $v_k + (\tau_{k+2} - \tau_k) \vDash \phi_b$.

By condition (ii) of independence, no clock mentioned in ϕ_b is reset in C_a hence $(v_{k+1} + (\tau_{k+2} - \tau_{k+1}))(x) = v_{k+1}(x) + (\tau_{k+2} - \tau_{k+1}) = (v_k + (\tau_{k+1} - \tau_k))(x) + (\tau_{k+2} - \tau_{k+1}) = (v_k + (\tau_{k+2} - \tau_k))(x)$ for any clock x mentioned in ϕ_b.

Therefore $v_{k+1} + (\tau_{k+2} - \tau_{k+1}) \vDash \phi_b$ iff $v_k + (\tau_{k+2} - \tau_k) \vDash \phi_b$.

Therefore the transition $s_k \stackrel{b,\phi_b,C_b}{\to} s'_{k+1}$ is enabled at τ_{k+2} yielding (s'_{k+1}, v'_{k+1}).

Similarly the transition $s'_{k+1} \stackrel{a,\phi_a,C_a}{\to} s_{k+2}$ is enabled at τ_{k+1} yielding a configuration (s_{k+2}, v'_{k+2}).

Since $C_a \cap C_b = \emptyset$ we get $v'_{k+2} = v_{k+2} + (\tau_{k+1} - \tau_{k+2})$ which implies $v'_{k+2} + (\tau_{k+3} - \tau_{k+1}) = v_2 + (\tau_{k+3} - \tau_{k+2})$.

This guarantees that the transition corresponding to a_{k+3} is still possible at τ_{k+3} and that $v'_{k+3} = v_{k+3}$. $\square$

However, Lemma 4 only claims commutability of runs without taking time progress into account. For the timed language $L_T(\mathcal{A})$ and consequently for $L_N(\mathcal{A})$, the normality condition may exclude some representatives in a trace: let *abc* and *acb* be two equivalent paths of Fig. 1. We already know that the latter one is in $L_N(\mathcal{A})$. For *abc*, any timed word $(a, \tau_1)(b, \tau_2)(c, \tau_3)$ labeling a run in $\mathcal{A}$ is such that $3 \leqslant \tau_1 \leqslant 4$, $6 \leqslant \tau_2 - \tau_1 \leqslant 8$ and $4 \leqslant \tau_2 + \tau_3 - \tau_2 < 5$. This set of inequalities has no solution such that the timed word is normal, i.e. $\tau_1 \leqslant \tau_2 \leqslant \tau_3$, as these inequations imply that $6 \leqslant \tau_2 \leqslant 8$ and $4 \leqslant \tau_3 < 5$. Therefore we introduce a weaker notion of normality:

A timed word $(a_1, \tau_1) \ldots (a_n, \tau_n)$ is *I-normal* iff for any two letters a_i, a_j with $i \leqslant j$ *and additionally* $a_i\ D\ a_j$ we have $\tau_i \leqslant \tau_j$. In Fig. 1, the timed word $(a, 3.2)(b, 6.2)(c, 4.5)$ is *I*-normal. The intuition behind this relaxation of constraints is that in practice, actions are dependent if they are executed by the same component in a network of timed automata. This non-decreasing condition on action occurrences model the sequential behavior of each component. In [8], this is modeled by considering a local time for each component. The interaction between components leads to the propagation of time progress to other components (formally due to dependency).

In analogy to realizable words, we say that $a_1 \ldots a_n$ is *I-realizable* iff it is the labeling of a run $(s_0, v_0) \xrightarrow{(a_1, \phi_{a_1}, C_{a_1}), \tau_1} (s_1, v_1) \ldots \xrightarrow{(a_n, \phi_{a_n}, C_{a_n}), \tau_n} (s_n, v_n)$ in $\mathcal{A}$ such that $(a_1, \tau_1) \ldots (a_n, \tau_n)$ is *I*-normal. As for L_N, let $L_I(\mathcal{A})$ denote the set of *I*-realizable words $a_1 \ldots a_n$ that are the labeling of an *accepted run* (i.e. $s_n \in F$). For instance, *abc* is *I*-realizable in Fig. 1 as time stamps 3.2, 6.2, 4.5 satisfy clock constraints of transitions from location (s_0, s_0) to (s_1, s_1) (see the inequality system above) and $(a, 3.2)(b, 6.2)(c, 4.5)$ is *I*-normal. Moreover, *abc* is also in $L_I(\mathcal{A})$ since in the automaton of Fig. 1 all states—hence (s_1, s_1)—are final. Obviously $L_N(\mathcal{A}) \subseteq L_I(\mathcal{A})$.

By definition $L_T(\mathcal{A}) = \emptyset$ if and only if $L_N(\mathcal{A}) = \emptyset$. Moreover, the following proposition implies that $L_N(\mathcal{A}) = \emptyset$ iff $L_I(\mathcal{A}) = \emptyset$, so that we can check this emptiness problem equivalently for either language.

Proposition 5. *For every I-normal labeling $(a_1, \tau_1) \ldots (a_n, \tau_n)$ of a run in an action deterministic, constraint consistent timed automaton $\mathcal{A}$, there exists $(a'_{\pi(1)}, \tau_{\pi(1)}) \ldots (a_{\pi(n)}, \tau_{\pi(n)})$ an equivalent normal labeling of an* (*equivalent*) *run in $\mathcal{A}$, where π is a permutation as defined in Lemma* 3.

Proof. Consider the following ordering on $\{1, \ldots, n\}$: $i \sqsubset j$ iff $\tau_i < \tau_j$ or $\tau_i = \tau_j$ and $i < j$. There is a unique permutation such that $i \sqsubset j$ iff $\pi(i) < \pi(j)$. Moreover, for $a_i D a_j$ and $i < j$, I-normality implies that $\tau_i \leqslant \tau_j$ and finally $\pi(i) < \pi(j)$, i.e. π yields an equivalent path. By Lemma 4, thus $(a'_{\pi(1)}, \tau_{\pi(1)}) \ldots (a_{\pi(n)}, \tau_{\pi(n)})$ is a timed labeling of some run and by the construction of $\sqsubset$ it is a normal timed word. $\square$

A sorting algorithm provides an efficient way of computing a normal timed labeling of a run from an I-normal labeling.

A key main feature of $L_I(\mathcal{A})$ is the closure under equivalence that is stated in Theorem 6. In principle this allows to limit exploration of realizable clocked words to representatives of equivalence class:

Theorem 6. (1) *Let $u \simeq v$ and $u \in L_I(\mathcal{A})$ then $v \in L_I(\mathcal{A})$.*

(2) $L_I(A) = \{u \mid \exists v \simeq u.v \in L_N(\mathcal{A})\}$.

Proof. (1) Let $u = a_1 \dots a_n$, $v = b_1 \dots b_n$ and π be the permutation linking $a_1 \dots a_n$ and $b_1 \dots b_n$ according to Lemma 3. Let $(a_1, \tau_1) \dots (a_n, \tau_n)$ an *I*-normal labeling of some accepting run of $\mathcal{A}$. Then $(b_1, \tau_{\pi(1)}) \dots (b_n, \tau_{\pi(n)})$ is a timed labeling of some accepting run according to Lemma 4 and it inherits *I*-normality since π preserves the order of occurrences of dependent actions.

(2) "$\supseteq$" follows from $L_N(\mathcal{A}) \subseteq L_I(\mathcal{A})$ (normality implies I-normality) and reflexivity of $\simeq$. "$\subseteq$" is an easy consequence of Proposition 5. $\square$

4. A symbolic automaton for L_I

The goal of this section is to build a symbolic automaton for $L_I(\mathcal{A})$ called *event zone automaton*. A state of the symbolic automaton will be a pair (location, *time stamp constraints*) where the latter is a set of inequalities between time stamps.

Time stamp constraints: let $\mathbb{T} = \{t_0, t_1, \dots\}$ be a set of time stamp variables, the set of *time stamp constraints* is defined by the grammar:

$$\psi := \mathit{true} \mid t_i - t_j \prec c \mid \psi_1 \wedge \psi_2,$$

where t_i, t_j are time stamp variables in $\mathbb{T}$, $\prec \in \{<, \leqslant\}$ and c is a constant in $\mathbb{Z}$. An atomic time constraint is a time constraint of the form $t_i - t_j \prec c$.

Like a valuation for a clock constraint, an *interpretation* of a time stamp constraint ψ is a function $v : \mathbb{T} \to \mathbb{R}^+$ assigning a non-negative real number τ_i to each time stamp variable t_i. The satisfaction of ψ by v is denoted $v \vDash \psi$ and in that case v is a *model* for ψ. A time stamp constraint ψ is *consistent* if it has a model otherwise it is *inconsistent*. For convenience, an absence of time stamp constraints $t_i - t_j \prec c$ for some pair t_i, t_j is denoted by $t_i - t_j < \infty$ which is satisfied by any interpretation by definition.

Two time stamp constraints are *equivalent* if they have the same models. We show how to compute a canonical time stamp constraint for every consistent time stamp constraint: this canonical form contains only the "tightest time stamp constraints" that can be derived from the initial time stamp constraints.

First, we extend the comparison relation $<$ over integers to elements in $\mathbb{Z} \times \{<, \leqslant\} \cup \{\infty, <\}$ by

$$(\prec, c) < (<, \infty) \quad \text{iff } (\prec, c) \neq (<, \infty),$$
$$(\prec_1, c_1) < (\prec_2, c_2) \quad \text{iff } c_1 < c_2 \text{ or } c_1 = c_2 \text{ and } \prec_1 < \prec_2,$$

where $<$ defined to be less than $\leqslant$.

We extend the addition over integers to elements in $\mathbb{Z} \times \{<, \leqslant\} \cup \{\infty, <\}$ by

$$(\prec, c) + (<, \infty) = (<, \infty)$$
$$(\prec_1, c_1) + (\prec_2, c_2) = \begin{cases} (\prec_1, c_1 + c_2) & \text{if } \prec_1 < \prec_2, \\ (\prec_2, c_1 + c_2) & \text{otherwise.} \end{cases}$$

A time stamp constraint ψ is in *canonical form* if for every atomic time stamp constraint $t_i - t_j \prec c$ in ψ $(\prec, c)$ is the minimal value of $(\prec_{ii_1}, c_{ii_1}) + (\prec_{i_1 i_2}, c_{i_1 i_2}) + \dots + (\prec_{i_k j}, c_{i_1 i_2})$

such that $t_i - t_{i_1} \prec_{ii_1} c_{ii_1}, t_{i_1} - t_{i_2} \prec_{i_1i_2} c_{i_1i_2}, \ldots, t_{i_k} - t_j \prec_{i_kj} c_{i_kj}$ are atomic time stamp constraints of ψ.

Computation of the canonical form:

Proposition 7. *A time stamp constraint ψ is inconsistent iff there exist indices $i_1, i_2, \ldots, i_j$ with $i_1 = i_j$ and such that $t_{i_k} - t_{i_{k+1}} \prec_{i_ki_{k+1}} c_{i_ki_{k+1}} \in \psi$ for $k = 1, \ldots, j-1$ and $(\prec_{i_1i_2}, c_{i_1i_2}) + (\prec_{i_2i_3}, c_{i_2i_3}) + \cdots + (\prec_{i_{j-1}i_j}, c_{i_{j-1}i_j}) < (\leqslant, 0)$. For each consistent time stamp constraint ψ there exists a unique canonical equivalent time stamp constraint, denoted by $cf(\psi)$.*

Proof. To a time stamp constraint ψ, we associate the complete weighted oriented graph (V, E) such that $E = \mathbb{T}$ and the edge t_i, t_j has weight $(\prec, c)$ if $t_i - t_j \prec c$. The data structure used to represent the graph is the difference-bound matrix [15] representation (DBM in short) which is an adjacency matrix whose indices are time stamp variables and entries are pairs $(\prec, c)$. We use the Floyd–Warshall algorithm [9] to compute the shortest paths in this graph. When the Floyd–Warshall algorithm computes a loop with weight $(\prec, c) < (\leqslant, 0)$, then ψ is inconsistent, otherwise the canonical form $cf(\psi)$ is the conjunction of atomic constraints $t_i - t_j \prec c$ where $(\prec, c)$ is the weight computed by the algorithm.

I-realizability: To express I-realizability in terms of time stamp constraints we need to define special positions in a path. Given a path labeling $a_1 \ldots a_n$ we define $last_a(a_1 \ldots a_n)$, the last occurrence of a, to be the maximal k such that $a_k = a$, if such a k exists, otherwise $last_a(a_1 \ldots a_n) = 0$. For instance in Fig. 1, $last_a(ac) = 1$ and $last_c(ac) = 2$. Similarly, we define $last_x(a_1 \ldots a_n)$ to be the maximal position k at which x is reset, that is $x \in C_{a_k}$, if such a position exists, otherwise $last_x(a_1 \ldots a_n) = 0$ (every clock is reset at the beginning). In Fig. 1, $last_{x_1}(ac) = 1$ and $last_{x_2}(ac) = 2$.

With these positions we express that in a word dependent actions are ordered according to their order of occurrence (condition (i) in the following) and that clock constraints are satisfied (conditions (ii) and (iii)) allowing to check I-realizability on the level of consistency.

For the labeling $a_1 \ldots a_n$ of a path in $\mathcal{A}$ let $\psi_{a_1 \ldots a_n}$ be the *associated time stamp constraint* which is the conjunction of the three following time stamp constraints:

- ψ_1 is the conjunction of atomic time stamp constraints $t_0 - t_j \leqslant 0$ and $t_i - t_j \leqslant 0$ for all i with $i \leqslant j$ and $a_i \; D \; a_j$ such that $i = last_{a_i}(a_1 \ldots a_{j-1})$;
- ψ_2 is the conjunction of atomic time stamp constraints $t_j - t_i \prec c$ with $1 \leqslant i \leqslant n$ and for all atomic clock constraints $x \prec c$ in ϕ_j, and $i = last_x(a_1 \ldots a_{j-1})$;
- ψ_3 is the conjunction of atomic time stamp constraints $t_i - t_j \prec -c$ with $1 \leqslant i \leqslant n$ and for all atomic clock constraints $x \succ c$ in ϕ_j, and $i = last_x(a_1 \ldots a_{j-1})$.

As an example, let us consider the labeling *abc* of a path in the timed automaton of Fig. 1. Any I-normal realization is some timed word $(a, \tau_1)(b, \tau_2)(c, \tau_3)$ where the time stamps τ_0 (the initial time stamp), τ_1, τ_2, τ_3 must satisfy the time stamps constraint ψ_{abc} which is the conjunction of $t_0 - t_1 \leqslant 0$, $t_0 - t_1 \leqslant 0$, $t_1 - t_2 \leqslant 0$, $t_0 - t_3 \leqslant 0$ (c depends on no preceding action), $t_0 - t_1 \leqslant -3$, $t_1 - t_0 \leqslant 4$, $t_1 - t_2 \leqslant -3$, $t_2 - t_1 \leqslant 4$, $t_0 - t_3 \leqslant -4$, $t_3 - t_0 < 5$ (the constraint is satisfied iff the replacement of t_i by τ_i in ψ_{abc} yields true).

Proposition 8. *Let $\mathcal{A} = (\Sigma, S, s_0, \rightarrow, F)$ be a deterministic timed automaton and I an independence relation. Moreover, let $a_1 \dots a_n$ be a path labeling of $\mathcal{A}$. The word $a_1 \dots a_n$ is I-realizable iff its associated time stamp constraint $\psi_{a_1 \dots a_n}$ is consistent.*

Proof. Suppose that $a_1 \dots a_n$ is the labeling of the run

$$(s_0, v_0) \xrightarrow{(a_1, \phi_1, C_1), \tau_1} (s_1, v_1) \dots \xrightarrow{(a_n, \phi_n, C_n), \tau_n} (s_n, v_n).$$

Time stamps τ_i trivially satisfy condition (i) as by definition $(a_1, \tau_1) \dots (a_n, \tau_n)$ is I-normal.

For conditions (ii) and (iii), consider an atomic clock constraint $x \bowtie c$ in ϕ_i. Let i be the maximal position in $a_1 \dots a_{j-1}$ where x is reset, that is $i = last_x(a_1 \dots a_{j-1})$. Again by definition of a run $v_i(x)$ has to be equal to 0. Hence as x is not reset between positions $i+1$ and $j-1$, $v_{j-1}(x) = (\tau_{j-1} - \tau_{j-2}) + (\tau_{j-2} - \tau_{j-3}) + \dots + (\tau_{i+1} - \tau_i) + v_i(x)$ which implies $v_{j-1}(x) = \tau_{j-1} - \tau_i$. Finally, from the satisfaction of $v_{j-1}(x) + (\tau_j - \tau_{j-1}) \bowtie c$ (by definition of a run), we obtain that $\tau_j - \tau_i \bowtie c$. Depending on the sign of $\bowtie$ this is the interpretation of $t_j - t_i \prec c$ or of $t_i - t_j \prec -c$ as required.

For the converse direction, let ψ be a consistent time stamp constraint, and let the interpretation $\tau_1, \tau_2, \dots$ be a model of ψ. From this interpretation, a run with valuations $(v_i)_{0 \leqslant i \leqslant n}$ is uniquely determined ($v_0(x) = 0$ for all clock x and for $i \neq 0$ $v_i(x) = 0$ if $x \in C_{a_i}$ otherwise $v_i(x) = v_{i-1}(x) + (\tau_i - \tau_{i-1})$). To check the run properties is similar to the above computation. □

Event zones and the $\simeq_{EZ}$ relation: this paragraph is devoted to the definition of an equivalence relation $\simeq_{EZ}$ between some symbolic states which is stable by transitions. This equivalence relation is then used to build the symbolic automaton.

An *event zone* is a triple $Z = (T, \psi, Last)$ where T is a set of time stamp variables, ψ is a time stamp constraint and $Last : X \cup \Sigma \rightarrow T$ is the *last occurrence function* that assigns to a clock or an action a the time stamps that represents, respectively, its last reset and the last occurrence of the action.

For instance, the preceding time stamp constraint ψ_{abc} associated to the labeling abc corresponds to the event zone $Z_{abc} = (T_{abc}, \psi_{abc}, Last_{abc})$ where $T_{abc} = \{t_0, t_1, t_2, t_3\}$, $Last_{abc}(a) = t_1$, $Last_{abc}(b) = t_2$, $Last_{abc}(c) = t_3$, $Last_{abc}(d) = Last_{abc}(e) = t_0$, $Last_{abc}(x_1) = t_2$, $Last_{abc}(x_2) = t_3$.

Formally, the event zone $Z_u = (T_u, \psi_u, Last_u)$ of the path labeling $u = a_1 \dots a_n$ is given by $T_u = \{t_0, \dots, t_n\}$, where ψ_u is the time stamp constraint associated to u and $Last_u(a) = t_i$ with $i = last_a(a_1 \dots a_n)$ for all action a, $Last_u(x) = t_i$ with $i = last_x(a_1 \dots a_n)$ for all clock x.

We extend the notion of consistency and canonical form to event zones: an event zone is consistent if its time stamp constraint is consistent. If an event zone $Z = (T, \psi, Last)$ is consistent, the canonical form of Z is the event zone $cf(Z) = (T, cf(\psi), Last)$.

A pair (s, Z) with s a location from the original timed automaton and Z an event zone is called a *symbolic state*.

The transition relation of the symbolic automaton is defined via the operation $\odot$ that takes a symbolic state (s, Z) and an action a and returns a symbolic state $(s', Z') = (s, Z) \odot a$. As usual the transition relation of the symbolic automaton is the set of $((s, Z), a, (s', Z'))$

such that $(s', Z') = (s, Z) \odot a$. Given a transition labeled with (a, ϕ_a, C_a) in the original timed automaton, the $\odot$ operation corresponds to the conjunction of the current time stamp constraint and the atomic time stamp induced by a for which a new time stamp variable is added. The last occurrence function is updated according to this addition.

More formally, the *extension* $(s_2, Z_2) = (s_2, (T_2, \psi_2, Last_2))$ of a symbolic state $(s_1, Z_1) = (s_1, (T_1, \psi_1, Last_1)$ by an action a is defined if there exists a transition $s_1 \overset{a,\phi_a,C_a}{\rightarrow} s_2$ such that

- $T_2 = T_1 \uplus \{t\}$ with t a fresh time stamp variable not in T_1,
- ψ_2 is the conjunction of
  - ψ_1,
  - $t_0 - t \leqslant 0$,
  - $t_i - t \leqslant 0$ for all $t_i = Last(b)$ for b such that $a\ D\ b$,
  - $t - t_i \prec c$ with $x \prec c$ in ϕ_a and $t_i = Last(x)$,
  - $t_i - t \prec -c$ with $x \succ c$ in ϕ_a and $t_i = Last(x)$.
- the function $Last_2$ is such that $Last_2(\alpha) = t$ for α a clock in C_a or $\alpha = a$ otherwise $Last_2(\alpha) = Last_1(\alpha)$.

This extension is denoted by $(s_1, Z_1) \odot a$.

The timed automaton in Fig. 1 gives rise to the extension $((s_1, s_0), Z_{ab}) \odot c = ((s_1, s_1), Z_{abc})$.

On these symbolic states we define an equivalence relation $\simeq_{EZ}$ compatible with the extension, that is such that if $(s_1, Z_1) \simeq_{EZ} (s_2, Z_2)$ then $(s_1, Z_1) \odot a \simeq_{EZ} (s_2, Z_2) \odot a$. Intuitively, we require that locations are the same and that time stamp constraints representing the same *last* occurrences are the same. This requirement comes from need to identify time stamp constraints of equivalent words while actions in the two words are not ordered in the same way (hence the same holds for the corresponding time stamps).

For convenience, we introduce the following notation: let $Z = (T, \psi, Last)$ be a consistent event zone and $\alpha, \beta \in X \cup \Sigma$. Let $t_i - t_j \prec c$ be the unique atomic time stamp constraint in $cf(Z)$ (i.e. in $cf(\psi)$) such that $Last(\alpha) = t_i$, $Last(\beta) = t_j$. We define $cf(Z)[\alpha, \beta]$ to be $(\prec, c)$.

The order relation $\lesssim_{EZ}$ is defined as follows: let (s_1, Z_1) and (s_2, Z_2) be two symbolic states. We say that $(s_1, Z_1) \lesssim_{EZ} (s_2, Z_2)$ if the following conditions are satisfied:

(i) $s_1 = s_2$,
(ii) Z_1 and Z_2 are both inconsistent, or Z_1 is consistent and Z_2 is inconsistent, or else they are both consistent and for all $\alpha, \beta \in X \cup \Sigma$ we have $cf(Z_1)[\alpha, \beta] \leqslant cf(Z_2)[\alpha, \beta]$.

We define $(s_1, Z_1) \simeq_{EZ} (s_2, Z_2)$ iff $(s_1, Z_1) \lesssim_{EZ} (s_2, Z_2)$ and $(s_2, Z_2) \lesssim_{EZ} (s_1, Z_1)$.

Moreover we extend $\lesssim_{EZ}$ and $\simeq_{EZ}$ to path labelings by defining $u \lesssim_{EZ} v$ iff $(s_u, Z_u) \lesssim_{EZ} (s_v, Z_v)$ with s_u, s_v the locations reached by the paths labeled, respectively, by u and v.

The following proposition states that $\lesssim_{EZ}$ is compatible with the zone extension.

Proposition 9. *Let* (s_1, Z_1) *and* (s_2, Z_2) *be two symbolic states such that* $(s_1, Z_1) \lesssim_{EZ} (s_2, Z_2)$, *let a be an action such that* $(s_1, Z_1) \odot a$ *is defined. Then* $(s_1, Z_1) \odot a \lesssim_{EZ} (s_2, Z_2) \odot a$ *holds.*

Proof. The proof is a tedious reasoning on the definition of $\lesssim_{EZ}$ but presents no particular difficulty. It is given in the Appendix. $\square$

Next we show that $\simeq_{EZ}$ is compatible with trace equivalence.

Proposition 10. *Let $u, v \in \Sigma^*$ to path labelings reaching locations s_u and s_v and with associated event zones Z_u, Z_v. Then $u \simeq v$ implies $(s_u, Z_u) \simeq_{EZ} (s_v, Z_v)$.*

Proof. First, we define the notion of zone isomorphism: let $Z_1 = (T_1, \psi_1, Last_1)$ and $Z_2 = (T_2, \psi_2, Last_2)$ be two event zones. We say that Z_1 and Z_2 are *isomorphic* iff $|T_1| = |T_2|$ and there exists a permutation $\pi : T_1 \to T_2$ such that for all i, j $t_i - t_j \prec c$ in ψ_1 iff $\pi(t_i) - \pi(t_j) \prec c$ in ψ_2 and $Last_2(\alpha) = \pi(Last_1(\alpha))$ for every $\alpha \in X \cup \Sigma$. The interesting point with this isomorphism is that if two symbolic states (s_1, Z_1) and (s_2, Z_2) with $s_1 = s_2$ and Z_1, Z_2 isomorphic then $(s_1, Z_1) \simeq_{EZ} (s_2, Z_2)$.

Secondly, we prove that the two event zones Z_{wab} and Z_{wba} are isomorphic for w some path labeling and a, b such that $a\ I\ b$.

Let Z_{wab} be such that $T_{wab} = T_w \cup \{t_1^a, t_1^{ab}\}$ and let Z_{wba} be such that $T_{wba} = T_w \cup \{t_2^b, t_2^{ba}\}$ are isomorphic.

The permutation π we consider only permutes times stamps introduced for a and b, i.e. $\pi(t_1^a) = t_2^{ba}$ and $\pi(t_1^{ab}) = t_2^b$ and otherwise $\pi(t) = t$.

For a constraint $t_i - t_j \prec c$ we have to consider four cases: (1) $t_i, t_j \in T_w$, (2) $t_i \in T_w, t_j \notin T_w$, (3) $t_j \in T_w, t_i \notin T_w$, and (4) $t_i, t_j \notin T_w$.

In the first case, the constraint is already in ψ_w and π preserves it identically.

For the other cases, we first remark that since a and b are independent there exists no $\alpha \in X \cup \Sigma$ whose last occurrence is updated both in Z_{wa} and in Z_{wab} such that $Last_{wa}(\alpha) = t_1^a$ and $Last_{wab}(\alpha) = t_1^{ab}$. The same holds in Z_{wb} and Z_{wba} with t_2^b and t_2^{ba}. Moreover, for a clock constraint $\alpha \bowtie c$ in ϕ_a or for $\alpha \in \Sigma$ with $\alpha\ D\ a$ it holds that $Last_w(\alpha) = Last_{wb}(\alpha)$ (and conversely $Last_w(\beta) = Last_{wa}(\beta)$ for $\beta \bowtie c$ in ϕ_b or for $\beta \in \Sigma$ with $\beta\ D\ b$).

For case (2) $t_i = Last_w(\alpha) = Last_{wb}(\alpha) = \pi(t_i)$ and $t_j = \pi(t_j)$, thus we have $t_i - t_j \prec c$ in ψ_{wa} iff $\pi(t_i) - \pi(t_j) \prec c$ in ψ_{wba}. Case (3) is similar. Case (4) is impossible ($c = \infty$), which ends the proof that Z_{wab} and Z_{wba} are isomorphic.

Finally, the general case follows by a straightforward induction on the number of permutations of actions in u to get v, that relies on the previous property and Proposition 9. $\square$

Let $\mathcal{Z}$ be the set of event zones and $Z_\varepsilon = (\{t_0\}, \psi_\varepsilon, Last_\varepsilon)$ be the special event zone associated to the empty word such that ψ_ε is $t_0 - t_0 \leqslant 0$ and $Last(\alpha) = t_0$ for all $\alpha \in C \cup \Sigma$ (everything is reset).

The *symbolic automaton* $\mathcal{A}' = (\Sigma', S', s_0', \to', F')$ associated to an action deterministic constraint consistent timed automaton $\mathcal{A} = (\Sigma, S, s_0, \to, F)$ is such that $\Sigma' = \Sigma$, $S' = (S \times \mathcal{Z})/_{\simeq_{EZ}}$ is the set of equivalence classes of symbolic states, the initial state is $s_0' = [(s_0, Z_\varepsilon)]$, the set of final quotient of symbolic states is $F' = \{[(s, Z)] \mid s \in F\}$ and the transition relation $\to' \subseteq S' \times [\Sigma \times \Phi(X) \times 2^X] \times S'$, is defined by $[(s, Z)] \xrightarrow{a} [(s', Z')]$ iff $s \xrightarrow{a, \phi_a, C_a} s'$ is in $\mathcal{A}$ and $Z' = (Z \odot a)$ is consistent.

Implementation: The major issue to implement event zones is to deal with the number of variables that are used. There are a priori as many variables as the length of the path plus one (t_0 for the beginning). However, if a path is shown to be in $L_I(\mathcal{A})$ then checking if it can be extended can be done on time stamp constraints between time stamps of last occurrences. The set of such time stamps is the set of time stamps in the co-domain of *Last*. We call the *delete operation* the restriction of an event zone over the set of variables not in the co-domain of *Last*.

We define $del(T, \psi, Last)$ to be the event zone $(T', \psi', Last')$ such that $Last' = Last$, T' is the co-domain of $Last'$ and ψ' is the restriction of ψ to atomic time stamp constraints with variables in T', i.e. ψ' is the conjunction of atomic time stamp constraint $t_i - t_j \prec c$ in ψ such that there exists $\alpha, \beta \in X \cup \Sigma$ and $t_i = Last(\alpha)$, $t_j = Last(\beta)$.

As this operation does not change the consistency of canonical event zones (part (i) in the following proposition) and commutes with the extension operation (part (ii) in the following proposition), we can change the transition relation of the symbolic timed automaton to lead to $Z' = del(cf(Z \odot a))$ while preserving $L_I(\mathcal{A})$.

Proposition 11.

(i) *Let u be the labeling of a path leading to the location* s_u *and a an action such that* $s_u \overset{(a,\phi_a,C_a)}{\rightarrow} s_{ua}$ *is a transition.* $(s_{ua}, Z_{ua}) \simeq_{EZ} (s_u, Z_u) \odot a$.

(ii) *For a symbolic state* (s, Z) *with Z a consistent event zone* $(s, Z) \simeq_{EZ} (s, del(cf(Z)))$.

Proof. (i) follows directly from the comparison between definitions of $\lesssim_{EZ}$, event zone of a word and the extension operation.

(ii) As Z is consistent, this follows from $(s, cf(Z)) \simeq_{EZ} (s, del(cf(Z)))$. This equivalence is obvious as the operation *del* does not affect time stamp constraints between time stamp constraints referring last occurrences and only constraints of this kind are concerned by $\lesssim_{EZ}$. $\square$

The maximal number of variables needed in an event zone is then the number of clocks plus the size of the alphabet plus 1, in contrast to the classical clock zone dimension of number of clocks plus one. However, the representation of clock zones can be significantly improved by static analysis: the references $Last(a)$ with $a \in \Sigma$, chosen here for simplicity of presentation, are used for the insertion of I-normality constraints into ψ. In [19], we have given instead a component based representation of these references, with one reference per component (sequential process). Then the total number of references can be limited to the number of clocks plus the number of components, which corresponds to a generalization of clock zones (one component!) and also to the local time approach of [8].

5. A language theoretic view

In the previous section we built a symbolic automaton for $L_I(\mathcal{A})$ based on event zones instead of clock zones. However, this construction does not include anything corresponding to the “greatest constant abstraction” [3], the state space of the automaton may thus be infinite. Therefore, the question arises whether event zones, like clock zones, can yield a finite automaton via some abstraction. We investigate this issue from a language theoretical point of view first.

Given a language $L \subseteq \Sigma^*$ the Myhill–Nerode right-congruence $\simeq_L$ is defined as $\{(u, v) \mid \forall w \in \Sigma^*. uw \in L \Leftrightarrow vw \in L\}$. The congruence classes of $\simeq_L$ define the minimal

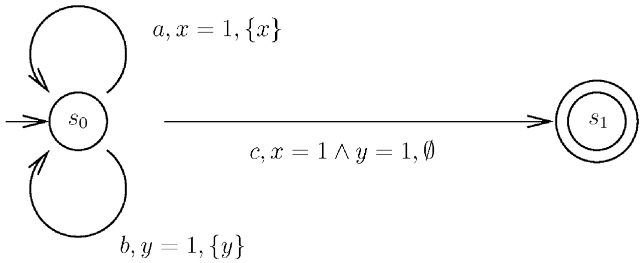

Fig. 2. An automaton $\mathcal{A}$ with $L_I(\mathcal{A})$ non-regular.

deterministic complete automaton of a language, hence it is finite iff L is regular. For our study, we consider a preorder version (called precongruence) of $\simeq_L$:

$$\lesssim_L = \{(u, v) \mid \forall w \in \Sigma^*.uw \in L \Rightarrow vw \in L\}.$$

Obviously $u \simeq_L v$ iff $u \lesssim_L v$ and $v \lesssim_L u$.

For the languages $L_I(\mathcal{A})$ and $L_N(\mathcal{A})$, we denote these relations by $\simeq_I$ and $\lesssim_I$, and $\simeq_N$ and $\lesssim_N$. By definition, the *index* of a preorder $\lesssim$ is the index of the equivalence relation $\lesssim \cap \gtrsim$. As mentioned previously, the preorder $\lesssim_{EZ}$ is defined on words by $u \lesssim_N v$ iff $(s_u, Z_u) \lesssim_{EZ} (s_v, Z_v)$. By definition, the *index* of a preorder $\lesssim$ is the index of the equivalence relation $\lesssim \cap \gtrsim$.

Why event zones have no finite abstraction? The classical clock zone approach can be explained using the three relations: $\lesssim_Z$ the zone inclusion relation without abstraction (and of infinite index), $\lesssim_{ZA}$ the zone inclusion relation with abstraction (that are extended to words in the same way as $\lesssim_{EZ}$), and $\lesssim_N$ as above and the following inclusions

$$\lesssim_Z \subseteq \lesssim_{ZA} \subseteq \lesssim_N.$$

These relations are precongruences and $\lesssim_{ZA}$ can be understood as a pragmatic implementation of $\lesssim_N$. Improvements of zone automata by better abstract interpretation like [4,11] are a way of pushing $\lesssim_{ZA}$ closer to $\lesssim_N$. However, the finiteness of $\lesssim_{ZA}$ implies the finiteness of $\lesssim_N$, which conversely is a precondition for the existence of a finite abstraction.

Reversing the argument for $\lesssim_I$, the following proposition shows that no finite state automaton for $L_I(\mathcal{A})$ can exist and there is no use in trying to generalize the known abstractions for $L_N(\mathcal{A})$ to event zones:

Proposition 12. *There exists a timed automaton and an independence relation I such that $\lesssim_I$ and $\simeq_I$ are of infinite index.*

Proof. Let us consider the timed automaton: $I = \{(a, b), (b, a)\}$ respects conditions (i) and (ii) for independence relations. $L_N(\mathcal{A}) = (ab + ba)^*c$ is a regular language, but $L_I(\mathcal{A}) = \{uc \mid |u|_a = |u|_b\}$ is not a regular language. More precisely, for any $i, j \in \mathbb{N}$ it holds that $a^i \lesssim_I a^j$ iff $i = j$. $\square$

This observation shows that the naïve hope of generalizing partial order reductions to timed automata by solving the commutation problem is bound to fail. A possible solution is to set (severe) restrictions on the class of systems [8] so that the languages remain finite state. But even under these restrictions, the index of $\simeq_I$ is often significantly bigger than the index of $\simeq_N$ (the smallest possible one), questioning the relevance of partial order reductions for timed automata. However, we show below that $\simeq_{EZ}$ can be very useful to check the emptiness of timed automata, which does not require exploring all equivalence classes of $\simeq_I$.

The relations $\simeq_{IN}$ *and* $\lesssim_{IN}$: At this point, we introduce a new pair of relations $\simeq_{IN}$ and $\lesssim_{IN}$ that aim at combining the best of $\simeq_N, \lesssim_N$ (finite index) and of $\simeq_I, \lesssim_I$ (compatibility with $\simeq$). We define $\lesssim_{IN}$ by $u \lesssim_{IN} v$ iff

$$\forall w \in \Sigma^*[(\exists u'\ u' \simeq u \wedge u'w \in L_N(\mathcal{A})) \Longrightarrow (\exists v'\ v' \simeq v \wedge v'w \in L_N(\mathcal{A}))]$$

The relation $\lesssim_{IN}$, while easily seen to be a preorder, is not a precongruence: let $\mathcal{A}$ be the automaton of Fig. 2, then $aa \lesssim_{IN} aaa$ since there is no way to extend either word to obtain a word in $L_N(\mathcal{A})$, but $aab \not\lesssim_{IN} aaab$, since $aab \simeq aba$ and $aba \cdot bc \in L_N(\mathcal{A})$, whereas for no permutation of *aaab* the extension *bc* yields a normal realization.

The relationships between $\lesssim_I, \lesssim_{IN}, \lesssim_{EZ}$ and $\lesssim_N$ (summarized in Fig. 3 below) provide the solution of the emptiness problem. However, these relationships are subtle and we introduce a technical tool the "separator action" \$ to investigate this issue.

The separator action \$: Let $\mathcal{A} = (\Sigma, S, s_0, \rightarrow, F)$ be a timed automaton, I an independence relation for $\mathcal{A}$. Let $\$ \notin \Sigma$ be a special action symbol, called the "separator action". The automaton $\mathcal{A}_\$ = (\Sigma \uplus \{\$\}, S, s_0, \rightarrow \cup \rightarrow_\$, F)$ is obtained from $\mathcal{A}$ by adding the separator transitions, $\rightarrow_\$ = \{(s, \$, \text{true}, \emptyset, s) \mid s \in S\}$. The separator action adds self loops to all states that do not refer to clocks and do not modify the states. While \$ structurally would allow to be independent of any other action, we choose to the contrary to generalize the independence relation I to $\Sigma \cup \{\$\}$ without extending it, i.e. $\$Da$ for all $a \in \Sigma \cup \{\$\}$: the separator depends of everything else, which is precisely its technical use for us.

Let $\lesssim_N^\$, \simeq_N^\$, \lesssim_I^\$, \simeq_I^\$, \lesssim_{IN}^\$, \simeq_{IN}^\$$ be the same relations as above, but for $\mathcal{A}_\$$ rather than $\mathcal{A}$. The important properties of $\mathcal{A}_\$$ are summarized in the following lemma.

Lemma 13. (1) $u\$v \in L_N(\mathcal{A}_\$)$ *iff* $uv \in L_N(\mathcal{A}_\$)$.

(2) *For* $u \in \Sigma^*$ *it holds that* $u \in L_I(\mathcal{A})$ *iff* $u \in L_I(\mathcal{A}_\$)$, $u \in L_N(\mathcal{A})$ *iff* $u \in L_N(\mathcal{A}_\$)$.

(3) $\lesssim_N^\$ \cap \Sigma^* \times \Sigma^* = \lesssim_N$.

(4) $\lesssim_I^\$ \cap \Sigma^* \times \Sigma^* \subseteq \lesssim_I$.

(5) $\lesssim_{IN}^\$ \cap \Sigma^* \times \Sigma^* = \lesssim_{IN}$.

(6) $u \lesssim_{IN}^\$ v$ *iff* $u\$ \lesssim_I^\$ v\$$.

(7) $\lesssim_I^\$ \subseteq \lesssim_{IN}^\$$.

(8) $\lesssim_{EZ}^\$ \cap \Sigma^* \times \Sigma^* = \lesssim_{EZ}$.

Proof of (1). $\Rightarrow$ direction: assume that $u\$v \in L_N(\mathcal{A}_\$)$. Then $u = a_1 \cdot \ldots \cdot a_n$, $v = b_1, \ldots, b_m$ and the word $a_1 \cdot \ldots \cdot a_n\$b_1, \ldots, b_m$ is realizable, i.e. there are time stamps $t_1, \ldots, t_n, t_\$, t'_1, \ldots, t_m$ such that $(a_1, t_1) \cdot \ldots \cdot (a_n, t_n)(\$, t_\$)(b_1, t'_1), \ldots, (b_m, t'_m)$ is a normal word. Therefore $(a_1, t_1) \ldots (a_n, t_n)(b_1, t'_1), \ldots, (b_m, t'_m)$ is a normal word and $uv \in L_N(\mathcal{A}_\$)$.

$\Leftarrow$ direction: assume that $uv \in L_N(\mathcal{A}_\$)$. Then there exist time stamps such that $(a_1, t_1) \cdots (a_n, t_n)(b_1, t'_1), \ldots, (b_m, t'_m)$ is a normal word, i.e. $t_1 \leqslant \cdots \leqslant t_n \leqslant t'_1 \leqslant \cdots \leqslant t'_m$. Choose $t'_\$$ such that $t_n \leqslant t'_\$ \leqslant t'_1$ to get a normal realization of $u\$v$.

Proof of (2). Permutation of independent actions do not depend on action that does not occur in the word.

Proof of (3). $\Rightarrow$ direction: assume that $u \lesssim_N^\$ v$ where $u, v \in \Sigma^*$. By definition $w \in (\Sigma \cup \{\$\})^*$ and $uw \in L_N(\mathcal{A}_\$) \implies vw \in L_N(\mathcal{A}_\$)$. Let $w = w_1\$w_2\$\ldots\$w_m$, where the w_i's are words of Σ^*. By (1) $uw \in L_N(\mathcal{A}_\$)$ iff $uw_1w_2\ldots w_m \in L_N(\mathcal{A}_\$)$. By definition of $\mathcal{A}_\$$, $uw_1w_2\ldots w_m \in L_N(\mathcal{A}_\$)$ implies $uw_1w_2\ldots w_m \in L_N(\mathcal{A})$ (since $u \in \Sigma^*$). The same holds for v, therefore $u \lesssim_N^\$ v$ implies $u \lesssim_N v$.

$\Leftarrow$ direction: assume that $u \lesssim_N v$ where $u, v \in \Sigma^*$. By definition of $\lesssim_N$, $w \in \Sigma^*$ and $uw \in L_N(\mathcal{A})$ implies $vw \in L_N(\mathcal{A})$. We prove that $w_\$ \in (\Sigma \cup \{\$\})^*$ and $uw_\$ \in L_N(\mathcal{A}_\$)$ implies $vw \in L_N(\mathcal{A}_\$)$. By repeated application of (1) $uw_\$ \in L_N(\mathcal{A}_\$)$ iff $uw \in L_N(\mathcal{A}_\$)$ where w is $w_\$$ stripped of \$. By definition of $(\mathcal{A}_\$)$, $uw \in L_N(\mathcal{A}_\$)$ iff $uw \in L_N(\mathcal{A})$. By hypothesis, $uw \in L_N(\mathcal{A})$ implies $vw \in L_N(\mathcal{A})$. By (2) $vw \in L_N(\mathcal{A})$ iff $vw \in L_N(\mathcal{A}_\$)$. By repeated application of (1) $vw \in L_N(\mathcal{A}_\$)$ iff $vw_\$ \in L_N(\mathcal{A}_\$)$, i.e. $u \lesssim_N^\$ v$.

Proof of (4). Assume that $\forall w \in (\Sigma \cup \{\$\})^*, uw \in L_I(\mathcal{A}_\$) \Longrightarrow vw \in L_I(\mathcal{A})$. Restricting to $w \in \Sigma^*$, we have $uw \in L_I(\mathcal{A}) \Longrightarrow vw \in L_I(\mathcal{A}_\$)$. By (2) we get $u \lesssim_I v$.

Proof of (5). $\Rightarrow$ direction: assume $u \lesssim_{IN}^\$ v$.

Let w such that $\exists u'\ u' \simeq u \wedge u'w \in L_N(\mathcal{A})$. By (1) and (2) $u'\$w \in L_N(\mathcal{A}_\$)$. By definition $\exists v'\ v' \simeq v \wedge u'\$w \in L_N(\mathcal{A}_\$)$. By (1) and (2) $u'w \in L_N(\mathcal{A})$, i.e. $u \lesssim_{IN} v$.

$\Leftarrow$ direction: assume $u \lesssim_{IN}^\$ v$.

Let $w_\$ \in (\Sigma \cup \{\$\})^*$ s.t. $\exists u'\ u' \simeq u \wedge u'w_\$ \in L_N(\mathcal{A}_\$)$. Let $w_\$ = w_1\$\ldots\$w_n$ and $w = w_1 \ldots w_n$. By (1), (2) $u'w_\$ \in L_N(\mathcal{A}_\$)$ iff $u'w \in L_N(\mathcal{A})$. By hypothesis $\exists v'\ v' \simeq v \wedge v'w \in L_N(\mathcal{A})$. By (1), (2) $v'w \in L_N(\mathcal{A})$ iff $v'w_\$ \in L_N(\mathcal{A}_\$)$ i.e. $u \lesssim_{IN}^\$ v$.

Proof of (6). $\Rightarrow$ direction: assume that $u \lesssim_{IN}^\$ v$. Let w such that $u\$w \in L_I(\mathcal{A}_\$)$. By Theorem 6, and since \$ depends of any action, there exists u', w' such that $u \simeq u$, $w' \simeq w$ such that $u'\$w' \in L_N(\mathcal{A}_\$)$. By definition of $\lesssim_{IN}$, there exists $v' \simeq v$ such that $v'\$w' \in L_N(\mathcal{A}_\$)$. By Theorem 6 again, we get $v\$w \in L_I(\mathcal{A}_\$)$ yielding $u\$ \lesssim_I^\$ v\$$.

$\Leftarrow$ direction: assume that $u\$ \lesssim_I^\$ v\$$.

Let $w = a_1 \ldots a_n$ such that $\exists u\ u' \simeq \wedge u'w \in L_N(\mathcal{A}_\$)$. Let $w' = \$a_1\$\ldots\$a_n$. By construction the equivalence class of w' is reduced to w' (the separator forbids any permutation). By (1) $u'w \in L_N(\mathcal{A}_\$)$. By Theorem 6, $uw' \in L_I(\mathcal{A}_\$)$. By hypothesis $vw' \in L_I(\mathcal{A}_\$)$. By Theorem 6, there is some w'' equivalent to vw' such that $w'' \in L_N(\mathcal{A}_\$)$. By construction w'' is some $v'w'$ where $v' \simeq v$. By (1) $v'w' \in L_N(\mathcal{A}_\$)$ implies $v'w \in L_N(\mathcal{A}_\$)$, yielding $u\$ \lesssim_{IN}^\$ v\$$.

Proof of (7). Assume that $u \lesssim_I^\$ v$. We must prove that $u \lesssim_{IN}^\$ v$. By (6) this is equivalent to $u\$ \lesssim_I^\$ v\$$. By definition of $\lesssim_I^\$$, we get that $uw \in L_I(\mathcal{A}_\$) \implies vw \in L_I(\mathcal{A}_\$)$ for all

$w \in (\Sigma \cup \{\$\})^*$. Choose $w = \$w'$ in the previous definition. We get $\forall w \in (\Sigma \cup \{\$\})^*$, $u\$w \in L_I(\mathcal{A}_\$) \Longrightarrow v\$w \in L_I(\mathcal{A}_\$)$ i.e. $u\$ \lesssim_I^\$ v\$$.

Proof of (8). By definition $\lesssim_{EZ}$ and $\lesssim_{EZ}^\$$ depends only of the past. □

The finite index preorder $\lesssim_C$: at this point, we have that

$$\lesssim_{EZ} = \lesssim_{EZ}^\$ \subseteq \lesssim_I^\$ \subseteq \lesssim_{IN}^\$ = \lesssim_{IN}.$$

However, we neither know how to test $\lesssim_I^\$$ nor $\lesssim_{IN}^\$$. Next, we abstract/relax $\lesssim_{EZ}$ in a manner to still respect $\lesssim_{IN}^\$$. More precisely, we give a sufficient criterion for two paths with $u_1 \lesssim_{EZ} u_2$ also to satisfy $u_1 \lesssim_{IN} u_2$. As explained before, constraints in the event zones for a pair of variables/events are pairs $(c, <)$ or $(c, \leqslant)$ where $c \in \mathbb{Z} \cup \{+\infty\}$. Our aim is to abstract constraints where c is finite and above or below a certain threshold. Such abstractions are known for classical timed automata, i.e. for the right precongruence $\lesssim_N$. The abstraction we use here is very closely related to the ones known for $\lesssim_N$.

Let (s_1, Z_1) and (s_2, Z_2) be two symbolic states and let $(s_1', Z_1') = (s_1, Z_1) \odot \$$, $(s_2', Z_2') = (s_2, Z_2) \odot \$$.

We say $(s_1, Z_1) \lesssim_C (s_2, Z_2)$ (or (s_2, Z_2) *catchup simulates* (s_1, Z_1)) iff $s_1 = s_2$ and

- either (s_1', Z_1') inconsistent,
- or both (s_1', Z_1'), (s_2', Z_2') are consistent and for all $\alpha \in X$, $\beta \in X \cup \{\$\}$ one of (a), (b), (c) holds: [4]
  - (a) $cf(Z_1')[\alpha, \beta] \leqslant cf(Z_2')[\alpha, \beta]$
  - (b) $cf(Z_1')[\alpha, \$] < (\prec, -c)$ and $cf(Z_2')[\alpha, \$] < (\prec, -c)$ for the greatest non-trivial upper bound "$\alpha \prec c$" in any constraint ϕ_a for the clock α. [5]
  - (c) $cf(Z_1')[\alpha, \beta] \geqslant (\prec, d)$ and $cf(Z_2')[\alpha, \beta] \geqslant (\prec, d)$ where $(\prec, d)$ is the greatest lower bound of clock constraints $\beta \succ d$ if β is a clock and if that bound exists, otherwise $(\prec, d) = (\leqslant, 0)$.

As with $\lesssim_{EZ}$, we define $u \lesssim_C v$ if $(s_u, Z_u) \lesssim_C (s_v, Z_v)$.

Moreover $u \simeq_C v$ (catchup equivalent) iff $u \lesssim_C v$ and $v \lesssim_C u$.

The intuition behind the naming *catchup* is that the definition, in particular in the second and third rule, abstracts from event zones extensions that occur in the past of already present events (e.g. events that would have occurred before the separator in the second rule). We consider such events as "late" and "catching up". The second rule addresses resulting bounds of relevance to upper bounds of clocks (upper catchup), the third with respect to lower bounds (lower catchup). The following theorem is the most complex result of this work and gives the theoretical foundation for the algorithm of Section 6.

Theorem 14. $\lesssim_{EZ} \subseteq \lesssim_C \subseteq \lesssim_{IN}^\$$.

---

[4] Recall that for a consistent event zone $Z = (T, \psi, Last)$ by $cf(Z)[\alpha, \beta]$ we denote the strongest constraint for $Last(\alpha) - Last(\beta)$.

[5] Reachable from s_1 without passing over a reset is a potential improvement known from the literature [4].

finite index not (pre)congruence			$\lesssim_C$	$\subseteq$	$\lesssim_{IN} = \lesssim^{\$}_{IN}$		
			$\cup\!\vert$		$\cup\!\vert$		
(pre)congruence not finite index	$\simeq$	$\subseteq$	$\lesssim_{EZ} = \lesssim^{\$}_{EZ}$	$\subseteq$	$\lesssim^{\$}_{I}$	$\subseteq$	$\lesssim_I$

Fig. 3. Summary of timed automata induced preorders on $\Sigma^* \times \Sigma^*$.

Proof (*Inclusion* $\lesssim_{EZ} \subseteq \lesssim_C$). This follows from the fact that $(s_1, Z_1) \lesssim_{EZ} (s_2, Z_2)$ implies $(s_1, Z_1) \odot \$ \lesssim_{EZ} (s_2, Z_2) \odot \$$ and can otherwise be directly read from the definitions of $\lesssim_{EZ}$ and $\lesssim_C$, where only case (a) is relevant.

Inclusion $\lesssim_C \subseteq \lesssim^{\$}_{IN}$. We use the following auxiliary lemma which is actually the key part of the proof:

Lemma 15. *If* $u \lesssim_C v$ *and the event zone* $Z_{v\$w'}$ *is inconsistent for a word* $w' = a_1\$a_2\$\ldots\$a_n$, *then* $Z_{u\$w'}$ *is inconsistent*.

The proof of the lemma is given in appendix. □

Then the proof of the inclusion proceeds as follows:

Let $w = a_1 \ldots a_n$ and $u' \simeq u$ such that $u'\$w \in L_N(\mathcal{A}_\$)$. By Lemma 13 then also $u'\$w' \in L_N(\mathcal{A}_\$)$, $u'\$w'$ realizable, $u\$w'$ I-realizable and $Z_{u\$w'}$ consistent. Hence $Z_{v\$w'}$ is consistent by (Fact 1), $v\$w'$ I-realizable and there exists $v' \simeq v$ such that $v'\$w'$ is realizable and has an accepting run. By Lemma 13, the same holds for $v'\$w$ and $v'\$w \in L_N(\mathcal{A}_\$)$. □

Proposition 16. *The index of* $\simeq_C$ *is finite. If n is the number of clocks and K is the biggest constant mentioned in constraints, then it is smaller than* $|S|(4K+3)^{n(n+1)}$.

Proof. Two words w_1, w_2 are distinguished by $\simeq_C$ iff for the corresponding symbolic states $(s_1, Z_{w_1\$}), (s_2, (Z_{w_2\$})$ either $s_1 \neq s_2$ or if the zones can be distinguished according to $\lesssim_C$. The latter are seen as rectangular matrices of size $n(n+1)$ and distinguishable entries range between $(-K, <)$ and $(K, \leqslant)$. □

Of course, the bound is an upper bound which simply gives an idea of the order of magnitude.

A summary of the relations considered in this section, with their properties under restriction to $\Sigma^* \times \Sigma^*$, is given in Fig. 3. In the next section, we will see how to exploit these relations algorithmically for emptiness checking.

6. A new algorithm for emptiness checking of timed automata

Now we combine the relation $\lesssim_C$ (that one may consider as an implementation of the finite index preorder $\lesssim_{IN}$) and the infinite event zone automaton of Section 4 to get an algorithm for deciding $L_I(\mathcal{A}) = \emptyset$. This generic exploration algorithm given in Fig. 1 is described abstractly without imposing unnecessary detail. [6] It manipulates four sets of pairs $([(s, Z)], w)$ of symbolic states and witness path labelings w such that $(s, Z) \simeq_{EZ} (s_w, Z_w)$:

- the "white" set contains pairs not yet visited;
- the "gray" set contains pairs waiting to be explored (sometimes it is referred to as the "waiting list" in the literature);
- the "black" set contains pairs that have been visited and explored (sometimes referred to as "past list");
- the "red" set contains pairs that are visited but not explored because there is a greater symbolic state $\lesssim_C$ in a pair belonging to gray or black.

The colored sets are used as invariants in the correctness proof. Compared to similar depth first search algorithms on finite graphs (see e.g. [9]) where in the end all vertices are black (hence all vertices have been explored), the color "red" is added and allows to explore a bounded fragment of the symbolic state space only, leaving an infinity of vertices white (hence unexplored) while the search is still complete w.r.t. the desired property (i.e. emptiness of $L_I(\mathcal{A})$).

Algorithm 1. Generic exploration algorithm

```
Gray ← {([(s_ε, Z_ε)], ε)}
Black ← ∅
Red ← ∅
while Gray ≠ ∅ do
  Choose ([(s, Z)], w) ∈ Gray
  Gray ← Gray \ {([(s, Z)], w)}
  Black ← Black ∪ {([(s, Z)], w)}
  for all w′ = wa with (s′, Z′) = (s, Z) ⊙ a consistent  do
    if ∃([(s′, Z″)], w″) ∈ Black ∪ Gray. and (s′, Z′) ≲_C (s′, Z″)
    /* or weaker (s′, Z′) ≃_C (s′, Z″) */
     then
      Red ← Red ∪ {([(s′, Z′)], w′)}
    else
      if  s′ ∈ F then
        return  "witness(w′)"
      end if
    end if
  end for
end while
return "empty"
```

Theorem 17. *For a timed automaton $\mathcal{A}$, Algorithm* 1 *terminates and yields a witness $w \in L_I(\mathcal{A})$ iff $L_I(\mathcal{A}) \neq \emptyset$ otherwise returns* "*empty*".

[6] Special thanks go to Walter Vogler for suggesting this presentation of the generic algorithm and of the correctness proof!

Proof. The proof is based on the following claims:

(Invariant 0) For $([(s, Z)], w) \in \mathsf{Black} \cup \mathsf{Gray} \cup \mathsf{Red}$ we have that $(s, Z) \simeq_{EZ} (s_w, Z_w)$ and *w* is I-realizable.

(Invariant 1) At the beginning of the while-loop, for any two $([(s_1, Z_1)], w_1)$, $([(s_2, Z_2)], w_2) \in \mathsf{Black} \cup \mathsf{Gray}$ and $w_1 \simeq_C w_2$ we have $w_1 = w_2$.

(Termination) The number of while-iterations is limited by the index of $\simeq_C$ (number of catchup incomparable zones).

The iterations of the for-loops inside the while loop is limited by the branching degree of $\rightarrow$ (number of successors of states in the timed automaton).

(Invariant 2) At the beginning of the while loop, all I-realizable successors $([(s', Z')], wa)$ of a black $([(s, Z)], w)$ are colored (black, gray or red).

(Invariant 3) For each red $([(s, Z)], w)$ there exists $([(s, Z')], w')$, gray or black, with $w \lesssim_C w'$.

(Invariant 4) For $([(s, Z)], w)$ colored (black, gray or red), $s \notin F$.

(Witness) A returned witness really belongs to $L_I(\mathcal{A})$ (as it is I-realizable and leads to a final state).

(No witness) If no witness is returned then $L_I(\mathcal{A}) = \emptyset$.

The claimed invariants (0–4), (Termination) and (Witness) are easy to check. The interesting and more difficult to prove claim is (No witness).

Let us assume that indeed $L_I(\mathcal{A}) \neq \emptyset$, but the algorithm terminates with "*empty*". Then we know also that $L_N(\mathcal{A}) \neq \emptyset$. Let $w \in L_N(\mathcal{A})$ and $w = w_1 w_2$ such that $([(s_1, Z_1)], w_1')$ black for some $w_1' \simeq w_1$ and $|w_2|$ minimal.

$w_2 = \varepsilon$ is not possible, since then $s_1 \in F$ contradicting Invariant (4), so $w_2 = a w_2'$. Since $([(s_1, Z_1)], w_1')$ is black, its successor $([(s', Z')], w_1' a)$ must be colored. At termination, there are no gray traces left, and $([(s', Z')], w_1' a)$ black would contradict the assumption of $|w_2|$ minimal. It follows that $([(s', Z')], w_1' a)$ is red.

Then there must exist a gray (excluded at termination) or black $([(s', Z'')], w_1'')$ with $w_1' a \lesssim_C w_1''$ and hence also $w_1' a \lesssim_{IN} w_1''$. By definition of $\lesssim_{IN}$ and Proposition 5 this implies that for some $w_1''' \simeq w_1''$, $w_1''' w_2' \in L_N(\mathcal{A})$, again contradicting the assumed minimality of w_2. $\square$

The exploration algorithm is just the central component of the verification system. If a witness *w* is actually returned, an I-normal timed word (w, t) should actually be computed (this is possible with the Bellman–Ford algorithm in a time quadratic in $|w|$) and finally "sorted" (see Proposition 5) thus providing a meaningful witness $(w', t') \in L_T(\mathcal{A})$.

Algorithm 1 is a way to explore a sufficient fragment of the event zone automaton in order to detect emptiness.

In the search we propose, the stopping criterion is not whether a certain symbolic state has been visited, but if there exists a previously visited state that "catchup simulates" (or that is equivalent to it) it.

An important difference in the actual implementation of this algorithm compared to the classical zone approach is that the gray set and the black set have different representations. The representation of the black set is the same as in the classical case (clock zones), whereas the gray set requires storing and exploring event zones. This can result in significant memory or computing requirements for the gray set. However, for search strategies like depth first

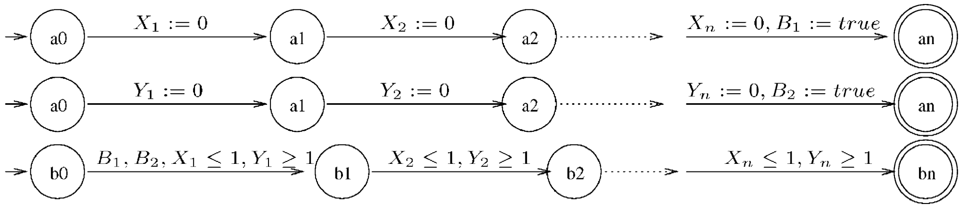

Fig. 4. The diamond example with 2n clocks.

or breadth first, the gray set on the whole can be stored in a data structure with very good locality properties. This suggests placing (parts of) the gray set in secondary memory (like a hard disk).

Experiments For practical evaluation, we have built a tool, *ELSE*, which is currently in prototype status. It allows both classical semantics (corresponding to clock zones) and event zones, implementing Algorithm 1. We measure reductions in terms of number of explored symbolic states (where feasible for the prototype) and do not compare execution times. [7] Also, since we do not include static analysis improvements for less clock zones, comparison with state numbers obtained by highly optimized tools like UppAal is not meaningful. We chose to compare the two modes of the same base implementation. Where there are gains, they should be complementary to gains by better static analysis.

We consider four examples. The first—artificial—example is the *diamond example* of Fig. 4: a network of two automata that just reset clocks in a fixed order and when both are done, a third—observer automaton—tests some properties of the interleavings. The product automaton has a quadratic number of reachable states and $\lesssim_I$ is actually finite for this example with a quadratic index. All accepted paths in $L_I(\mathcal{A})$ are equivalent. Clock zone automata, however, have to distinguish all possible shuffles of the resets of clocks X_i and Y_j and contain a lot of dead ends. So this artificial example gives polynomial against exponential growth.

More realistic, the second example is a timed version of the dining philosophers, which yield forks taken if they do not obtain the second fork before a timeout (in order to avoid deadlocks). While both the event zone approach and the clock zone approach yield exponential blowups, the difference between the two is impressing and encouraging for applications with some distribution.

The third example, popular Fischer's protocol [1] is a very unfavorable example, since there is hardly any independence in the models. Still, we report it to show that even in such cases, event zones yield a reduction.

The fourth example is a series of scheduling problems, see for instance [2] for scheduling with timed automata. Whereas it has been noticed that generic abstraction techniques avoid zone splitting when modeling job shop scheduling problems, this is not the case if task deadlines have to be taken into account. The problems are of type *n* jobs, *n* tasks, each task using one of *n* machines. The numbers given are raw, naive exploration of the full state

[7] Since our implementation is optimized for event zones, such a comparison would favor our approach in a questionable manner.

process number	2	3	4	5	6	7	8	9	10	100
Diamond, EZC	19	29	41	55	71	89	109	131	155	3571
Diamond, CZ	56	198	711	2596	9607	35923	135407	–	–	–
Philosophers, EZC	13	48	153	478	1507	4791	15369	49662	161393	–
Philosophers, CZ	13	66	393	2772	23103	223052	2453967	–	–	–
Fischer, EZC	24	209	2048	21077	224536	2480277	–	–	–	–
Fischer, CZ	25	229	2393	26961	322525	4081295	–	–	–	–
Scheduling, EZC	26	435	11509	413326	–	–	–	–	–	–
Scheduling, CZ	33	1094	79089	9645848	–	–	–	–	–	–

Fig. 5. Experimental results.

space without any heuristics. The reduction is explained by the fact that the relative starting times of independent tasks are not recorded in the event zones. For this example we not only observed a significantly lower number of symbolic states but an even stronger bias concerning long zone lists: for the case 5–5–5 the event zone exploration gave at most 120 distinct zones for a single control location, whereas for clock zones this number grows up to 672. This is interesting since the time required for building long zone lists is quadratic in their length and is typically the bottleneck of this type of exploration algorithms. On the negative side we were not able to go beyond 5–5–5 in either approach: in the event zone approach we ran out of memory for the gray set (whereas the black set required reasonable space according to the expected growth). This shows the importance of putting the gray set to secondary memory, a feature we have not implemented yet.

The experimental results are summarized in Fig. 5, where "EZC" stands for exploration with event zone automata and catchup preorder whereas "CZ" stands for clock zone automata. Each case concerns scalable examples with a parameter m (number of clock of each process in the diamond example, number of philosophers, number of processes Fischer protocol).

7. Conclusions and future work

We have established a novel formal framework for emptiness checking of timed automata based on partial order semantics. Moreover, we have implemented it in form of the currently experimental ELSE tool [27] and found very encouraging results. There are several open ends to our approach, current and future work.

From the point of view of applicability, the modeling framework requires the inclusion of state invariants, see detailed discussion below. Based on this framework we work on a revision of ELSE that will be able to analyze UppAal models.

A second direction of interest is the inclusion of actual partial order reduction algorithms. A naïve integration is not possible for many types of reduction as they are almost always incompatible with the definition of our abstraction, $\simeq_C$. However, the sleep set reduction [16] might be compatible with the correctness proof of the algorithm. The sleep set reduction has the particularity of not removing any reachable states but of avoiding the double exploration of traces, thereby significantly reducing the number of transitions. Since the computation of transitions is expensive for timed automata, the sleep set reduction might therefore speed up significantly the exploration without effects on memory consumption. We will explore this in the near future.

A question of theoretical interest is a better understanding of the link between the preorder chosen as cutoff criterion and classical "clock zones". Indeed, an event zone abstracted by $\simeq_C$ is related to a clock zone by a change of variables, which is more than a syntactic coincidence: it seems that the latter clock zone corresponds to the "convex hull" [12] of the clock zones of all of the equivalent sequences, which would explain the savings obtained by the method.

Integrating local state invariants: Although this is work in progress, it seems relevant to the appreciation of our setting to know whether it can be extended to incorporate state invariants. Here is a sketch.

State invariants are conditions limiting the passage of time during a stay at a state without taking a transition. Technically, the timed automaton is extended by an assignment *Inv*: $\mathcal{A} = (\Sigma, S, s_0, \rightarrow, F, Inv)$ such that for each state $s \in S$, the *invariant* $Inv(s)$ is a set of upper bounds $x \leqslant c$ or $x < c$ that must be satisfied while staying in this state. While this automata model is not in principle more expressive (intuitively, state invariants can be shifted to transition guards), the interest in state invariants is due to their application in *networks of timed automata*. For example: a timeout in a communication protocol represents the reaction of a process to an event of a partner process that *does not occur* within a given interval. This can be modeled by stating that the process may remain in the waiting state only until the duration of the timeout has elapsed and will then do a transition corresponding to the timeout event.

The question we discuss is the extension of independence to networks of timed automata with state invariants.

Consider for instance two transitions $s_1 \overset{(a_1,\phi_1,C_1)}{\rightarrow} s_1'$ of $\mathcal{A}_1$ and $s_2 \overset{(a_2,\phi_2,C_2)}{\rightarrow} s_1'$ of $\mathcal{A}_2$ in some network $\mathcal{A}_1 \times \mathcal{A}_2 \times \cdots$. In order for a_1 and a_2 to be independent in the synchronous product, the following conditions must be satisfied:

- a_1 and a_2 must be performed by distinct automata and must not be synchronized (by communication or shared variables), as is usual for defining independence based on concurrency.
- The conditions of Section 3 concerning clock constraints and resets must be satisfied (no read/write conflicts).
- Additionally, no clock x in C_1 must be constraint in $Inv(s_2)$ and vice versa (where we assume not having to deal with the paradox of arriving in states with their invariant already violated, which can be trivially excluded by adding to ϕ_2 all constraints $x \leqslant c$ or $x < c$ of $Inv(s_2')$ where $x \notin C_2$).

When performing a transition a in the event zone approach, then as before we add all clock constraints of the partner transitions, additionally the local invariants of the partners, and *all constraints* $x \leqslant c$ *or* $x < c$ *belonging to some* $Inv(s_i)$ *of the component i* (*partner or not*) *which is reset by some partner of a*.

While this construction already ensures commutativity of independent transitions in the event zone automaton, in order to guarantee that discrete states reachable in the event zone automaton are also reachable by normal runs (cf. Proposition 5), accepted runs must be restricted to terminate in (final) states with the invariants of all components satisfied (this is similar to resynchronization in [8]).

Acknowledgements

We thank Victor Braberman, Sergio Yovine, Stavros Tripakis, Oded Maler, Eugene Asarin, Yasmina Abdeddaim, Bengt Johnsson and Rom Langerak for discussions about the challenging topic. Many thanks go to Walter Vogler and an anonymous reviewer for their helpful constructive critique.

Appendix

Proof that $\lesssim_{EZ}$ is a precongruence

Recall that by definition, for two words u, v we have $u \lesssim_{EZ} v$ if and only if $Z_u \lesssim_{EZ} Z_v$ where Z_u and Z_v are, respectively, the event zone associated to u and Z_v the one associated to v.

As a consequence, to prove that $\lesssim_{EZ}$ is a precongruence, it is sufficient to show the following property.

Given two symbolic states such that $(s_1, Z_1) \lesssim_{EZ} (s_2, Z_2)$, an action a such that $(s_1, Z_1) \odot a$ is defined, we prove that $(s_1, Z_1) \odot a \lesssim_{EZ} (s_2, Z_2) \odot a$ holds.

Proof. Let $Z'_1 = (s'_1, (T'_1, \psi'_1, Last'_1)) = (s_1, (T_1, \psi_1, Last_1)) \odot a$ and similarly let $Z'_2 = (s'_2, (T'_2, \psi'_2, Last'_2)) = (s_2, (T_2, \psi_2, Last_2)) \odot a$.

The first condition in the definition of $\lesssim_{EZ}$ is immediate as $s_1 = s_2$ and as timed automata are deterministic there is at most one transition labeled by a, i.e. $s'_1 = s'_2$.

For the second condition in the definition of $\lesssim_{EZ}$, let us consider the cases where:

(1) Z_1 and Z_2 are both inconsistent. Then so are Z'_1 and Z'_2;

(2) Z_1 is inconsistent and Z_2 is consistent. Then Z'_1 is inconsistent and we are done whether Z'_2 is consistent or not;

(3) Z_1 and Z_2 are both consistent. We have two sub-cases to consider:
   (3.1) Z'_1 is inconsistent. We are in the same case as in (2);
   (3.2) Z'_1 is consistent.

We show that for each sequence of time stamp constraints (concerning last occurrences) in Z'_2 there exists a corresponding tighter sequence in Z'_1. This will imply consistency of Z'_2 and the conditions for $\lesssim_{EZ}$ concerning $cf(Z'_1)$ and $cf(Z'_2)$.

For this purpose, let us note t^1_a the fresh time stamp variable introduced for the extension of Z_1 by a that is $T'_1 = T_1 \cup \{t^1_a\}$. Similarly let us note t^2_a the one introduced for the extension of Z_2 that is $T'_2 = T_2 \cup \{t^2_a\}$.

First, we show the following technical lemma.

Lemma 18. *Let $t'_1, t'_2, \ldots, t'_k$ be a sequence of time stamp variables in Z'_2 such that $t'_1 = Last_2(\alpha)$ and $t'_k = Last_2(\beta)$ where $\alpha, \beta \in X \cup \Sigma$. Then there exists a sequence $t_1, t_2, \ldots, t_p$*

of time stamp variables in Z'_1 *such that* $t_1 = Last_1(\alpha)$, $t_p = Last_1(\beta)$ *and*

$$((\prec_{1,2}, c_{1,2}) + (\prec_{2,3}, c_{2,3}) + \cdots + (\prec_{p-1,p}, c_{p-1,p}) \\ \leqslant (\prec'_{1,2}, c'_{1,2}) + (\prec'_{2,3}, c'_{2,3}) + \cdots + (\prec'_{k-1,k}, c'_{k-1,k}),$$

where

for all $m = 1, \ldots, p-1$ *the constraint* $t_m - t_{m+1} \prec_{m,m+1} c_{m,m+1}$ *is in* ψ'_1
for all $m = 1, \ldots, k-1$ *the constraint* $t'_m - t'_{m+1} \prec'_{m,m+1} c'_{m,m+1}$ *is in* ψ'_2.

The proof is an induction on the number of indices j_m such that $t_{j_m} = t^2_a$.
Basic case: assume that t^2_a does not occur in $t'_1, \ldots, t'_p$.
We have

$$(\prec'_{1,2}, c'_{1,2}) + (\prec'_{2,3}, c'_{2,3}) + \cdots + (\prec'_{k-1,k}, c'_{k-1,k}) \geqslant cf(Z_2)[\alpha, \beta],$$

since all added constraints in Z'_2 with respect to the one in Z_2 involve t^2_a.
By definition of $\lesssim_{EZ}$ we also have

$$cf(Z_2)[\alpha, \beta] \geqslant cf(Z_1)[\alpha, \beta].$$

The two previous relations imply that

$$(\prec'_{1,2}, c'_{1,2}) + (\prec'_{2,3}, c'_{2,3}) + \cdots + (\prec'_{k-1,k}, c'_{k-1,k})$$

is greater than or equal to the minimal sum

$$(\prec_{1,2}, c_{1,2}) + (\prec_{2,3}, c_{2,3}) + \cdots + (\prec_{p-1,p}, c_{p-1,p}) = cf(Z_1)[\alpha, \beta].$$

Inductive step: let us suppose the property is true for every sequence of time stamps $t'_1, t'_2, \ldots, t'_k$ such that there exists at most $n > 0$ indices m with $t'_m = t^2_a$.

The sum $(\prec'_{1,2}, c'_{1,2}) + (\prec'_{2,3}, c'_{2,3}) + \cdots + (\prec'_{k-1,k}, c'_{k-1,k})$ can be split into three parts:
(1) $(\prec'_{1,2}, c'_{1,2}) + \cdots + (\prec'_{l-2,l-1}, c'_{l-2,l-1})$,
(2) $(\prec'_{l-1,l}, c'_{l-1,l}) + (\prec'_{l,l+1}, c'_{l,l+1})$ with $t'_l = t^2_a$,
(3) $(\prec'_{l+1,l+2}, c'_{l+1,l+2}) + \cdots + (\prec'_{k-1,k}, c'_{k-1,k})$.

Otherwise, by the definition of $\odot$, there exist $\gamma, \delta \in X \cup \Sigma$ with $t'_{l-1} = Last_2(\gamma)$, $t'_{l+1} = Last_2(\delta)$ such that
(i) $t'_{l-1} - t^2_a \prec'_{l-1,l} c'_{l-1,l} \in \psi'_2$,
(ii) either γ is an action and $(\prec'_{l-1,l}, c'_{l-1,l})$ is $(\leqslant, 0)$ or γ is a clock and ϕ_a contains an atomic clock constraint $-c'_{l-1,l} \prec'_{l-1,l} \gamma$.

In both cases, by definition of $\odot$ (with action a), ψ'_1 must contain the time stamp constraint $Last_1(\gamma) - t^1_a \prec'_{l-1,l} c'_{l-1,l}$.

Likewise, for δ, $t^2_a - t'_{l+1} \prec'_{l,l+1} c'_{l,l+1}$ in ψ'_2 such that δ is a clock and $\delta \prec'_{l,l+1} c'_{l,l+1}$ is an atomic clock constraint in ϕ_a (δ cannot be an action).

In that case, also ψ'_1 contains the time stamp constraint $t^1_a - Last_1(\delta) \prec'_{l,l+1} c_{l,l+1}$.

Applying the induction hypothesis on (1) and the pair α, γ there exists a sequence $(\prec_{1,2}, c_{1,2}) + \cdots + (\prec_{o-2,o-1}, c_{o-2,o-1}) \leqslant (\prec'_{1,2}, c'_{1,2}) + \cdots + (\prec'_{l-2,l-1}, c'_{l-2,l-1})$ with $t_{o-1} = Last_1(\gamma)$; similarly, for (3) there exists a sequence

$$(\prec_{o+1,o+2}, c_{o+1,o+2}) + \cdots + (\prec_{p-1,p}, c_{p-1,p}) \\ \leqslant (\prec'_{l+1,l+2}, c'_{l+1,l+2}) + \cdots + (\prec'_{k-1,k}, c'_{k-1,k}) \text{ with } t_{o+1} = Last_1(\delta).$$

By setting $t_{i_o} = t^1_a$ and

$$(\prec_{o-1,o}, c_{o-1,o}) = (\prec'_{l-1,l}, c'_{l-1,l}) \text{ and } (\prec_{o,o+1}, c_{o,o+1}) = (\prec'_{l,l+1}, c'_{l,l+1}),$$

we obtain the desired sequence which completes the proof of the lemma. □

Now we show the consistency of Z'_2.

Let $(\prec', c') = (\prec'_{1,2}, c'_{1,2}) + (\prec'_{2,3}, c'_{2,3}) + \cdots + (\prec'_{k-1,k}, c'_{k-1,k})$ where $t'_m - t'_{m+1} \prec'_{m,m+1} c'_{m,m+1}$ in ψ'_2 for all $m = 1, \ldots, k-1$ and $t'_1 = t'_k$. Then

(1) either there exists m with $t'_m = Last_2(\alpha)$ for some α in which case we can suppose $t'_1 = t'_k = Last_2(\alpha)$. By Lemma 18 there exists in Z'_1 a corresponding sequence $(\prec_{1,2}, c_{1,2}) + (\prec_{2,3}, c_{2,3}) + \cdots + (\prec_{p-1,p}, c_{p-1,p}) \leqslant (\prec', c')$ with $t_1 = t_j = Last_1(\alpha)$. Since Z'_1 is consistent this implies $(\prec', c') \geqslant (\leqslant, 0)$.

(2) or no such α exists, then the above constraints all belong to ψ_2 and by consistency of Z_2 we get $(\prec', c') \geqslant (\leqslant, 0)$.

For the comparison of $cf(Z'_1)[\alpha, \beta]$ and $cf(Z'_2)[\alpha, \beta]$ we have to distinguish four cases depending on whether $Last_2(\alpha) = Last'_2(\alpha)$ or not and whether $Last_2(\beta) = Last'_2(\beta)$ or not.

Assume that $Last_2(\alpha) = Last'_2(\alpha)$ and $Last_2(\beta) = Last'_2(\beta)$. By Lemma 18, we obtain immediately $cf(Z'_1)[\alpha, \beta] \leqslant cf(Z'_2)[\alpha, \beta]$.

For the other cases, e.g. $Last'_2(\alpha) = t^2_a$, a reasoning similar to the reasoning done in the induction step of the proof of Lemma 18 applies. □

Proof of Lemma 15. We restate the lemma.

Lemma 15. *If $u \lesssim_C v$ and the event zone $Z_{v\$w'}$ is inconsistent for a word $w' = a_1\$a_2\$\ldots\$a_n$, then $Z_{u\$w'}$ is inconsistent.*

Proof. Assume that $Z_{v\$w'} = (T_2, \psi_2, Last_2)$ is inconsistent. Let $T = T_{v\$} \uplus T_{w'}$ be a partition of the time stamps according to their origin in the word $v\$w'$ and in particular let $t_\$ = Last_{v\$}(\$)$ be the time stamp of the occurrence of \$ after v. Likewise, let $Z_{u\$w'} = (T_1, \psi_1, Last_1)$ and $T_1 = T_{u\$} \uplus T^2_{w'}$ and for convenience we assume that the time stamps on the extensions $T_{w'}, T^2_{w'}$ are identical (if not, a transformation of one of the zones to achieve this yields an isomorphic zone) and that the restriction of the zones to T_w is identical (isomorphic).

Since $Z_{v\$w'}$ is inconsistent, there exists a cycle of time stamps $t_1 \ldots t_k$ with $t_{k+1} = t_1$ and constraints $t_i - t_{i+1} \prec_i c_i$, such that $\sum_{i=1}^{k} (\prec_i, c_i) < (\leqslant, 0)$ (by Proposition 7).

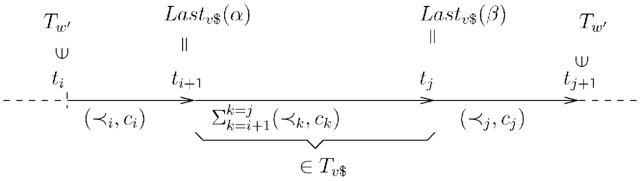

Fig. 6. The sequence in $v\$w'$.

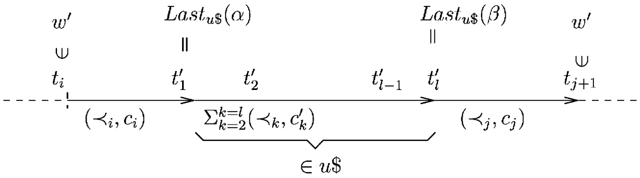

Fig. 7. The sequence in $u\$w'$.

We distinguish three cases:

(i) all $t_1, \ldots, t_k \in T_{v\$}$ then already Z_v inconsistent and Z_u inconsistent by definition of $u \lesssim_C v$, therefore $Z_{u\$w'}$ is inconsistent;

(ii) all $t_1, \ldots, t_k \in T_{w'}$ then the cycle exists isomorphically in $Z_{u\$w'}$ therefore $Z_{u\$w'}$ is again inconsistent;

(iii) $t_1, \ldots, t_k$ alternates between the two sets and it contains sequences $t_i, t_{i+1} \ldots t_j, t_{j+1}$ such that t_i and t_{j+1} are in w', hence also in $u\$w'$ (up to renaming), and $t_{i+1}, t_{i+2}, \ldots, t_{j-1}, t_j$ are in $v\$$.

Case (iii) requires a detailed analysis to construct a negative cycle for $Z_{u\$w'}$. We achieve this goal by showing that there is a path $t_i, t'_1, \ldots, t'_l, t_{j+1}$ in $u\$w'$ that has a shorter length than $t_i, t_{i+1}, \ldots, t_j, t_{j+1}$ in $v\$w$ or that contains a negative cycle (in each case, this yields a negative cycle in $u\$w'$).

For each t_k, t_{k+1} for $k = i, i+1, \ldots, j$ (resp. t'_k, t'_{k+1} for $k = 1, \ldots, l-1$) let $(\prec_k, c_k)$ (resp. $(\prec'_k, c'_k)$) be the relevant constraint. By definition $t_k - t_{k+1} \prec_k c_k$ and $(\prec_k, c_k) < (<, \infty)$ (otherwise the length of the path is not negative).

By definition there exist $\alpha \in X$ and $\beta \in X \cup \Sigma$ so that the constraint $(\prec_i, c_i)$ is due to a constraint $\alpha \prec_i c_i$ for an action occurrence a in w' corresponding to t_i and that $(\prec_j, c_j)$ is either due to a dependency constraint for some action β or to a lower clock constraint $\beta \succ_j -c_j$ where $Last_{v\$}(\beta) = t_j$. The configuration of $v\$w'$ is summarized in Fig. 6.

By definition of *cf*, we have $\sum_{m=2}^{j-1} (\prec_j, c_j) \geqslant cf(Z_{v\$})[\alpha, \beta]$.

By hypothesis $u \lesssim_C v$, then $u\$ \lesssim_C v\$$. We finish the proof by discussing the three possible cases according to the definition of $\lesssim_C$.

Case (a) *holds*: $cf(Z_{u\$})[\alpha, \beta] \leqslant cf(Z_{v\$})[\alpha, \beta]$.

Let $t'_2, \ldots, t'_{p-1}$ realizing the minimal length between $Last(\alpha)$ and $Last(\beta)$ (in $u\$w'$). By definition $\sum_{k=2}^{p-1} (\prec'_k, c'_k) = cf(Z_{u\$})[\alpha, \beta]$.

The configuration in $u\$w'$ is described in Fig. 7.

By hypothesis $cf(Z_{u\$})[\alpha, \beta] \leqslant cf(Z_{v\$})[\alpha, \beta]$.

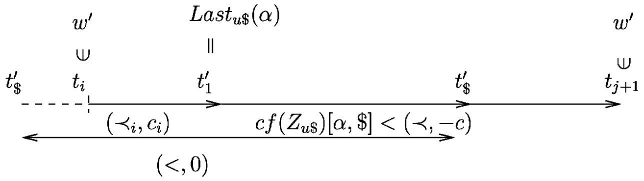

Fig. 8. The sequence in $u\$w'$.

Therefore

$$(\prec_1, c_1) + \sum_{k=2}^{p-1} (\prec'_k, c'_k) + (\prec_j, c_j) \leqslant (\prec_1, c_1) + cf(Z_{v\$})[\alpha, \beta] + (\prec_j, c_j)$$
$$\leqslant (\prec_1, c_1) + \sum_{k=2}^{p-1} (\prec_k, c_k) + (\prec_j, c_j).$$

This yields a shorter path in $u\$w'$ with the same ending points t_i, t_{j+1}.

Case (b) *holds*: then there exists a path in $Z_{u\$}$ from $Last_{u\$}(\alpha)$ to $Last_{u\$}(\$)$ of length $< (\prec_i, -c_i)$. On the other hand, for every $t \in T_{w'}$ (and in particular for t_i) there exists a path of length $(\leqslant, 0)$ from $Last_{u\$}(\$)$ to t. Hence, we obtain a path of length $(\prec_i, -c_i)$ from $Last_{u\$}(\alpha)$ to t_i, closing a negative cycle. This situation is described in Fig. 8 (in the figure the time stamp $t'_\$$ has been duplicated for convenience, but there is actually only one time stamp $t'_\$ = Last_{u\$}(\$)$).

Case (c) *holds*: $cf(Z_{u\$})[\alpha, \beta] \geqslant (\prec, d)$ and $cf(Z_{v\$})[\alpha, \beta] \geqslant (\prec, d)$ where $(\prec, d)$ is the largest clock constraint $\beta \prec d$ for β if β is a clock and if such constraint exists, otherwise $(\prec, d) = (\leqslant, 0)$.

We show that the sequence $t_i, t'_1, t'_\$, t_{j+1}$ in $T_{u\$}$ has a shorter length than the sequence $t_i, t_{i+1}, \ldots, t_j, t_{j+1}$ in $T_{v\$}$.

Firstly, we compute the length of this sequence:

(1) $t'_1 \in T_{u\$}$ and corresponds to an action preceding \$, hence $t'_1 \leqslant t'_\$$,

(2) $t_{j+1} \in T_{w'}$, $t'_\$ \in T_{u\$}$ therefore $t'_\$ \leqslant t_{j+1}$.

Therefore the length is $(\prec_i, c_i) + (\leqslant, 0) + (\leqslant, 0) = (\prec_i, c_i)$.

Secondly we show that this is less than or equal to

$$(\prec_i, c_i) + \sum_{p=i+1}^{j-1} (\prec_p, c_p) + (\prec_j, c_j)$$

which we get by proving that

$$\sum_{p=i+1}^{j-1} (\prec_p, c_p) + (\prec_j, c_j) \geqslant (\leqslant, 0).$$

Depending on the definition of $\lesssim_C$, we discuss two cases:

Sub-case 1: β is a clock and there exists a clock constraint $\beta \succ c$.

The constraint $(\prec_j, c_j)$ comes from a clock constraint $\beta_j \prec -c_j$ and the configuration in $v\$w'$ (not $u\$w'$) is described in Fig. 9.

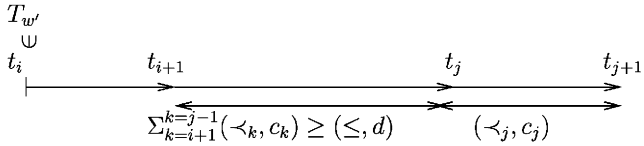

Fig. 9. The sequence in $v\$w'$.

Since $(\prec, d)$ is the maximal value of all pairs $(\prec_k, c_k)$, we have

$$\begin{aligned}\sum_{k=i+1}^{j-1}(\prec_k, c_k) + (\prec_j, c_j) &\geqslant (\prec, d) + (\prec_j, c_j)\\ &\geqslant (\leqslant, 0),\end{aligned}$$

which terminates this case.

Sub-case 2: $\beta = \$$ or β is a clock but there is no constraint $\beta \succ c$.

By definition $(\prec, d) = (\leqslant, 0)$.

The constraint $(\prec_j, c_j)$ must be a normality condition ($t_{j+1} \in T_{w'}$), therefore $(\prec_j, c_j) \geqslant (0, \leqslant)$.

Therefore

$$\sum_{p=i+1}^{j-1}(\prec_p, c_p) + (\prec_j, c_j) \geqslant (\leqslant, 0) + (\leqslant, 0) = (\leqslant, 0)$$

which terminates this case and the proof. □

References

[1] M. Abadi, L. Lamport, An old-fashioned recipe for real time, ACM Transactions on Programming Languages and Systems 16 (5) (1994) 1543–1571.

[2] Y. Abdeddaim, O. Maler, Job-shop scheduling using timed automata, in: Internat. Conf. on Computer Aided Verification (CAV), 2001, pp. 478–492.

[3] R. Alur, D. Dill, A theory of timed automata, Theoret. Comput. Sci. 126 (2) (1994) 183–235.

[4] G. Behrmann, P. Bouyer, E. Fleury, K. Larsen, Static guard analysis in timed automata verification, in: TACAS, Lecture Notes in Computer Science, Vol. 2619, Springer, Berlin, 2003, pp. 254–277.

[5] G. Behrmann, A. David, K.G. Larsen, O.M. ler, P. Pettersson, W. Yi, Uppaal—present and future, in: Proc. of 40th IEEE Conference on Decision and Control, IEEE Computer Society Press, 2001.

[6] G. Behrmann, K. Larsen, J. Pearson, C. Weise, W. Yi, J. Lind-Nielsen, Efficient timed reachability analysis using clock difference diagrams, in: Internat. Conf. on Computer Aided Verification (CAV), Vol. 1633 of Lecture Notes in Computer Science, 1999, pp. 341–353.

[7] W. Belluomini, C. Myers, Verification of timed systems using POSETs, in: Internat. Conf. on Computer Aided Verification (CAV), 1998, pp. 403–415.

[8] J. Bengtsson, B. Jonsson, J. Lilius, W. Yi, Partial order reductions for timed systems, in: Internat. Conf. on Concurrency Theory (CONCUR), Lecture Notes in Computer Science, Vol. 1466, Springer, Berlin, 1998, pp. 485–500.

[9] T. Cormen, C. Leiserson, R. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1990.

[10] D. Dams, R. Gerth, B. Knaack, R. Kuiper, Partial-order reduction techniques for real-time model checking, Formal Aspects Comput. 10 (1998) 469–482.

[11] C. Daws, Optikron: a tool suite for enhancing model-checking of real-time systems, in: Internat. Conf. on Computer Aided Verification (CAV), Lecture Notes in Computer Science, Vol. 1427, Springer, Berlin, 1998, pp. 542–545.

[12] C. Daws, S. Tripakis, Model checking of real-time reachability properties using abstractions, Lecture Notes in Computer Science 1384 (1998) pp. 313. URL citeseer.ist.psu.edu/daws98model.html.

[13] C. Daws, S. Yovine, Reducing the number of clock variables of timed automata, in: IEE Real-Time Systems Symposium, 1996, pp. 73–81.

[14] V. Diekert, G. Rozenberg (Eds.), The Book of Traces, World Scientific, Singapore, 1995.

[15] D.L. Dill, Timing assumptions and verification of finite-state concurrent systems, in: Proc. of the International Workshop on Automatic Verification Methods for Finite State Systems, Lecture Notes in Computer Science, Vol. 407, Springer, Berlin, 1989, pp. 197–212.

[16] P. Godefroid, Partial-order methods for the verification of concurrent systems: an approach to the state-explosion problem, Lecture Notes in Computer Science, Vol. 1032, Springer, Berlin, New York, NY, USA, 1996.

[17] T. Henzinger, X. Nicollin, J. Sifakis, S. Yovine, Symbolic model checking for real-time systems, Inform. and Comput. 111 (1994) 193–244.

[18] K. Larsen, P. Pettersson, W. Yi, Model-checking for real-time systems, in: Fundamentals of Computation Theory, Lecture Notes in Computer Science, Springer, Berlin, 1995, pp. 62–88.

[19] D. Lugiez, P. Niebert, S. Zennou, A partial order semantics approach to the clock explosion problem of timed automata, in: Tools and Algorithms for the Construction and Analysis of Systems, Tenth Internat. Conf., TACAS 2004, Lecture Notes in Computer Science, Vol. 2988, Springer, Berlin, 2004, pp. 296–311.

[20] P. Niebert, M. Huhn, S. Zennou, D. Lugiez, Local first search—a new paradigm in partial order reductions, in: Internat. Conf. on Concurrency Theory (CONCUR), Lecture Notes in Computer Science, Vol. 2154, 2001, pp. 396–410.

[21] F. Pagani, Ordres partiels pour la vérification de systèmes temps réel, Ph.D. Thesis, Centre d'Etudes et de Recherches, Toulouse, 1997.

[22] D. Peled, All from one, one for all: on model checking using representatives, in: Internat. Conf. on Computer Aided Verification (CAV), Lecture Notes in Computer Science, Vol. 697, 1993, pp. 409–423.

[23] A. Valmari, Stubborn sets for reduced state space generation, Application and Theory of Petri Nets, Vol. 2, 1989, pp. 1–22.

[24] A. Valmari, A stubborn attack on state explosion, in: Proc. of the Second International Workshop on Computer Aided Verification, Lecture Notes in Computer Science, Vol. 531, Springer, Berlin, 1990, pp. 156–165.

[25] T. Yoneda, B.-H. Schlingloff, Efficient verification of parallel real-time systems, Formal Methods Syst. Design 11 (2) (1997) 197–215.

[26] S. Yovine, Kronos: a verification tool for real-time systems, Software Tools for Technol. Trans. 1 (1) (1997) 123–133.

[27] S. Zennou, M. Yguel, P. Niebert, Else: a new symbolic state generator for timed automata, in: Proc. of the First International Conference FORMATS, Lecture Notes in Computer Science, Vol. 2791, Springer, Berlin, 2003, pp. 273–280.