LIF

Laboratoire d'Informatique Fondamentale
de Marseille

Unité Mixte de Recherche 6166
CNRS – Université de Provence – Université de la Méditerranée

A Partial Order Semantics Approach to the Clock Explosion Problem of Timed Automata

D. Lugiez, P. Niebert, S. Zennou

Rapport/Report 16-2003

December 19, 2003

Les rapports du laboratoire sont téléchargeables à l'adresse suivante
Reports are downloadable at the following address

http://www.lif.univ-mrs.fr

A Partial Order Semantics Approach to the Clock Explosion Problem of Timed Automata

D. Lugiez, P. Niebert, S. Zennou

LIF – Laboratoire d'Informatique Fondamentale de Marseille
UMR 6166
CNRS – Université de Provence – Université de la Méditerranée

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

[lugiez,niebert,zennou]@cmi.univ-mrs.fr

Abstract/Résumé

We propose a new approach for the symbolic exploration of timed automata that solves a particular aspect of the combinatory explosion occurring in the widely used clock zone automata (e.g. Kronos, UppAal). The latter approach suffers from splitting of symbolic states depending on the order of transition occurrences, even if these transitions concern unrelated components in a parallel system. Our goal is to preserve independence (commutation of transitions) from the original timed automaton to the symbolic level. We achieve this goal in three steps:

(1) we lift the theory of Mazurkiewicz traces to timed words and symbolic state exploration, generalising previous work;
(2) we propose a language theoretic setting for the study of the problem of symbolic state exploration, explaining difficulties of previous approaches to *partial order reductions* of timed automata and providing a roadmap for
(3) new data structures and an algorithm for symbolic reachability in timed automata.

It has to be underlined that our algorithm solves the same problem as the classical clock zone algorithms, but in a different manner and that we preserve the worst case estimations of the classical algorithms without restricting the systems considered. We have implemented our algorithm in a new state explorer and demonstrate potential savings resulting from our approach.

Keywords: verification, timed automata, partial order.

Nous proposons une nouvelle approche pour les méthodes symboliques d'exploration des états accessibles pour les automates temporisés qui résoud un problème d'explosion combinatoire apparaissant avec les automates de zones qui sont largement utilisés dans les outils (c.f. Kronos, UppAal). Cette dernière approche souffre de la décomposition des états symboliques liée à l'ordre des occurrences de transition, même si celles-ci

concernent des composants indépendants d'un système parallèle. Notre objectif est de préserver l'indépendance (i.e. la commutation des transitions) des automates temporisés au niveau symbolique, ce que nous obtenons en trois étapes:

(1) nous adaptons la théorie des traces de Mazurkiewicz aux mots temporisés et à l'exploration des états symboliques, généralisant ainsi des travaux précédents,
(2) nous proposons un cadre *langage formel* pour l'étude du problème de l'exploration symbolique, qui permet d'expliquer les difficultés des approches précédentes utilisant les réductions d'ordre partiel, ce qui fournit aussi un cadre pour
(3) de nouvelles structures de données et un algorithme pour décider l'accessibilité symbolique pour les automates temporisés.

Il faut insister sur le fait que notre algorithme résoud le même problème que les automates de zones classiques mais avec une approche différente et que nous avons la même estimation de coût que pour les algorithmes classiques, sans faire d'hypothèse restrictive sur les systèmes utilisés. Notre algorithme de recherche d'états accessibles a été implémenté et les premières expérimentations montrent que l'approche est prometteuse.

Mots-clés : verification, automates temporisés, ordres partiels.

Relecteurs/Reviewers: Remi Morin.

Notes: This work was supported by the IST project AMETIST (Advanced Methods in Timed Systems, contract IST-2001-35304, `http://ametist.cs.utwente.nl`).

1 Introduction

Timed automata [AD94] 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 and tools [NSY92,LPY95] for timed automata represent a considerable achievement (and indeed impressive industrial applications have been treated), the combinatory explosion particular to this kind of modelling and analysis – sometimes referred to as "clock explosions" (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 been invested into the optimisation of representations of clock constraints, see e.g. [DY96,BLP+99]. Another line of research is devoted to the overall reduction of reachability to logic constraint solving, e.g. [NMA+02].

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 [Val89,Pel93,God96,NHZL01], 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 losely 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" transitions a resetting clock $X := 0$, and b resetting clock $Y := 0$. Executing a first and then b means that afterwards (time may have elapsed) $X \geq Y$ whereas executing b first and then a implies that afterwards $X \leq Y$. 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 states.

Previous works trying to transfer partial order methods to the timed automata setting [YS97,DGKK98,BJLY98,Min99] have considered this problem as an obstacle to overcome in order to be able to apply the reductions known from discrete systems. E.g. [BJLY98,Min99] reestablish independence of the above transitions a and b by introducing the notion of *local time semantics*. The idea is that each component in a network has its own independent time progress, only when synchronisation occurs between two components, time is synchronised. The price is that clock differences in that model arbitrarily diverge, and that in general, this reestablished commutation leads to an unavoidably *infinite*

state space (where the aim was reduction!), see Proposition 8 for more details. [BJLY98,Min99] answer by restricting the class of automata in order to allow finite bounds on the state space. However, practically almost always the resulting state spaces are considerably bigger than with the classical approach and the benefit of partial order reduction is anihilated by this explosion.

The present work takes a completely new viewpoint on the problem of non-commutation of symbolic transitions: First of all, we clean up the theory of **timed Mazurkiewicz traces**. Where a path in a timed automaton must satisfy timing constraints, we relax a crucial assumption that transitions occur in in the same order sequentially and temporally: We restrict this requirement to dependent transitions. Our formalisation generalises "local time semantics" and also the partial formalisation given in [DT98]. We believe that this formalisation is a valuable contribution as such. To us, it was the necessary precondition for what follows:

The second important step is a **language theoretic view** on the verification problem of timed automata. Rather than considering immediately the problem of "symbolic states", typically representing sets of clock values, we look at the problem of possible paths through the timed automaton and the implied Nerode right congruence (as well as a correspondingpreorder notion), which is known to be equivalent to minimal automata in classical language theory. Our understanding is that all previous automata based approaches to the reachability problem in timed automata is related to this Nerode congruence, and attempts to avoid incomparable states (by better abstractions, etc.) aim to get closer to the actual Nerode congruence.

For the framework with commutation, the Nerode congruence is typically of infinite index (see Proposition 8), whereas the classical interleaving approaches prove its finiteness for the interleaving case. So is it better to avoid commutation?

In the third part of our contribution, **the semantical basis of a new search algorithm**, we manage to get "the best of both worlds": We compute symbolic states with respect to the infinite index Nerode congruence for the trace semantics (but avoiding state splitting for independent transitions), but we compare states with the finite index preorder (to cut branches in the search tree), **"catchup preorder"**, which is a right congruence for the classical interleaving semantics but obviously not for the relaxed semantics. It is closely related to zone inclusion in the classical setting and preserves paths in the interleaving semantics. We thus preserve the worst case bounds from classical clock zone algorithms and a good number of heuristic improvements that have been applied to improve those bounds carry over to our setting.

The miracle, that this approach is actually correct (i.e. yields exactly the same set of reachable control locations of the timed automaton as the standard semantics and as clock zone automata for that matter) relies on our timed Mazurkiewicz theory, which gives us for each timed word with relaxed constraints on the temporal order an equivalent path that does respect the stronger interleaving constraints.

Moreover, it is interesting to observe that the equivalence class of a timed trace with respect to catchup preorder corresponds to the "convex hull" of the clock zones reached by its representatives in the classical semantics. This observation gives an idea of the relation of the state spaces explored by our method and clock zone automata.

We have build our algorithm into a prototype tool, which allows both classical and partial order semantics for a fair comparison (we have not yet included all heuristic improvements published in the literature). In **no** example considered, our semantics gave more states than the classical semantics (although this seems possible for very pathological cases). On examples which are tightly coupled like Fischer's protocol with very little independence, a little reduction can be observed. On more loosely coupled systems such as the dining philosophers, the reduction turns out to be impressing. We also give an artificial series of examples where our method exposes polynomial growth and the classical approach is exponential. Concluding, the first tests show that our approach has a considerable potential. And this without partial order reduction!

In the future, we intend to add urgency to the framework and we want to investigate the question whether some limited form of partial order reduction is compatible with our framework.

The paper is structured as follows: In Section 2, we introduce the formal framework of clocked and timed words and the standard semantics of timed automata. In Section 3, we introduce the theory of Clocked and Timed Mazurkiewicz traces. In Section 4, we set up a plan of the subsequent construction in language theory terms and define equivalence relations of interest. In Section 5, we develop event zones as representation of the right congruence for realisable traces. In Section 6, we define the finite index catchup preorder and combine it with the event zone automaton of Section 5 for our reachability algorithm. In Section 7, we give some experimental results, which demonstrate the potential impact of our approach.

2 Basics

In this section, we introduce basic notions of words, languages, automata, as well as their timed counterparts.

Words and Automata. Given an alphabet Σ of actions denoted by $a, b, c \ldots$, Σ^* denotes the set of words on Σ with ϵ the empty word. Words are denoted by $u, v, w \ldots$ and a non-empty word is some finite sequence $a_1 \ldots a_n$. The length of a word w is denoted by $|w|$. As usual a Σ-automaton $\mathcal{A}$ is a quadruple $(S, s_0, \rightarrow, F)$ where S is a set of states, $s_0 \in S$ is the initial state, $F \subseteq S$ is the set of final states and $\rightarrow \subseteq Q \times \Sigma \times Q$ is a set of transitions. The set $L(\mathcal{A})$ is the set of words accepted by $\mathcal{A}$. The automaton is *deterministic* if for each state s and action a there is a at most one $s' \in S$ such that $s \xrightarrow{a} s'$.

Timed words. In real time systems, we associate to each position i of a sequence of actions $w = a_1 \dots a_n$ a time stamp which indicates when the corresponding action takes place. More precisely, a *timed word* is a pair (w, t) with $w \in \Sigma^*$ and t is a function assigning to each position of w an element of $\mathbb{R}^+$, the set of non-negative reals. For convenience, we set $t(0) = 0$ to be an additional time stamp for the beginning. In the literature, timed words are often represented as $(a_1, t_1), (a_2, t_2) \dots$ i.e. $t(i)$ is replaced by t_i. A timed word is *normal* if $t(i) \leq t(j)$ for $i \leq j$ like in $(a, 3.2)(c, 4.5)(b, 6.3)$ but not in $(a, 3.2)(c, 2.5)(b, 6.3)$. Normal timed words represent temporally ordered sequences of events and serve as standard semantics of timed automata in the literature [AD94]. Concatenation of normal words is only a partial function and the set of normal words is thus a *partial monoid* only.

Clocked words. In a timed system, events can occur only if some time constraints are satisfied. In timed automata, *clocks* belong to some finite set $\mathcal{C}$ and are used to express the time constraints between an event that resets a clock and another event that refers to the clock. This leads to the introduction of *clocked labels* which are triples (action, constraints on clocks, set of reset clocks). The constraints permitted here[1] associate to each clock an interval (min, max) which gives the set of possible values for the clock. The interval can be left-open, right-open, left-closed, right-closed and the bounds can be finte or infinite $-\infty, +\infty$. The interval $]-\infty, +\infty[$ means that no constraint exists and such constraints will not be written explicitly. To preserve decidability, all finite bounds are assumed to be integers (or syntactically more general: rational numbers). We are interested in finite subsets Δ of the infinite set of clocked labels, called clocked alphabets. A *clocked word* over Δ, usually denoted by ω, is simply a word in Δ^*.

Normal realisations of clocked words. In a clocked word $\omega = (a_1, c_1, r_1)$ $(a_2, c_2, r_2) \cdots (a_n, c_n, r_n)$ let $last_C(\omega)$ denote the last position m where the clock C is reset, i.e. s.t. $C \in r_m$ (for $(1 \leq m \leq n)$). By definition we set $last_C(\omega) = 0$ if no such position exists (therefore we assume that all clocks are reset at time $t_0 = 0$).

Definition 1. A timed word (w, t) is a *normal realisation* of a clocked word $\omega = \alpha_1 \dots \alpha_m$ with $\alpha_i = (a_i, c_i, r_i)$, iff

(i) they have the same length ($|w| = m$) and the same action sequence $w(i) = a_i$ for $i \in \{1, \dots, m\}$,
(ii) (w, t) is normal,
(iii) and for all prefixes $\alpha_1 \dots \alpha_{k-1}$ and all clocks C with $l = last_C(\alpha_1 \dots \alpha_{k-1})$, $t(k) - t(l) \in c_k(C)$, i.e. the time elapsed since the last reset of clock C before position k meets the interval constraint at position k.

[1] In some works, differences of clocks are also permitted, but this is problematic, as pointed out by [Bou02].

For instance the timed word $w = (a, 3.2)(c, 4)(b, 6.2)$ is a normal realisation of $\omega = \alpha\gamma\beta$ (as defined in Figure 1). A clocked word is *realisable* iff it has a normal realisation. We say that α is a *realisable extension* of ω if $\omega\alpha$ is realisable. The set of realisable clocked words over some clocked alphabet is closed under the prefix relation.

A *timed automaton* is a Δ-automaton for some clocked alphabet Δ, as in Figure 1 (where all states are final). The language of a timed automaton $\mathcal{A}$ is denoted by $L(\mathcal{A})$, and the *timed language* $L_T(\mathcal{A})$ of $\mathcal{A}$ is the set $\{(w, t) \mid (w, t)$ is a normal realisation of some $\omega \in L(\mathcal{A})\}$. On the level of clocked words, let L_N be the language of realisable clocked words accepted by $\mathcal{A}$. For instance $(a, 3.2)(c, 4)(b, 6.2) \in L_T(\mathcal{A})$ is a normal realisation of $\alpha\gamma\beta \in L_N(\mathcal{A})$.

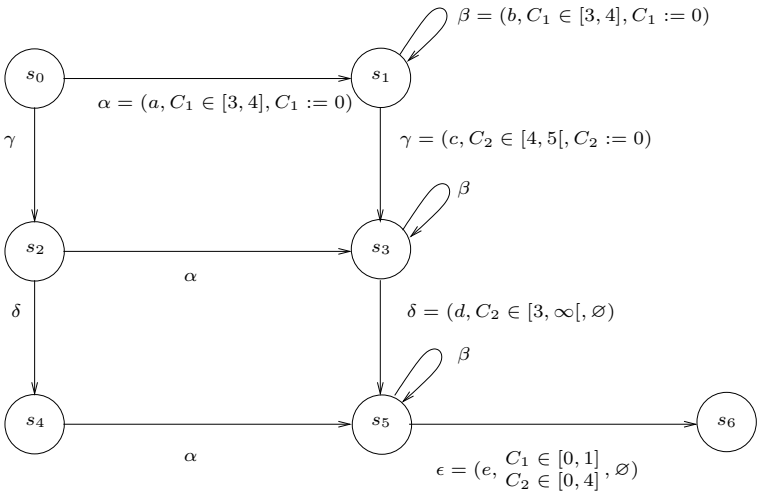


Fig. 1. A timed automaton

3 Clocked and Timed Mazurkiewicz Traces

As a representation of concurrency, we introduce an independence relation and generalise the theory of Mazurkiewicz traces to the timed setting. We first recall the basics of Mazurkiewicz trace theory in the untimed case. For an exhaustive treatment see [DR95].

Independence Relation and Traces. For an alphabet Σ, an *independence relation* is a symmetric and irreflexive relation $I_\Sigma \subseteq \Sigma \times \Sigma$. We call (Σ, I_Σ) a *partially commutative alphabet*. For convenience, we also use the dependence relation $D_\Sigma = \Sigma \times \Sigma - I_\Sigma$, which is reflexive and symmetric. As a representation of parallel systems we assume without loss of generality that $\Sigma = \bigcup_{i=1}^{l} \Sigma_i$

where $a\ D_\Sigma\ b$ iff $a, b \in \Sigma_i$ for some $i \in \{1, \ldots, l\}$. We call $(\Sigma_1, \ldots, \Sigma_l)$ a *distributed alphabet* of (Σ, I_Σ) and Σ_i a *component*. For convenience, we call the set $\{1, \ldots, l\}$ "*Comp*" (for components). For instance $(\Sigma_1 = \{a, b, e\}, \Sigma_2 = \{c, d, e\})$ is a distributed alphabet corresponding to $\Sigma = \{a, b, c, d, e\}$ and an independence relation $I_\Sigma = \{(a, c), (c, a), (b, c), (c, b), (a, d), (d, a), (b, d), (d, b)\}$. It is well known that every partially commutative alphabet corresponds to a distributed alphabet and conversely. Intuitively, I_Σ represents concurrency between actions, whereas the distributed alphabet proposes as explanation of concurrency occurrence on distinct processes in a distributed system. In order to reference actions or locations depending on an action we define $dep(a) = \{b \in \Sigma \mid a\ D_\Sigma\ b\}$ and $loc(a) = \{i \mid a \in \Sigma_i\}$. It is obvious that $dep(a) = \bigcup_{i \in loc(a)} \Sigma_i$. In analogy to last occurrences of clock resets, we define $last_i(a_1 \ldots a_n)$, the *last occurrence* of an action of the component Σ_i, as the maximal k such that $a_k \in \Sigma_i$, if such a k exists, otherwise $last_i(a_1 \ldots a_n) = 0$. For instance, $last_1(acb) = 3$ and $last_2(acb) = 2$ for $(\Sigma_1 = \{a, b, e\}, \Sigma_2 = \{c, d, e\})$.

The *Mazurkiewicz trace equivalence* associated to the partially commutative alphabet (Σ, I_Σ) is the least congruence $\simeq_M$ over $\Sigma^\star$ such that $ab \simeq_M ba$ for any pair of independent actions $a\ I_\Sigma\ b$. A *trace* $[u]$ is the congruence class of a word $u \in \Sigma^\star$. We denote by $\mathbb{M}(\Sigma, I_\Sigma)$ the set of all traces w.r.t. (Σ, I_Σ).

Before adapting these notions to the timed setting, we give the connection between independence relations and automata as a condition on transition relations:

Definition 2 (asynchronous automaton). An *asynchronous automaton* over (Σ, I_Σ) is a deterministic Σ-automaton such that the following two properties hold for any two letters $a, b \in \Sigma$ with $a\ I_\Sigma\ b$:

ID: $s \xrightarrow{a} s_1 \xrightarrow{b} s_2$ implies $s \xrightarrow{b} s_1' \xrightarrow{a} s_2$ for some state s_1' *[Independent Diamond]*
FD: $s \xrightarrow{a} s_1$ and $s \xrightarrow{b} s_1'$ implies $s_1 \xrightarrow{b} s_2$ for some state s_2 *[Forward Diamond]*

It is important to note that the languages of asynchronous automata are closed with respect to equivalent words. The theoretical foundation of many partial order reduction approaches is based on this fact. For instance, reductions that preserve at least one representative for each equivalence class do preserve non-emptiness of the language of an automaton.

Intuitively, two words are equivalent with respect to $\simeq_M$ iff they can be obtained from each other by exchanging adjacent independent letters. In other words, this permutation of letters between two equivalent words lets the relative order of dependent letters unchanged. This property is formally stated in the following lemma.

Lemma 3. *Let (Σ, I_Σ) be a partially commutative alphabet and $a_1 \ldots a_n \simeq_M b_1 \ldots b_n$ be two equivalent words. There exists a uniquely determined permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, such that $a_i = b_{\pi(i)}$ and for $a_i\ D_\Sigma\ a_j$ we have $i < j$ iff $\pi(i) < \pi(j)$.*

Conversely, let $a_1 \dots a_n$ *be a word and* $\pi : \{1, \dots, n\} \to \{1, \dots, n\}$ *be a permutation of indices such that for each pair* i, j $a_i \; D_\Sigma \; a_j$ *we have* $i < j$ *iff* $\pi(i) < \pi(j)$*. Then* $a_{\pi(1)} \dots a_{\pi(n)} \simeq_M a_1 \dots a_n$*.*

For convenience, we assume π *to be defined on* 0 *with* $\pi(0) = 0$*.*

Proof. First direction. By definition, $\simeq_M$ is the reflexive, symmetric, transitive closure of the binary relation $\sim$ defined by: $uabv \sim ubav$ for $u, v \in \Sigma^*$ iff $a \; I_\Sigma \; b$. Hence, $a_1 \dots a_n \simeq_M b_1 \dots b_n$ iff there exists a word sequence $w_0, \dots w_k$ such that $w_0 = a_1 \dots a_n$, $w_k = b_1 \dots b_n$ and $w_{i-1} \sim w_i$ for all $i \in 1, \dots, k$.

Let m_i be the position in w_{i-1}, w_i such that w_{i-1} and w_i differ only from the exchange of $w_i(m_i)$ and $w_i(m_i + 1)$. Moreover, let $\pi_i : \{1, \dots, n\} \to \{1, \dots, n\}$ be the permutation of indices from w_{i-1} to w_i. Hence, π_i is defined by: For all $l \in \{1, \dots, m_i - 1\} \cup \{m_i + 2, \dots, n\}$ $\pi_i(l) = l$ and $\pi_i(m_i) = m_i + 1$ and $\pi_i(m_i + 1) = m_i$. The required permutation π is the composition of these π_i. It remains to show that π is unique, and that each π_i satisfies the required property and that this property is closed under composition:

> *π is unique.* Suppose there exists $\pi' \neq \pi$ permutation of indices which satisfies the required property. There must exists at least two occurrences k_1, k_2 of an action a, i.e. $a_1 = a_2$, such that $\pi(k_1) < \pi(k_2)$ and $\pi'(k_1) > \pi'(k_2)$ or the reverse. As $\simeq_M$ is reflexive, $a_{k_1} \; D_\Sigma \; a_{k_2}$ and hence because of the permutaion property, $k_1 < k_2$ iff $\pi(k_1) < \pi(k_2)$ and $\pi'(k_1) < \pi'(k_2)$, which contradicts the hypothesis.
> *π_i satisfies the property.* Let $p \neq q$ be indices such that $w_{i-1}(p) D_\Sigma w_{i-1}(q)$. By definition of $\sim$, either $m_i \neq p$ or $m_i \neq q$. So $p < q$ iff $\pi_i(p) < \pi_i(q)$ as:
> If $p \neq m_i$ and $q \neq m_{i+1}$ then $\pi_i(p) = p$ and $\pi_i(q) = q$. Obviously, $p < q$ iff $\pi_i(p) < \pi_i(q)$.
> If $p = m_i$ and $q \neq m_{i+1}$ then $\pi_i(p) = m_{i+1}$ and $\pi_i(q) = q$. If $p < q$ then $q > m_{i+1}$ and hence $\pi_i(p) < \pi_i(q)$. If $\pi_i(p) < \pi_i(q)$ then $\pi_i(q) > m_{i+1}$, i.e. $q > m_{i+1}$ which implies that $p = m_i < m_{i+1} < q$.
> If $p \neq m_i$ and $q = m_{i+1}$ then $\pi_i(p) = p$ and $\pi_i(q) = m_i$. If $p < q$ then $p < m_i$ and hence $\pi_i(p) < \pi_i(q)$. If $\pi_i(p) < \pi_i(q)$ then $\pi_i(p) < m_i$, i.e. $p < m_i$ which implies that $p < m_i < m_{i+1} = q$.
> *the property is closed under composition.* Let $a_1 \dots a_n \simeq_M b_1 \dots b_n$, $b_1 \dots b_n \simeq_M c_1 \dots c_n$ and π, π' their respective permutations with the desired property. Let $a_i D_\Sigma a_j$. From the construction of π, we deduce $b_{\pi(i)} D_\Sigma b_{\pi(j)}$ and hence by hypothesis on π', we get $\pi(i) < \pi(j)$ iff $\pi'(\pi(i)) < \pi'(\pi(j))$. By hypothesis on π, we have $i < j$ iff $\pi(i) < \pi(j)$. Finally, from the two equivalences, we obtain the desired result: For $a_i D_\Sigma a_j$, we have $i < j$ iff $\pi'(\pi(i)) < \pi'(\pi(j))$.

The proof of the second property is by induction on the number of inversions, that is pairs of positions $i < j$ such that $\pi(i) > \pi(j)$.

The basic case – no inversion – is immediate: π is the identity and $a_1 \dots a_n \simeq_M a_{\pi(1)} \dots a_{\pi(n)}$ as $\simeq_M$ is reflexive.

For the inductive case, suppose that π is not the identity. Hence, there must exist at least one index k such that $\pi(k) > \pi(k + 1)$ which implies $a_k \; I_\Sigma \; a_{k+1}$ (by construction of π.)

Let π_1 be the index permutation which permutes the indices k and $k+1$ (and let the other ones unchanged). By definition of $\simeq_M$, we have $a_1 \dots a_k a_{k+1} \dots a_n \simeq_M a_1 \dots a_{k+1} a_k \dots a_n = a_{\pi_1(1)} \dots a_{\pi_1(n)}$ as $a_k \; I_\Sigma \; a_{k+1}$.

Let π_2 be the index permutation such that $\pi = \pi_2 \circ \pi_1$. Note that if π satisfies for $a_i \; D_\Sigma \; a_j \; i < j$ iff $\pi(i) < \pi(j)$ on the word $a_1 \dots a_n$ then so π_2 does on the word $a_{\pi_1(1)} \dots a_{\pi_1(n)}$. Moreover the number of inversions in the permutation π_2 is equal to the inversion number in π minus one (for this one of π_1).

Applying the induction hypothesis on π_2 (because of the two above arguments), we get $a_{\pi_1(1)} \dots a_{\pi_1(n)} \simeq_M a_{\pi_2(\pi_1(1))} \dots a_{\pi_2(\pi_1(n))}$. Finally, it comes from the transitivity of $\simeq_M$ that $a_1 \dots a_n \simeq_M a_{\pi_2(\pi_1(1))} \dots a_{\pi_2(\pi_1(n))} = a_{\pi(1)} \dots a_{\pi(n)}$. □

Generalisation to Clocked Words.

Timed traces. The independence relation I_Σ immediately carries over to (non normal) timed words. The resulting congruence classes are called *timed traces*. Here, the exchange of two independent actions also exchanges their time stamps, e.g. $(a, 3.2)(b, 3.5)(c, 6.3) \simeq_M (a, 3.2)(c, 6.3)(b, 3.5)$ where $b \; I_\Sigma \; c$, which means that normality (temporal order of actions) is not preserved under commutation. Therefore we introduce a weaker notion of normality: a timed word (w, t) is I_Σ-*normal* iff for any two letters $a = w(i), b = w(j)$ with $i \leq j$ **and additionally** $a \; D_\Sigma \; b$ we have $t(i) \leq t(j)$. This relaxed normality condition is preserved under Mazurkiewicz equivalence, allowing to define normality on the level of traces: We call a timed trace $[(w, t)]$ I_Σ-*normal* iff (w, t) is I_Σ-normal.

Proposition 4. *Every I_Σ-normal timed word (w, t) is equivalent to a normal timed word (w', t').*

Proof. Let us write (w, t) as a sequence of pairs $(\alpha_1, t(1)), \dots, (\alpha_n, t(n))$ where $\alpha_i = (a_i, c_i, r_i)$. We define a total ordering on these pairs as follows:
$(\alpha_i, t(i)) < (\alpha_j, t(j))$ iff

- either $i < j$ and there is some k such that $i \leq k \leq j$ and $a_i \; D_\Sigma \; a_k$ and $a_k \; D_\Sigma \; a_j$,
- or there is no such k but $t(i) < t(j)$,
- or none of the previous cases occurs but $i < j$.

The last condition yields totality by using the index to order pairs which are uncomparable with respect to dependent actions or time stamps. A straightforward sorting algorithm is obtained by repeatedly using the exchange rule $\dots (\alpha_i, t(i))(\alpha_j, t(j)) \dots \to \dots (\alpha_j, t(j))(\alpha_i, t(i)) \dots$ if $(\alpha_i, t(i)) > (\alpha_j, t(j))$. Since (w, t) is I_Σ-normal and the exchange preserves this property, the process ends with a sorted word (w', t') which is I_Σ-normal. By definition of $<$, this yields that (w', t') is normal.

Using a more efficient algorithm, we can compute (w', t') in $O(|w| log(|w|))$ if we already know that (w, t) is I_Σ-normal, otherwise we use a bubble-like sorting algorithm which costs quadratic time. □

Independence for clocked words. To extend the independence relation I_Σ to clocked words, we define $I_\Delta \subseteq \Delta \times \Delta$ based on I_Σ as follows: $(a_1, c_1, r_1)\ I_\Delta\ (a_2, c_2, r_2)$ iff (i) $a\ I_\Sigma\ b$, (ii) $r_1 \cap r_2 = \emptyset$ and (iii) For all $C \in r_1$ we have $c_2(C) =]-\infty, \infty[$ and conversely for all $C \in r_2$ we have $c_1(C) =]-\infty, \infty[$.

Intuitively, conditions (ii) and (iii) arise from the view of clocks as shared variables in concurrent programming: An action resetting a clock is writing it whereas an action with a non-trivial condition on a 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.

We call the (Δ, I_Δ) constructed in this way a *partially commutative clocked alphabet* and say that I_Δ *respects* I_Σ. The notion of traces and equivalence $\simeq_M$ are defined as for I_Σ.

For the rest of the paper, we will silently assume some partially commutative clocked alphabet (Δ, I_Δ). If clear from the context, we write I instead of I_Δ.

Relaxing the notion of normal realisations, the following definition establishes the relation between clocked words and I-normal timed words.

Definition 5 (I_Δ-normal realisation). Let $\omega = \alpha_1 \dots \alpha_n$ with $\alpha_j = (a_j, c_j, r_j)$ be a clocked word over a partially commutative clocked alphabet (Δ, I_Δ). A timed word (w, t) *I_Δ-realises* ω iff

(i) (same length) $|\omega| = |w|$, (same actions) for $j = 1, \dots, |w|$ we have $w(j) = a_j$,
(ii) (normality) (w, t) is I_Σ-normal,
(iii) (satisfaction of constraints) for all prefixes $\alpha_1 \dots \alpha_{k-1}$ and all clocks C with $l = last_C(\alpha_1 \dots \alpha_{k-1})$ $t(k) - t(l) \in c_k(C)$. In that case, we also say that ω is *I_Δ-realisable* and by extension that $[(w, t)]$ is a I_Δ-realisation of ω.

For instance, the timed word $(c, 4)(a, 3.2)(b, 6.2)$ I_Δ-realises the clocked word $\gamma\alpha\beta$ (for the automaton in Figure 1, assuming $\alpha\ I_\Delta\ \gamma$).

The main result of this section establishes the tight link between clocked and timed traces, in particular it shows that I-realisability is invariant under trace equivalence, allowing in principle the exploration of realisable clocked words on representatives.

Theorem 6. *Let $\alpha_1 \dots \alpha_n \simeq_M \beta_1 \dots \beta_n$ be two equivalent clocked words over (Δ, I_Δ) and π be the permutation as defined in Lemma 3. Then $(b_1, t_1) \dots (b_n, t_n)$ is an I-normal realisation of $\beta_1 \dots \beta_n$ iff $(b_{\pi(1)}, t_{\pi(1)}) \dots (b_{\pi(n)}, t_{\pi(n)})$ is an I-normal realisation of $\alpha_1 \dots \alpha_n$.*

Proof. First implication.

Let $(b_1, t_1) \dots (b_n, t_n)$ be a I-normal realisation of $\beta_1 \dots \beta_n$ with $\beta_j = (b_j, c_j, r_j)$ for $j \in \{1, \dots, n\}$.

$(b_{\pi(1)}, t_{\pi(1)}) \dots (b_{\pi(n)}, t_{\pi(n)})$ is a I-normal realisation of $\alpha_1 \dots \alpha_n$ as the three conditions in the definition 5 are satisfied:

(i) Obviously, $|b_{\pi(1)} \dots b_{\pi(n)}| = |\alpha_1 \dots \alpha_n|$. Moreover, by definition $\alpha_j = \beta_{\pi(j)}$ is equal to $(b_{\pi(j)}, c_{\pi(j)}, r_{\pi(j)})$.

(ii) Let $\pi(i) < \pi(j)$ be indices such that $b_{\pi(i)}\ D_\Sigma\ b_{\pi(j)}$. Hence, $t_{\pi(i)} \leq t_{\pi(j)}$ as $(b_1, t_1) \dots (b_n, t_n)$ I_Σ-realises $\beta_1 \dots \beta_n$ (condition ii in Definition 5).

(iii) Let $\alpha_{\pi(1)} \dots \alpha_{\pi(k-1)}$ be a prefix and C be a clock with $\pi(l) = last_C(\alpha_{\pi(1)} \dots \alpha_{\pi(k-1)})$. The proof that $t_{\pi(k)} - t_{\pi(l)} \in c_{\pi(k)}$ depends on the value of $c_{\pi(k)}$. If $c_{\pi(k)}$ is $]-\infty, +\infty[$, then the constraint is trivially satisfied. In the other case, by definition of I_Δ (condition ii), as $\alpha_{\pi(k)}$ and $\alpha_{\pi(l)}$ are two labels which respectively refers to a clock C and resets it, their actions have to be dependent, i.e. $a_{\pi(k)}\ D_\Sigma\ a_{\pi(l)}$. Hence and as $\pi(k) > \pi(l)$, by Lemma 3, we deduce that $l < k$, i.e. l is an index in $b_1 \dots b_{k-1}$. Also, by Lemma 3, we conclude that l is the greatest index in $b_1 \dots b_{k-1}$ of a label which resets C, i.e. $l = last_C(b_1 \dots b_{k-1})$: Otherwise, assume that there exists $i < k$ such that $i = last_C(b_1 \dots b_{k-1})$ and $i > l$. As α_i and α_l reset the same clock, their actions a_i and a_l are dependent which implies by Lemma 3 that $\pi(i) > \pi(l)$ and contradicts that $\pi(l) = last_C(\alpha_{\pi(1)} \dots \alpha_{\pi(k-1)})$. Finally, $t_{\pi(k)} - t_{\pi(l)} \in c_{\pi(k)}$ as $\pi(l) = last_C(\alpha_{\pi(1)} \dots \alpha_{\pi(k-1)})$ and $(b_1, t_1) \dots (b_n, t_n)$ I-realises $\beta_1 \dots \beta_n$ (condition iii in Definition 5)

Second implication. Symmetric to the first one: Exchange $\alpha_1 \dots \alpha_n$ and $\beta_1 \dots \beta_n$ together with their I-realisations. The indices like i and $\pi(i)$ are also exchanged. □

Applications to the verification problem. In analogy to the definition of $L_N(\mathcal{A})$ let $L_I(\mathcal{A})$ denote the set of I-realisable clocked words accepted by $\mathcal{A}$. It is straight forward by definition that $L_T(\mathcal{A}) = \emptyset$ iff $L_N(\mathcal{A}) = \emptyset$ iff $L_I(\mathcal{A}) = \emptyset$, so that we can check this emptiness problem equivalently for either language. The important aspect of $L_I(\mathcal{A})$ is that it is closed under equivalence as expressed in the following corollary of Theorem 6:

Corollary 7. *Let $\omega \simeq_M \omega'$ be equivalent clocked words, then ω is I-realisable iff ω' is I-realisable and $\omega \in L_I(\mathcal{A})$ iff $\omega' \in L_I(\mathcal{A})$.*

If $\omega \in L_I(\mathcal{A})$ then there exists $\omega' \simeq_M \omega$ such that $\omega' \in L_N(\mathcal{A})$.

Proof. For the first claim, suppose ω to be I-realisable and apply Theorem 6 on realising I-normal timed word to find a witness of the I-realisability of ω'.

For the second part, assume an I-normal timed word (w, t) realising ω and its normal counterpart (w', t') according to Proposition 4. Then the permutation of indices linking (w, t) and (w', t') applied to ω yields the realisable ω' as desired. □

This observation gives rise to the hope that partial order reduction techniques could be applied when checking for emptiness of $L_I(\mathcal{A})$. However, as explained in the following sections, this language cannot always be represented by a finite automaton and more sophisticated methods are needed to actually solve this emptiness problem.

4 A language theoretic view

Our primary goal is to build a finite automaton for the language $L_I(\mathcal{A}) = \{\omega \mid \omega \ I\text{-realisable and } \omega \in L(\mathcal{A})\}$, which yields an immediate way to decide the emptiness of the language. For any language, the classical way to build an automaton is to consider the Myhill-Nerode right-congruence which yields the minimal automaton accepting the language (the states are the equivalence classes of the congruence)[2]. In our case the relevant congruence would be [3] $\omega_1 \simeq_I \omega_2$ `iff` $\omega_1 \lesssim_I \omega_2$ `and` $\omega_2 \lesssim_I \omega_1$ where $\omega_1 \lesssim_I \omega_2$ iff $\forall \omega, \omega_1\omega$ I-realisable implies $\omega_2\omega$ I-realisable. By definition $\simeq_M \subseteq \lesssim_I$, which justifies to write $[\omega_1] \lesssim_I [\omega_2]$. Unfortunately, this congruence is not of finite index:

Proposition 8. *There exist finite Δ for which $\lesssim_I$ and $\simeq_I$ are of infinite index.*

Proof. Let $\alpha = (a, X \in [1,1], X := 0)$, $\beta = (b, Y \in [1,1], Y := 0)$, $\gamma = (c, X \in [1,1], Y \in [1,1], Y := 0)$ with $\alpha I \beta$. Then for $i \neq j$ we have that $\alpha^i \not\lesssim_I \alpha^j$, because the extension $\omega = \beta^i\gamma$ makes $\alpha^i\beta^i\gamma$ I-realisable whereas $\alpha^j\beta^i\gamma$ is not I-realisable. □

This ruins our primary goal explains the problems with partial order reductions known from the literature. The solution chosen by [BJLY98,Min99] is to restrict the class of systems so that the languages remain finite state. But even under these somewhat severe restrictions, the index of $\simeq_I$ is often significantly bigger than what would be obtained $\simeq_N$, questioning the endeavor of partial order reductions for timed automata on the whole.

Our approach is to use an indirect and complex approach to decide $L_I(\mathcal{A}) \stackrel{?}{=} \emptyset$. Keeping the Myhill-Nerode congruence idea in mind, we define several relations which help understanding the problems and that provide constructions similar to zones for timed automata while preserving properties of realizable traces. Again the resulting automaton is infinite but we define a relation on zones which has a finite index and allows to decide the emptiness of $L_I(\mathcal{A})$ in a finite amount of time.

Given some language L, a *right-precongruence* is a relation $\lesssim_L$ such that $u \lesssim_L v$ iff $\forall w$, if $uw \in L$ implies $vw \in L$. The obvious link with $\simeq_L$ is $\simeq_L = (\lesssim_L \cap \gtrsim_L)$. The index of a preorder $\lesssim$ is by definition the index of the equivalence $\lesssim \cap \gtrsim$. We describe now all the relations that we use, apart $\lesssim_I$ and $\simeq_I$ that are already defined.

Definition 9 ($\lesssim_N$,$\simeq_N$, $\lesssim_{IN}$, $\simeq_{IN}$). For clocked words ω_1, ω_2, let

- $\omega_1 \lesssim_N \omega_2$ iff $\forall \omega$, if $\omega_1\omega$ has a normal realisation then $\omega_2\omega$ has a normal realisation. In general $\simeq_M \not\subseteq \lesssim_N$, so $\lesssim_N$ cannot be lifted to traces and is given for comparisons only.

[2] but this automaton is finite for regular languages only!

[3] for simplicity we forget momentarily the finite automaton $\mathcal{A}$

- $\omega_1 \lesssim_{IN} \omega_2$ iff $\forall \omega$ if there exists $\omega_1' \simeq_M \omega_1$ such that $\omega_1'\omega$ has a normal realisation, then there exists $\omega_2' \simeq_M \omega_2$ such that $\omega_2'\omega$ has a normal realisation. We define $\omega_1 \simeq_{IN} \omega_2$ by $\omega_1 \lesssim_{IN} \omega_2$ and $\omega_2 \lesssim_{IN} \omega_1$. This relation still concerns normal realization, but weakens $\lesssim_N$ by forgetting the interleaving of the past.
- $\lesssim_{EZ}$ (defined in section 5) is defined in terms of difference constraints sets generated by clock constraints and can be seen as an implementation of $\lesssim_I$ since $\lesssim_{EZ} \subseteq \lesssim_I$.
- $\lesssim_C$ is defined from $\lesssim_{EZ}$ and can be seen as an implementation of $\lesssim_{IN}$ since $\lesssim_C \subseteq \lesssim_{IN}$.

The relations $\lesssim_I, \lesssim_{EZ}$ are precongruences that are used to define automata, but they may have infinite index while $\lesssim_{IN}, \lesssim_C$ have finite index but may not be precongruences. Their properties are summarized in Figure 2

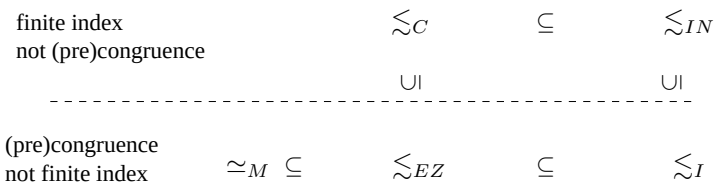


Fig. 2. Right preorders for clocked traces

The proof of Proposition 8 supports the claim that $\lesssim_{IN}$ is not a precongruence: $\alpha\alpha \simeq_{IN} \alpha\alpha\alpha$, but $\alpha\alpha\beta \not\simeq_{IN} \alpha\alpha\alpha\beta$. However, the relation $\lesssim_{IN}$ is a crucial tool for solving $L_I(\mathcal{A}) \stackrel{?}{=} \emptyset$ because (i) the inclusion $\lesssim_I \subseteq \lesssim_{IN}$ holds (see Proposition 22) provided some slight assumptions on the alphabet Δ, (ii) it is of finite index, (iii) it preserves the non-emptiness of $L_I(\mathcal{A})$ (in a weak sense). The relations $\lesssim_{EZ}, \lesssim_C$ represent the computational aspects of our approach and give an effective way to approximate the relations $\lesssim_I$ and $\lesssim_{IN}$.

A similar approach underlies the theory of timed automata: the language of the realisable clocked words $L_N(\mathcal{A})$ of an automaton $\mathcal{A}$ is represented by a *zone automaton* and the constructions given in the litterature can be understood as computating precongruences $\lesssim_{ZA} \subseteq \lesssim_{L_N(\mathcal{A})}$. These precongruences may have (many) more states than the ideal $\lesssim_{L_N(\mathcal{A})}$ and works for improving timed automata constructions can often be seen as tentatives to get closer to $\lesssim_{L_N(\mathcal{A})}$. But whatever the finite size of these zone automata, they prove that $\lesssim_{L_N(\mathcal{A})}$ is of finite index. The reader should notice that the bound that we get for the index of $\lesssim_C$ in Proposition 25 is remarquably close to the bound for the number of clock zones of classical timed automata.

5 Event zones for the representation of $\lesssim_I$

This section is devoted to the construction of $\lesssim_{EZ}$ and $\simeq_{EZ}$ with *event zones*. The aim is to obtain a right precongruence reasonably close to $\lesssim_I$ that allows efficient data structures and algorithms for the representation of congruence classes and for testing $\lesssim_{EZ}$.

Difference constraint sets provide the tool needed to achieve this goal, leading to the construction of an *event zone automaton*, which specifies the set of I-realisable traces. This automaton may still be infinite and section 6.3 will show how to decide emptiness of the accepted language.

Difference Constraint Sets

Difference constraint sets and their relation to the all pairs shortest path problem are well known, see e.g. [CLR90]. We briefly introduce them here for the convenience of the reader.

Difference constraint sets are set of inequations of the form $x - y \leq c$ or $x - y < c$ where x and y are real valued variables and c is a numerical constant (a rational number or an integer). A classical representation of such constraints is to use a graph where the vertices are labelled by the variables, and there is an edge from x_i and x_j labelled by $c, \leq$ (resp. $c, <$) iff $x_i - x_j \leq c$ (resp. $x_i - x_j < c$) is one of the constraints (when several constraints relate the same variables, we choose the stricter one). The graph is completed by adding the constraints $x - x \leq 0$ for every x and $x_i - x_j < +\infty$ when no constraints $x_i - x_j \leq c$ (or $x_i - x_j < c$) exist.

As an example constraint set, let us consider the clocked word $\alpha\beta$ of the timed automaton of Figure 1. Any I-normal realisation is some timed word $(a, x_1)(b, x_2)$ where the time stamps x_0 (the initial time stamp), x_1, x_2 must satisfy the constraints:

$$\begin{array}{lll} 3 \leq x_1 - x_0 \leq 4 & \textit{or equivalently} & x_0 - x_1 \leq -3, x_1 - x_0 \leq 4 \\ 3 \leq x_2 - x_1 \leq 4 & & x_1 - x_2 \leq -3, x_2 - x_1 \leq 4 \\ x_1 \leq x_2 & & x_1 - x_2 \leq 0 \end{array}$$

The representation of the previous set is given in Figure 3 (left part).

Formally, a *difference constraint set* S over a finite set of variables $V = \{x_1, \ldots, x_n\}$ is a set of inequalities $x_i - x_j \prec c$ where $\prec$ is $\leq$ or $<$, $c \in \mathbb{Z} \cup \{+\infty\}$. To define the associated constraint graph, we introduce the minimum operator on pairs (inequality sign, constant) as follows:

$$min\{(c_1, \prec_1), (c_2, \prec_2)\} = \begin{cases} (c_1, \prec_1) \; if \; c_1 < c_2 \; or \; c_1 = c_2 \; and \; \prec_1 = <, \\ (c_2, \prec_2) \; otherwise \end{cases}$$

For technical convenience, we introduce also operators $<, \leq, >, \geq$ over couples $(c, \prec)$ as usual derived from the min operator.

A difference constraint set S is represented by the directed complete labeled graph (V, E) such that an edge between x and y is labeled by $E(x, y) = min\{(c, \prec) \mid x - y \prec c \in S\}$ for $x \neq y$. If no inequality $x - y \prec c$ occurs in S, the value is

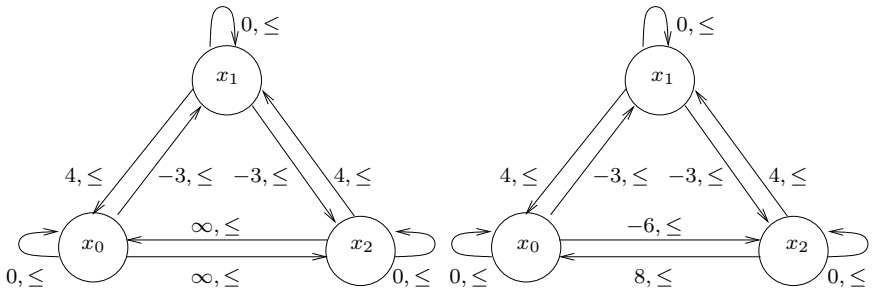


Fig. 3. A difference constraint set (left) and its closure (right).

$(+\infty, <)$, and $E(x, x) = (0, \leq)$. There is a one to one correspondence between difference constraint sets and graphs when there is a unique constraint $x - y \prec c$ for the ordered pair x, y (which is always the case for the constraints that we consider thereafter) and we shall identify a difference constraint graph and the corresponding set.

A *solution* of a difference constraint set is a valuation $v : V \rightarrow \mathbb{R}$ such that all inequations of S are satisfied, i.e. for $E(x, y) = (c, \prec)$ it must hold that $v(x_i) - v(x_j) \prec c$. A solution $v : V \rightarrow \mathbb{R}$ is said *positive* iff for all $x \in V$ $v(x) \geq 0$. A difference constraint set with at least one solution is *consistent*, otherwise it is *inconsistent*.

Proposition 10. *A difference constraint set is consistent iff it has a positive solution.*

As the time stamps in the timed words are positive, we only consider positive solutions in the sequel.

Operations on difference constraint sets For our applications, we recall two important operations on difference constraint sets: closure, and projection.

Closure of difference constraint sets. First, we define $\oplus$ on pairs $(c, \prec)$ by $(c_1, \prec_1) \oplus (c_2, \prec_2) = (c_1 + c_2, \prec_1)$ if $\prec_1 = \prec_2$ and $(c_1 + c_2, <)$ otherwise. With respect to solutions, we see that if $v(x) - v(y) \prec_1 c_1$ and $v(y) - v(z) \prec_2 c_2$ then this implies that also $v(x) - v(z) \prec_3 c_3$ where $(c_3, \prec_3) = (c_1, \prec_1) \oplus (c_2, \prec_2)$. The closure $Cl(V, E)$ of (V, E) combines all these implicit constraints: $Cl(V, E)$ is the difference constraint set (V, E') where $E'(x, y) = min\{E(x_1, x_2) \oplus \ldots \oplus E(x_{p-1}, x_p) \mid x = x_1, \ldots, x_p = y \in V\}$, i.e. the length of the shortest path from x to y if it exists, $-\infty, <$ otherwise. The closure of the previous difference constraint graph is given in Figure 3 (left part).

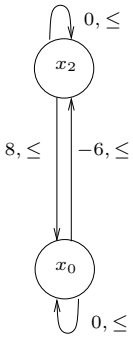
Proposition 11. *A difference constraint set is consistent iff its closure is consistent.*

Actually, there are only three possible cases concerning $Cl(E,V) = (E',V')$:

- $E'(x,x') = (+\infty, <)$ and there is no path of finite length in (E,V) joining x and x',
- $E'(x,x') = (-\infty, <)$ and there is a path of finite length in (E,V) joining x and x' which contains a cycle of negative length,
- $E'(x,x')$ is the length of a simple path (a variable may occur at most most) joining x and x' in (E,V).

The closure can be efficiently computed using an all pairs shortest path algorithm such as Floyd-Warshall [CLR90]. All pairs shortest path algorithms either return the shortest paths between all pairs if they exist, otherwise they detect a failure as a negative cycle in the constraint graph, i.e. a cycle with the sum of labels yielding $(c, \prec)$ with $c < 0$ or $c = 0$ and $\prec = <$, (the label of a path is the sum of all edge labels with respect to the operator $\oplus$.)

Projection of difference constraint sets. Given $V' \subseteq V$, the projection $\Pi_{V'}$ of (V,E) on V' is the difference constraint set (V',E') such that $E'(x,y) = E(x,y)$ for $x,y \in V'$. The figure at right gives the projection on $\{x_0, x_2\}$ of the closure of the difference constraint graph of Figure 3. Projection is normally only a sensible operation on closed constraint sets. Then indeed $Cl(\Pi_{V'}(CL(V,E))) = \Pi_{V'}(CL(V,E))$, but in general $Cl(\Pi_{V'}(V,E))$ is not always equal to $Cl(\Pi_{V'}(CL(V,E)))$.



Event Zones and the $\lesssim_{EZ}$ Relation

In this subsection, the link between clocked words and difference constraint sets is done in the context of I-normality via *event zones*. Then the right precongruence $\lesssim_{EZ}$ is defined and some properties of Figure 2 are proved.

Let I be an independence relation which respects I_Σ, the independence relation for some distributed alphabet $\Sigma = (\Sigma_1, \ldots, \Sigma_l)$. Let $\omega = \alpha_1 \ldots \alpha_n$ be some fixed clocked word with $\alpha_i = (a_i, c_i, r_i)$. For each position i of ω we associate an event variable x_i which corresponds to a time stamp, plus an additional x_0 for the initial stamp. Since the edge labels in difference constraint graphs are couples (constant, sign), we need functions extracting from the clock constraint intervals the upper and lower (actually its opposite) bounds together with their sign:

$upper((c_1, c_2[) = (c_2, <)$ and $upper((c_1, c_2]) = (c_2, \leq)$
$lower(]c_1, c_2)) = (-c_1, <)$ and $upper([c_1, c_2)) = (-c_1, \leq)$ (note the $-$ sign)

Then the difference constraint set $S_\omega = (V_\omega, E_\omega)$ associated to ω to check for its I-realisability is defined over $V_\omega = \{x_0, x_1, \ldots, x_{|\omega|}\}$ and E_ω such that for all $x_i, x_j \in V$ $E_\omega(x_i, x_j) = min\{(m, \prec) \mid x_i - x_j \prec m \in A_\omega\}$ with A_ω the following set of constraints:

respect clock constraints: for k, l with $l = last_C(\alpha_1 \dots \alpha_{k-1})$ for some $C \in \mathcal{C}$
$x_k - x_l \prec m \in A_\omega$ and $(m, \prec) = upper(c_k(C))$
$x_l - x_k \prec m \in A_\omega$ and $(m, \prec) = lower(c_k(C))$
I_Σ-normality: for k, l with $l = last_i(a_1 \dots a_{k-1})$ for some $i \in loc(a_k)$ $x_l - x_k \leq 0 \in A_\omega$
totality: $x_i - x_i \leq 0 \in A_\omega$ and $x_i - x_j < +\infty \in A_\omega$.

The difference constraint set $S_{\alpha\beta}$ of Figure 3 is the difference constraint set associated to the clocked word $\alpha\beta$ of the timed automaton of Figure 1.

The difference constraint set S_ω does not convey enough information about the link between variables and clocks or components, since it gives no way to know which variable corresponds to the last occurrences of clock reset or of some action of Σ_i. For this purpose, we define the function $Last$ which given a clock or a component returns its last variable occurrence, i.e. its last reset or participation. As an example, the $Last$ function of $S_{\alpha\beta}$ of Figure 3 is such that $Last_{\alpha\beta}(C_1) = x_2$ and $Last_{\alpha\beta}(2) = x_0$ for $(\Sigma_1 = \{a, b, e\}, \Sigma_2 = \{c, d, e\})$. Actually, the $Last$ function is built from the two functions $last_C$ and $last_i$ which return the last positions in clocked words of last reset or component participation. For instance, $1 = last_{C_1}(\alpha\beta)$ implies $Last_{\alpha\beta}(C_1) = x_1$. A difference constraint set enriched by its $Last$ function is called *event zone*:

Definition 12 (event zone). The *event zone* for ω is a triple $Z_\omega = (V_\omega, E_\omega, Last_\omega)$ where $V_\omega = \{x_0, x_1, \dots, x_{|\omega|}\}$, $Last_\omega : \mathcal{C} \cup Comp \rightarrow V_\omega$ is the function which gives the last event variable occurrence of a clock C or an action of Σ_i i.e. $Last_\omega(C) = x_i$ such that $i = last_C(\omega)$ and $Last_\omega(i) = x_j$ such that $j = last_i(\omega)$, and $(V_\omega, E_\omega) = S_\omega$ is the difference constraint set associated to ω.

The closure of the event zone $Z_\omega = (V_\omega, E_\omega, Last_\omega)$ is simply $Cl(Z_\omega) = (Cl(V_\omega, E_\omega), Last_\omega)$ and the projection is $\Pi_{V'}(Z_\omega) = (\Pi_{V'}(V_\omega, E_\omega), Last_\omega)$.

Proposition 13. *An event zone Z_ω is consistent iff its associated difference constraint set S_ω is consistent.*

Proof. Starihtforward by Definition 12. □

By construction an event zone Z_ω is consistent iff ω is I-realisable (see Definition 5.) In the sequel, for a word ω let us denote its associated event zone by $Z_\omega = (V_\omega, E_\omega, Last_\omega)$ and the corresponding closure by $Z_{Cl(\omega)} = Cl(Z_\omega) = (V_\omega, E_{Cl(\omega)}, Last_\omega)$.

On the level of consistent event zones, a sufficient criterion for a relation $Z_{\omega_1} \lesssim_{EZ} Z_{\omega_2}$ to be a right precongruence is that the difference constraints between variables representing last occurrences in $Z_{Cl(\omega_1)}$ (i.e. in the codomain of $Last_{\omega_1}$) are tighter than the constraints between the corresponding last occurrences in $Z_{Cl(\omega_2)}$(i.e. in the codomain of $Last_{\omega_2}$.)[4]. In other words, the variables not refering the last reset or component participation in a clocked word are useless for the I-realisability of its extension. The following event zone precongruence is built in this sense:

[4] This criterion can be seen as the zone inclusion in classical timed automata.

Definition 14 (event zone precongruence). Let ω_1, ω_2 be two clocked words over Δ. The event zone precongruence is defined in the following way: $Z_{\omega_1} \lesssim_{EZ} Z_{\omega_2}$ iff Z_{ω_1} and Z_{ω_2} are both inconsistent or Z_{ω_1} is inconsistent and Z_{ω_2} is consistent or else there are both consistent and for all $\xi_1, \xi_2 \in \mathcal{C} \cup Comp$, $E_{Cl(\omega_1)}(Last_{\omega_1}(\xi_1), Last_{\omega_1}(\xi_2)) \leq E_{Cl(\omega_2)}(Last_{\omega_2}(\xi_1), Last_{\omega_2}(\xi_2))$.

By extension, we say that $\omega_1 \lesssim_{EZ} \omega_2$ iff $Z_{\omega_1} \lesssim_{EZ} Z_{\omega_2}$. and we get the following properties (see Figure 2):

Proposition 15. *Let ω_1, ω_2 be two clocked words. Then (i) $\omega_1 \simeq_M \omega_2$ implies $\omega_1 \lesssim_{EZ} \omega_2$, (ii) $\lesssim_{EZ}$ is a right precongruence, (iii) $\omega_1 \lesssim_{EZ} \omega_2$ implies $\omega_1 \lesssim_I \omega_2$.*

Proof.

(i) Assume $\omega_1 = \alpha_1 \dots \alpha_n \simeq_M \omega_2$. Then let π be the permutation such that $\omega_2 = \alpha_{\pi(1)} \dots \alpha_{\pi(n)}$ as defined in Lemma 3. Let $x_1, x_2, \dots$ and $y_1, y_2, \dots$ be the variables of the event zones Z_{ω_1} and Z_{ω_2} respectively where x_i, y_i correspond to α_i. By definition $E_{Cl(\omega_1)}(x_i, x_j) = E_{Cl(\omega_2)}(y_{\pi(i)}, y_{\pi(j)})$. Therefore the event zones and their closure are isomorphic yielding that $\omega_1 \lesssim_{EZ} \omega_2$.

(ii) Assume that $\omega_1 \lesssim_{EZ} \omega_2$ and let $\alpha \in \Delta$. If ω_1 and ω_2 are inconsistent then $\omega_1\alpha$ and $\omega_2\alpha$ are also inconsistent and we are done. The same situation occurs if $\omega_1\alpha$ is not consistent and $\omega_2\alpha$ is consistent.
Let us assume that ω_1 and ω_2 are consistent. Let $\xi, \xi' \in \mathcal{C} \cup Comp$. We have to compare

$$E_{Cl(\omega_1\alpha)}(Last_{Cl(\omega_1\alpha)}(\xi), Last_{Cl(\omega_1\alpha)}(\xi'))$$

and

$$E_{Cl(\omega_2\alpha)}(Last_{Cl(\omega_2\alpha)}(\xi), Last_{Cl(\omega_2\alpha)}(\xi'))$$

Let $V_{\omega_1} = \{x_1, \dots, x_n\}$ and x_α be the new variable introduced for α. Similary we define $V_{\omega_2} = \{y_1, \dots, y_m\}$ and y_α.
Let us consider $Z_{\omega_1}, Z_{\omega_2}$ together with $Z_{\omega_1\alpha}, Z_{\omega_2\alpha}$ as constraint graphes. In this view, let $E_{Cl(\omega_2\alpha)}(Last_{\omega_2\alpha}(\xi), Last_{\omega_2\alpha}(\xi'))$ be the length of a path joining $y = Last_{\omega_2\alpha}(\xi)$ and $y' = Last_{\omega_2\alpha}(\xi')$ that is such that $y \neq y_\alpha$ and $y' \neq y_\alpha$. We want to show that there exists a path in $Z_{\omega_1\alpha}$ from $x = Last_{\omega_1\alpha}(\xi)$ to $x' = Last_{\omega_1\alpha}(\xi')$ which is less weighted. The proof is based on an induction on the occurrence number of y_α in the path from y to y':

- There is no occurrence: The path goes only through vertices of Z_{ω_2} and is weigthed by $E_{Cl(\omega_2)}(Last_{\omega_2\alpha}(\xi), Last_{\omega_2\alpha}(\xi'))$ which is tighter than $E_{Cl(\omega_1)}(Last_{\omega_1\alpha}(\xi), Last_{\omega_1\alpha}(\xi))$ as $\omega_1 \lesssim_{EZ} \omega_2$
- The path from y to y' passes through y_α.
  Let us decompose the path from y to y' into a path from y to y_k then an edge from $y_k = Last_{\omega_2}(\xi_i)$ to $y_\alpha = Last_{\omega_2\alpha}(\xi_i)$, followed by an edge from $y_\alpha = Last_{\omega_2}(\xi_j)$ to $y_p = Last_{\omega_2\alpha}(\xi_j)$, and finally a path from y_p to y'. The paths from y to y_k and this from y_p to y' contain less occurrence of y_α than the path from y to y', and by induction hypothesis we have that $E_{Cl(\omega_2\alpha)}(y, y_k) \leq E_{Cl(\omega_1\alpha)}(x, Last_{\omega_1}(\xi_i))$ (1) and $E_{Cl(\omega_2\alpha)}(y_p, y') \leq E_{Cl(\omega_1\alpha)}(Last_{\omega_1}(\xi_j), x')$ (2). Moreover, $E_{Cl(\omega_2\alpha)}(y_k, y_\alpha)$ is a constraint

of the form $lower(c_\alpha(\xi_i))$ if ξ_i is a clock or $(0, \leq)$ if it is a component. Finally, $E_{Cl(\omega_1\alpha)}(Last_{\omega_1\alpha}(\xi_i), x_\alpha)$ is the minimal value of $lower(\xi'')$ or $(0, \leq)$ such that $Last_{\omega_1}(\xi'') = Last_{\omega_1}(\xi)$ and $Last_{\omega_1\alpha}(\xi'') = x_\alpha$. In particular $E_{Cl(\omega_1\alpha)}(Last_{\omega_1\alpha}(\xi_i), x_\alpha) \leq E_{Cl(\omega_2\alpha)}(y_k, y_\alpha)$ (3). The same reasoning holds for ξ_j –with only the case that $E_{Cl(\omega_2\alpha)}(y_\alpha, y_p)$ is of the form $upper(c_\alpha(\xi_j))$– leading to $E_{Cl(\omega_1\alpha)}(x_\alpha, Last_{\omega_1\alpha}(\xi_j)) \leq E_{Cl(\omega_2\alpha)}(y_\alpha, y_p)$ (4). Finally, additioning the four inequalities leads to the required property.

(iii) Assume that $\omega_1 \lesssim_{EZ} \omega_2$. For all ω such that $\omega_1\omega$ is I-realisable, we have that $Z_{\omega_1\omega}$ is consistent (by proposition 13) and since $\lesssim_{EZ}$ is a congruence (as proved above), we have that $\omega_1\omega \lesssim_{EZ} \omega_2\omega$. Therefore $\omega_2\omega$ is I-realisable i.e. $\omega_1 \lesssim_I \omega_2$.

□

The Event Zone Automaton

This subsection is devoted to the construction of an asynchronous automaton implementing $\lesssim_{EZ}$ with a finite number of variables, those corresponding to last occurrences of reset or component participation.

The first step is to construct event zones in an incremental manner. Informally, if Z_ω is an event zone and α is a clocked label then getting $Z_{\omega\alpha}$ turns out to add a fresh variable for α position in ω to Z_ω and some *non trivial* difference constraints $A_{\omega\odot\alpha}$ between the new variable and these of last resets and component participations. This extension operation on Z_ω is denoted $Z_\omega \odot \alpha$.

Definition 16. An *extension of an event zone* $Z_\omega = (V = \{x_0, \ldots, x_n\}, E, Last)$ of ω by $\alpha = (a, c, r) \in \Delta$, denoted $Z_\omega \odot \alpha$, is the triple $(V', E', Last')$ such that:
(i) **The difference constraint set is extended:** $V' = V \cup \{x_{n+1}\}$ and E' is defined by: $E'(x_i, x_j) = E(x_i, x_j)$ for all $x_i, x_j \neq x_{n+1}$,
$E'(x_{n+1}, x_i) = min\{(m, \prec) \mid x_{n+1} - x_i \prec m \in A_{\omega\odot\alpha}\}$
$E'(x_i, x_{n+1}) = min\{(m, \prec) \mid x_i - x_{n+1} \prec m \in A_{\omega\odot\alpha}\}$
with $A_{\omega\odot\alpha}$ the following set of difference constraints:

clock constraint condition: For all $x_l = Last(C)$ with C a clock,
$x_{n+1} - x_l \prec m \in A_{\omega\odot\alpha}$ and $(m, \prec) = upper(c(C))$
$x_l - x_{n+1} \prec m \in A_{\omega\odot\alpha}$ and $(m, \prec) = lower(c(C))$
I_Σ-normality: For all $x_l = Last(i)$ with $i \in loc(a)$ $x_l - x_{n+1} \leq 0 \in A_{\omega\odot\alpha}$
totality: for all $x_i \in V$, $x_i - x_{n+1} < +\infty, x_{n+1} - x_i < +\infty \in A_{\omega\odot\alpha}$,
and $x_{n+1} - x_{n+1} \leq 0 \in A_{\omega\odot\alpha}$.

(ii) **Last occurrences are updated:** if $i \in loc(a)$ then $Last'(i) = x_{n+1}$, if $C \in r$ then $Last'(C) = x_{n+1}$, otherwise $Last'(\xi) = Last(\xi)$ for $\xi \in \mathcal{C} \cup Comp$.

By definition, we get $Z_\omega \odot \alpha \simeq_{EZ} Z_{\omega\alpha}$ which justifies the use of $Z_\omega \odot \alpha$ instead of $Z_{\omega\alpha}$ to check for I-realisability.

In the sequel, as the event zones are now incrementaly built, the ω subscript is omitted in the event zone notation. The second step is to justify the practical use of $\lesssim_{EZ}$, that is to find a data structure with a bounded number of variables to represent the I-realisations of all the clocked words (currently this number is equal to the clocked word length). The next proposition aims at proving that if an event zone Z is closed and *consistent* then it is sufficient to consider difference constraints between variables of last occurrences (i.e. in the codomain of its $Last$ function) to check if Z can be extended. Formally, let $last(Z_\omega)$ denote the projection of the closed zone $Cl(Z_\omega) = Cl(V_\omega, E_\omega, Last_\omega)$ on V_{last}, the codomain of $Last_\omega$. That is $last(Z_\omega) = \Pi_{V_{last}}(Cl(V_\omega, E_\omega), Last_\omega)$. As an example, $last(Z_{\alpha\beta})$ ($Z_{\alpha\beta}$ is depicted on the left part of Figure 3) is the projection of the closure $Cl(Z_{\alpha\beta})$ (right part of Figure 3) on the set $V_{last} = \{x_0, x_2\}$ (Figure below the Figure 3). The abstraction $last(Z)$ from Z limits the number of relevant variables in event zones to $|\mathcal{C}| + |Comp|$. Moreover, this projection behaves well with respect to extension and $\lesssim_{EZ}$:

Proposition 17. *Let Z be a consistent event zone and α be a clocked label. Then $last(Z \odot \alpha)) \simeq_{EZ} last(last(Z) \odot \alpha)$.*

Proof. We recall that the function $last(Z)$ computes the closure of Z before projecting on the set of last event variables. For S a zone being any of $Z, Z \odot \alpha, Cl(Z), \ldots$, we denote by $E_S(x, y)$ the value of the function E and we write $x \in V_Z$ for $x \in Z$. By definition of the closure and extension of zones, $last(Z \odot \alpha)$ and $last(last(Z) \odot \alpha))$ have the same set of variables which contains x_α the variable corresponding to α. By definition the value of $E_{Cl(S)}(x, y)$ is the minimum of the length of paths of S starting from x and ending in y.

Let us consider a minimal weighted path in $Z \odot \alpha$ starting from x ending in y (it can be the case that $x = x_\alpha$ or $y = x_\alpha$), such that $x, y \in last(Z \odot \alpha)$. Note first that the case where the path does not goes through x_α is obvious. Moreover among all the minimal path there exists once going only once through x_α as in a consistent event zone is consistent every cycle (containing x_α for instance) is of positive weight. Such a path is depicted in Figure 4.

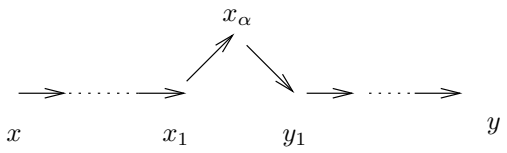


Fig. 4. a path in $Z \odot \alpha$

If x_1 or y_1 is not in $last(Z)$, by definition $E(x_\alpha, y_1)$ or $E(x_1, x_\alpha)$ is $(+\infty, <)$, therefore the length of the path is $(+\infty, <)$. Therefore either $E_{Cl(Z \odot \alpha)}(x, y) = (+\infty, <)$ and the same holds for $E_{Cl(Cl(Z) \odot \alpha)}(x, y)$ or it is finite. In the later case,

we can restrict ourselves to the paths of length different from $(+\infty, <)$, hence we can assume that $x_1, y_1 \in last(Z)$. By minimality of subpaths of an optimal path and because of the previous remark on the occurrence number of x_α, we have that the length of the path joining x to x_1 is $E_{Cl(Z)}(x, x_1)$ and that the length of the path joining y_1 to y is $E_{Cl(Z)}(y_1, y)$. Since $E_{Cl(Z)\odot\alpha}(x_1, x_\alpha) \geq E_{Cl(Z)\odot\alpha}(x_1, x_\alpha)$ and $E_{Cl(Z)\odot\alpha}(x_\alpha, x_1) \geq E_{Cl(Z)\odot\alpha}(x_\alpha, y_1)$, we get that

$$E_{Cl(Z\odot\alpha)}(x, y)) \geq E_{Cl(Cl(Z)\odot\alpha)}(x, y)$$

Conversely, a minimal path in $Cl(last(Z) \odot \alpha)$ is depicted in Figure 5 where

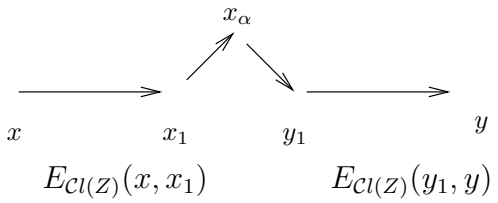


Fig. 5. a path in $Cl(Z) \odot \alpha$

$E_{Cl(Z)}(x, x_1)$ is the length of a minimal path joining x to x_1 in Z (and similary for $E_{Cl(Z)}(y_1, y)$). Moreover $E_{Cl(Z)\odot\alpha}(x_1, x_\alpha)$ is greater than or equal to $E_{Cl(Z\odot\alpha)}(x_1, x_\alpha)$ and similarly $E_{Cl(Z)\odot\alpha}(x_\alpha, y_1) \geq E_{Cl(Z\odot\alpha)}(x_\alpha, y_1)$. Therefore, the length of all minimal paths is greater than or equal to $E_{Cl(Z\odot\alpha)}(x, y))$ yielding:

$$E_{Cl(Cl(Z)\odot\alpha)}(x, y) \geq E_{Cl(Z\odot\alpha)}(x, y))$$

□

This justifies the use of last(Z) to define the event automaton in the following construction where $\mathcal{Z}$ denotes the set of event zones over Δ and Z_ϵ is the special event zone $(V_\epsilon, E_\epsilon, Last_\epsilon)$ associated to the empty word such that $V_\epsilon = \{x_0\}$ (the initial time stamp), $E(x_0, x_0) = (0, \leq)$ and $Last(\xi) = x_0$ for all $\xi \in C \cup Comp$ (everything is reset).

Definition 18 (event zone automaton). The event zone automaton $\mathcal{A}' = (S', s'_0, \rightarrow', F')$ associated to an asynchronous timed automaton $\mathcal{A} = (S, s_0, \rightarrow, F)$ is such that $S' = S \times \mathcal{Z}/_{\simeq_{EZ}}$, couples of discrete states and (quotients of) event zones, the initial state is $s'_0 = (s_0, [Z_\epsilon])$, the set of final states is $F' = \{(s, Z) \mid s \in F\}$ and the transition relation $\rightarrow'$: $S' \times \Delta \hookrightarrow S'$, is defined by $(s, [Z]) \xrightarrow{\alpha} (s_1, [Z_1])$ iff $s \xrightarrow{\alpha} s_1$ is in $\mathcal{A}$ and $Z_1 = last(Z \odot \alpha)$ is consistent.

Proposition 19. *The event zone automaton for an asynchronous timed automaton is an asynchronous timed automaton accepting exactly the clocked words having an I-realisation.*

Proof. A straightforward structural induction on ω shows that if $s'_0.\omega = (s, Z)$ then Z_ω is consistent (use proposition 17) hence ω is I-realisable.

Similarly a structural induction on ω proves that if ω is I-realisable, then $s'_0.\omega = (s, Z)$ with $Z = last(Z_\omega)$ (use propositions 13, 15 and 17).

By theorem 6, Z_{ω_1} and Z_{ω_2} are isomorphic if $\omega_1 \simeq_M \omega_2$. Combined with the properties ID and FD of asynchronous automata, this ensures that $s'_0.\omega = s'_0.\omega_2$ □

6 Catchup preorder for language emptiness checking

In this section, we introduce the finite index *catchup preorder* and show how it can be used to obtain a bounded complexity algorithm for non-emptiness of timed automata languages.

6.1 A useful tool, the separator action $

For the rest of this section, we introduce a useful tool, a separator:

Definition 20 (separator $). Any clocked alphabet Δ with dependence relation I can be trivially extended to Γ with $\Delta \cup \{\$\}$ (possibly $\$ \in \Delta$), $\$ = (a, c, r)$ called *separator*, such that c is trivial (no conditions on any clocks), $r = \emptyset$ (no resets), and for all $\alpha \in \Delta$ it holds that $\alpha \not I_\Delta \$$.

The separator $, being dependent of all actions but being trivial on the clocks, does not modify (except for shifting by one position) the clock constraints in $\alpha_1 \dots \alpha_n \$ \beta_1 \dots \beta_m$ compared to $\alpha_1 \dots \alpha_n \beta_1 \dots \beta_m$, but its dependency on every other clocked label imposes that every α_i must happen before every β_j. It separates in a clocked word temporal past and future. Lemma 21 shows the use of the separator respect to $\lesssim_{IN}$:

Lemma 21. *For all $\omega_1, \omega_2 \in \Gamma^*$, $\omega_2 = \alpha_1 \dots \alpha_n$ it holds that there exists $\omega'_1 \in [\omega_1]$ with $\omega'_1\omega_2$ is realisable iff $\omega_1\$\alpha_1\$\alpha_2\$ \dots \$\alpha_n$ is I-realisable.*

Proof. The $\$ = (a, c, r)$ we require is such that c is trivial (no conditions on any clocks), $r = \emptyset$ (no resets), and for all $\alpha \in \Delta$ it holds that $\alpha \, D_\Sigma \, \gamma$. Either such a $ already exists in $\Delta =: \Gamma$ or we take a fresh $a \notin \Sigma$ and set $\Gamma := \Delta \cup \{\$\}$ and extend I_Σ accordingly.

Now suppose that there exists $\omega'_1 \in [\omega_1]$ such that $\omega'_1\omega_2$ has a normal realisation. Then $\omega'_1\$\alpha_1\$\alpha_2\$ \dots \$\alpha_n$ also has a normal realisation by letting the $ before α_i occur at the same time as α_i. Then according to Theorem 6 and Corrollary 7, the equivalent $\omega_1\$\alpha_1\$\alpha_2\$ \dots \$\alpha_n$ is I-realisable.

Conversely, if $\omega_1\$\alpha_1\$\alpha_2\$ \dots \$\alpha_n$ is I-realisable, then according to Proposition 4, there exists $\omega' \in [\omega_1\$\alpha_1\$\alpha_2\$ \dots \$\alpha_n]$ having a normal realisation. Since $ is dependent of every other label, this word must be of the form $\omega' = \omega'_1\$\alpha_1\$ \dots \$\alpha_n$, where $\omega'_1 \simeq_M \omega_1$. Projecting away the additional occurrences of $, $\omega'_1\omega_2$ has a normal realisation. □

From now on, we will assume that Δ already contains such a \$ as described in Lemma 21. Note that in all previous arguments, the presence or absense of such a \$ plays no rôle whatsoever. But in what follows, it is a useful tool in some situations. In particular, it gives us:

Proposition 22. *If Δ contains a separator \$, then $\omega_1 \lesssim_I \omega_2$ implies $\omega_1 \lesssim_{IN} \omega_2$.*

Proof. We assume $\omega_1 \lesssim_I \omega_2$. Now choose an arbitrary $\omega = \alpha_1\alpha_2 \dots \alpha_n$ such that $\omega_1'\omega$ has a normal realisation for some $\omega_1' \in [\omega_1]$. By Lemma 21, $\omega_1\$\alpha_1\$\alpha_2 \dots \$\alpha_n$ is I-realisable. By $\omega_1 \lesssim_I \omega_2$ and the right precongruence property of $\lesssim_I$, also $\omega_2\$\alpha_1\$\alpha_2 \dots \$\alpha_n$ is I-realisable. Again by Lemma 21, there exist $\omega_2' \in [\omega_2]$ such that $\omega_2'\omega$ is realisable. Since the choice of ω was arbitrary, we have indeed shown that $\omega_1 \lesssim_{IN} \omega_2$. □

6.2 Catchup equivalence

In the previous section we have developed a representation of $\lesssim_I$. The representation uses event zones which are matrices of bounded dimension, but with unbounded entries. Hence, there may be an infinity of such zones, which is unavoidable, because $\lesssim_I$ itself is of infinite index. In terms of right congruences, we understand zones as representations of equivalence classes $\simeq_{EZ}$.

At this point, we have that $\lesssim_{EZ} \subseteq \lesssim_I \subseteq \lesssim_{IN}$. However, we neither know how to test $\lesssim_I$ nor $\lesssim_{IN}$ and we have no information about the index of $\lesssim_{IN}$. Next, we abstract/relax $\lesssim_{EZ}$ in a manner to still respect $\lesssim_{IN}$. More precisely, we give a sufficient criterion for two traces $[\omega_1], [\omega_2]$ with $[\omega_1] \lesssim_{EZ} [\omega_2]$ also to satisfy $[\omega_1] \lesssim_{IN} [\omega_2]$. As explained before, constraints in the event zones for a pair of variables/events are pairs $(c, <)$ or $(c, \leq)$ 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 $\lesssim_N$.

Definition 23 (catchup simulation of event zones). Let ω_1, ω_2 be two clocked words and let $Z_{\omega_1\$} = (V_1, E_1, Last_1)$ and $Z_{\omega_2\$} = (V_2, E_2, Last_2)$ the event zones for $\omega_1\$, \omega_2\$$ respectively, where \$ is a separator. Moreover, for all pairs $\xi_1 \in \mathcal{C}$, $\xi_2 \in \mathcal{C} \cup \{1\}$ ($1 \in Comp$[5]):

- $E(Last(\xi_1), Last(\xi_2)) \leq E'(Last'(\xi_1), Last'(\xi_2))$;
- or $E(Last(\xi_1), Last(1)), E'(Last'(\xi_1), Last'(1))$ (constraint between clock reset events and the separator) are both strictly smaller than $(-c, \prec)$ for the greatest non-trivial upper bound $(c, \prec)$ for ξ_1 in Δ (upper catchup);
- or both $E(Last(\xi_1), Last(\xi_2))$, $E'(Last'(\xi_1), Last'(\xi_2))$ greater or equal to the biggest lower bound for ξ_2 in Δ (lower catchup).

[5] This choice is arbitrary, the last action for any component is \$

Then we write that $\omega_1 \lesssim_C \omega_2$ (and say that ω_2 catchup simulates ω_1. Moreover $\omega_1 \simeq_C \omega_2$ (catchup equivalent) iff $\omega_1 \lesssim_C \omega_2$ and $\omega_2 \lesssim_C \omega_1$.

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 event (e.g. events that would have occur 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).

Theorem 24. *$\omega_1 \lesssim_C \omega_2$ implies $\omega_1 \lesssim_{IN} \omega_2$.*

Proof. Let $\omega_1 \lesssim_C \omega_2$, $Z_{\omega_1\$} = (V_1, E_1, Last_1)$ and $Z_{\omega_2\$} = (V_2, E_2, Last_2)$ the corresponding zones of $\omega_1\$$ and $\omega_2\$$. To show $\omega_1 \lesssim_{IN} \omega_2$, let $\omega = \alpha_1 \dots \alpha_n$ such that there exists $\omega_1' \in [w_1]$ and $\omega_1'\omega$ has a normal realisation. We have to show that also there exists $\omega_2' \in [\omega_2]$ and $\omega_2'\omega$ a normal realisation.

According to Lemma 21, we have that $\omega_1\$\alpha_1\$\alpha_2 \dots \$\alpha_n$ is I-realisable. We show that $\omega_2\$\alpha_1\$\alpha_2 \dots \$\alpha_n$ is I-realisable as well. Assuming the contrary, there exists a negative cycle in the event zone corresponding to $\omega_2\$\alpha_1\$\alpha_2 \dots \$\alpha_n$ that must pass through both the variables corresponding to $\omega_2\$$ and the variables corresponding to $\alpha_1\$\alpha_2 \dots \$\alpha_n$ (otherwise either $Z_{\omega_2\$}$ is already inconsistent or there is a negative cycle within $\alpha_1\$\alpha_2 \dots \$\alpha_n$, which contradicts the assumption that $\omega_1\$\alpha_1\$\alpha_2 \dots \$\alpha_n$ is I-realisable). Let X be the set of variables corresponding to $\omega_2\$$ and Y be the set of variables corresponding to $\alpha_1\$\alpha_2 \dots \$\alpha_n$. A negative cycle constituated by an alternation

$$y_1^1 y_2^1 ... y_{m_1}^1 x_1^1 x_2^1 ... x_{n_1}^1 y_1^2 ... y_{m_2}^2 x_1^2 ... x_{n_2}^2 ... y_1^k ... y_{m_k}^k x_1^k ... x_{n_k}^k y_1^1$$

where $x_i^j \in X$, $y_i^j \in Y$, $m_i, n_i > 0$, and $k \geq 1$, and the sum of the edges on the cycle is negative. Let us assume k minimal. Our aim is to find a negative cycle for $\omega_1\$\alpha_1\$\alpha_2 ... \$\alpha_n$ also. Let Z be the set of variables corresponding to $\omega_1\$$. The idea is to replace parts of the path $x_{n_l}^l y_1^{l+1} ... y_{m_{l+1}}^{l+1} x_1^{l+1}$ by parts $x_{n_l}^l z_1^{l+1} ... z_{o_{l+1}}^{l+1} x_1^{l+1}$ (or in one case $x_{n_l}^l z_1^{l+1} ... z_{o_{l+1}}^{l+1} x_{-p_{l+1}}^{l+1} ... x_0^{l+1} x_1^{l+1}$ with some additional variables from X at the end) with a smaller sum of the edges. Then, we obtain a cycle for $\omega_1\$\alpha_1\$\alpha_2 ... \$\alpha_n$ with a total sum of edges smaller than the total sum of edges for $\omega_2\$\alpha_1\$\alpha_2 ... \$\alpha_n$, hence a negative cycle contradicting realisability.

Let the edge $x_{n_l}^l y_1^{l+1}$ result from a constraint $C \leq c$ for the label α corresponding to $x_{n_l}^l$ for some clock C such that $Last_2(C) = y_1^{l+1}$ in Z_1. Likewise let the edge $y_{m_{l+1}}^{l+1} x_1^{l+1}$ result from a constraint $D \geq d$ (or from a causality constraint) with $Last_2(D) = y_{m_{l+1}}^{l+1}$. We do a case analysis, following the definition of $\lesssim_C$:

- $E_1(Last_1(C), Last_1(D)) \leq E_2(Last_2(C), Last_2(D))$; by definition of E_2, the actual length of path $z_1^{l+1} ... z_{o_{l+1}}^{l+1}$ with $Last_1(C) = z_1^{l+1}$ and $Last_2(D) = z_{o_{l+1}}^{l+1}$ is of length $E_1(Last_1(C), Last_1(D))$. Combining it with the edges $x_{n_l}^l z_1^{l+1}$

and $z_{o_{l+1}}^{l+1} x_1^{l+1}$ that have the same lengths as $x_{n_l}^l y_1^{l+1}$ and $y_{m_{l+1}}^{l+1} x_1^{l+1}$ respectively, we obtain that $x_{n_l}^l z_1^{l+1} ... z_{o_{l+1}}^{l+1} x_1^{l+1}$ is shorter than $x_{n_l}^l y_1^{l+1} ... y_{m_{l+1}}^{l+1} x_1^{l+1}$, as desired.

- There exist ξ_1, ξ_2 such that $E_1(Last_1(C), Last_1(\xi_1))$, $E_2(Last_2(C),\ Last_2(\xi_2)) < (-c, \leq)$. Then we directy find a negative cycle for $\omega_1 \$ \alpha_1 \$ \alpha_2 ... \$ \alpha_n$ as follows: Here, we exploit the fact that \$ depends on every other label, because, the resulting causality constraints give us a path of length $(0, \leq)$ from any vertex in Z to any vertex in X, in particular from $Last_1(\xi_1)$ to $x_{n_l}^l$. From $x_{n_l}^l$ to $Last_1(C)$ we have the edge $(c, \leq)$ and from $Last_1(C)$ to $Last_1(\xi_1)$ we have a path strictly shorter than $(-c, \leq)$. This gives us a negative cycle contradicting the assumption that $\omega_1 \$ \alpha_1 \$ \alpha_2 ... \$ \alpha_n$ has a realisation respecting I_Σ.
- Both $E_1(Last_1(C), Last_1(D))$ and $E_2(Last_2(C), Last_2(D))$ are bigger than $(d, \leq)$. Then the length of the path $y_1^{l+1} ... y_{m_{l+1}}^{l+1} x_1^{l+1}$ (without the initial edge $x_{n_l}^l y_1^{l+1}$ is greater than $(0, \leq)$. As in the previous case, there exists a path of length $(0, \leq)$ from $Last_1(C)$ to x_1^{l+1}, due to a causality chain. For easier understanding, let us add that this path is of the form $z_1^{l+1} ... z_{o_{l+1}}^{l+1} x_{-p_{l+1}}^{l+1} ... x_0^{l+1} x_1^{l+1}$ where $z_1^{l+1} = Last_1(C)$, i.e. the causality chain inducing zero weight edges has a part in Z followed by a part in X). Concluding, we obtain $x_{n_l}^l z_1^{l+1} ... z_{o_{l+1}}^{l+1} x_{-p_{l+1}}^{l+1} ... x_0^{l+1} x_1^{l+1}$ shorter than $x_{n_l}^l y_1^{l+1} ... y_{m_{l+1}}^{l+1} x_1^{l+1}$.

On the whole, we thus obtain a cycle of shorter, hence negative, length for $\omega_1 \$ \alpha_1 \$ \alpha_2 ... \$ \alpha_n$, contradicting the assumption of a realisation respecting I_Σ. Therefore the assumption that $\omega_2 \$ \alpha_1 \$ \alpha_2 ... \$ \alpha_n$ has no realisation respecting I_Σ leads to a contradiction. Therefore it is realizable and by Lemma 21, there exists $\omega_2' \in [\omega_2]$ so that $\omega_2' \omega$ has a normal realisation, which proves that $\omega_1 \lesssim_{IN} \omega_2$. □

Proposition 25. *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* $(4K + 3)^{n(n+1)}$.

Proof. Two traces ω_1, ω_2 are distinguished by $\simeq_C$ iff for the corresponding zones $Z_{\omega_1 \$}, Z_{\omega_2 \$}$ of Definition 23 there exist ξ, ξ' such that one can distinguish the entries $E_1(Last_1(\xi), Last_1(\xi'))$ and $E_2(Last_2(\xi), Last_2(\xi'))$. These entries are not distinguished, if both entries are strictly smaller than $(-c, \prec)$ where $(c, \prec)$ is the greatest non-trivial upper bound for ξ or both are greater or equal to the greatest lower bound for ξ' or if they are in between and have the same value. For a finite Δ, this gives a finite number of separations per pair ξ, ξ'.

Since \$ belongs to each Δ_i, the dimension of the event zones in Definition 23 is bounded to $n+1$ (n for the clock resets, 1 for the $last_i$ all pointing to the same event corresponding to \$). Let K be the overall greatest constant mentioned in the timed automaton, this gives the upper bound of $4K + 3$ ($(\pm d, \prec)$, special cases $(0, <)$, $(0, \leq)$, $(+\infty, <)$) distinguished entries per pair between $\mathcal{C}_l$ and $\mathcal{C}_u$, hence on the whole the index of $\simeq_C$ is limited by $(4K + 3)^{n(n+1)}$. □

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

6.3 An algorithm for emptiness of $L_I(\mathcal{A})$

Now we are set to give an algorithm for emptiness or reachability analysis. The basis of the algorithm is the (infinite) event zone automaton.

With the help of $\lesssim_C$, we can obtain a correct and terminating algorithm for emptiness of the event zone automaton. We do this rather abstractly with the generic exploration Algorithm 1 without imposing unneccessesary detail[6]. Note, that for readability, we formulate the algorithm with clocked traces as states, but that we could equally have taken states of the event zone automaton (because $\simeq_M \subseteq \lesssim_{EZ}$), which is a good way of implementing the algorithm. Also note that the "red" set in the algorithm is there for the sake of the presentation of the proof (for nice invariants) and that it is not actually needed in an implementation. Some more remarks on implementation will be given in the next section.

The basic idea of the algorithm is to consider a partition of the set of traces in four colours: White traces are those we have not visited yet, gray traces are awaiting exploration, black traces have been explored, in particular concerning successors, and red traces have been rejected because of catchup equivalence. Whereas similar descriptions of depth first search (see e.g. [CLR90]) lead to an exhaustive exploration of finite graphs (with all vertices black in the end), our algorithm is intentionally unexhaustive and leaves some nodes red and an infinity of nodes white, unexplored.

Algorithm 1 is generic with respect to the choice of the element of Gray at the beginning of the while loop. For instance, organising Gray as a stack would result in a DFS-like algorithm, organising Gray as a queue would result in a BFS-like algorithm. Instead of fixing on a strategy here, we show the correctness of the generic algorithm, allowing the use of any kind of heuristic in this choice. Note however, that the strategy applied in this choice may result in a completely different fragment of the event zone automaton to be explored.

Theorem 26. *For an asynchronous timed automaton $\mathcal{A}$, Algorithm 1 terminates and yields a witness $\omega \in L_I(\mathcal{A})$ iff $L_I(\mathcal{A}) \neq \emptyset$ otherwise returns "empty".*

Proof. The proof is based on the following claims:

(Invariant 1) At the beginning of the while-loop, for any two $[\omega_1], [\omega_2] \in \mathsf{Black} \cup \mathsf{Gray}$ either $\omega_1 \simeq_M \omega_2$ or $s_{\omega_1} \neq s_{\omega_2}$ or $\omega_1 \not\simeq_C \omega_2$.

(Termination) The number of while-iterations is limited by the product of $|S|$ (number of states of the timed automaton) and 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-realisable successors $[\omega\alpha]$ (with a run in the timed automaton) of a black $[\omega]$ are coloured (black,gray or red).

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

Algorithm 1 Generic exploration algorithm

```
Gray ← {[ϵ]}
Black ← ∅
Red ← ∅
while Gray ≠ ∅ do
  Choose [ω] ∈ Gray
  Gray ← Gray \ {[ω]}
  Black ← Black ∪ {ω}
  for all ω′ = ωα with (s_ω, α, s_ωα) ∈ → and Z_ωα consistent do
    if ∃[ω″] ∈ Black ∪ Gray.s_ω′ = s_ω″ and ω′ ≲_C ω″ /* or weaker ≃_C */ then
      Red ← Red ∪ {[ω′]}
    else
      /* Gray optimisation */
      if ∃[ω″] ∈ Gray.s_ω′ = s_ω″ and ω″ ≲_C ω′ then
        Gray ← Gray \ {[ω″]}
        Red ← Red ∪ {[ω″]}
      end if
      /* end Gray optimisation */
      Gray ← Gray ∪ {[ω′]}

      if s_ω′ ∈ F then
        return "witness(ω′)"
      end if
    end if
  end for
end while
return "empty"
```

(Invariant 3) All coloured traces are I-realisable.
(Invariant 4) For each red $[\omega]$ there exists a gray or black $[\omega']$ with $\omega \lesssim_C \omega'$.
(Witness) A returned witness really belongs to $L_I(\mathcal{A})$ (because it is I-realisable and leads to a final state).
(No witness) If no witness is returned then $L_I(\mathcal{A}) = \emptyset$.

The claimed invariants, (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 $\omega \in L_N(\mathcal{A})$ and $\omega = \omega_1\omega_2$ such that $[\omega_1]$ black and $|\omega_2|$ minimal.

Either $\omega_2 = \epsilon$, but then $[\omega]$ is black and was added to the gray set at some point, where the algorithm should have returned it as witness.

Or $\omega_2 = \alpha\omega_2'$. Since $[\omega_1]$ is black, its successor $[\omega_1\alpha]$ must be coloured. At termination, there are no gray traces left, so $[\omega_1\alpha]$ is black or red. But $[\omega_1\alpha]$ black would contradict the choice of ω and ω_1, ω_2 (minimality of $|\omega_2|$). Hence $[\omega_1\alpha]$ must be red.

Then there must exist a gray (excluded at termination) or black $[\omega']$ with $\omega\alpha \lesssim_C \omega'$ and $s_{\omega\alpha} = s_{\omega'}$. By definition of $\lesssim_C$ and Proposition 4 this implies that for some $\omega'' \simeq_M \omega'$, $\omega''\omega_2' \in L_N(\mathcal{A})$ with $[\omega'']$ black, again contradicting the assumed minimality of $|\omega_2|$. □

The exploration algorithm is just the central component of the verification system. If a witness ω is actually returned, an I-normal timed word (w, t) should actually be computed, "sorted" to an equivalent normal timed word $(w', t') \in L_T(\mathcal{A})$ according to Proposition 4, which is the actual diagnostic trace to be handed to the user, who need not be aware of the Mazurkiewicz trace approach in the computation of this witness at all.

Algorithmic issues 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 vertex has been visited, but a vertex state that is in relation $\simeq_C$ or $\lesssim_C$ with the current vertex. Hence, it has to be pointed out that we need not hash/store actual couples (s_ω, Z_ω) or traces $[\omega]$ but only a sufficient abstraction $abs(s_\omega, Z_\omega)$ of the latter allowing a test for $(s_{\omega_1}, Z_{\omega_1}) \lesssim_C (s_{\omega_2}, Z_{\omega_2})$ based on[7] $(s_{\omega_1}, Z_{\omega_1})$ and $abs(s_{\omega_2}, Z_{\omega_2})$.

There are two possible applications:

- Rather than checking for $\lesssim_C$, we only check for $\simeq_C$. In this case, we can hash the state s and the zone $Z_{\omega\$}$ where we replace in the matrix entries concerned by the second and third rule of Definition 23 by special values for representing the abstraction. Then, the test for $(s', Z') \simeq_C (s, Z)$ is equivalent to the test $abs(s', Z') = abs(s, Z)$.

[7] The asymmetric definition is intentional: We have to store permanently $abs(s_{\omega_2}, Z_{\omega_2})$, so we want it to occupy as little memory as possible.

Under this assumption, all kinds of hashing techniques can be applied for a compact representation of visited states up to $\simeq_C$.

– In analogy to clock zone inclusion tests known from classical timed automata, we actually want to test $\lesssim_C$, hoping that this will result in a smaller graph. In this case, either a more sophisticated data structure than a hash table must be used or the following compromise can be applied: Search for a pair of abstraction functions abs_1, abs_2 and a function $test$ such that $(s, Z) \lesssim_C (s', Z')$ iff $abs_1(s, Z) = abs_1(s', Z')$ and $test((s, Z), abs_2(s', Z'))$ where we intend abs_1 to be as discriminating as possible, abs_2 as little as , e.g. abs_2 will not depend on s, etc.. Then, a hash table indexed with abs_1 carrying for each value a list of values from abs_2 can be used.
This is a somewhat abstract description of implementations with zone lists for discrete states.

7 Comparison with classical zone automata

Several propositions have been done to consider partial order reductions for timed automata but the results were usually disappointing, yielding the feeling that this research direction was a dead-end. In section 8, we give some figures that validate our approach, and in the present section we give some theoretical grounds to support our claim that the approach is worthwhile.

Firstly, we consider the case without independence and we show that our algorithm computes an object similar to the simulation graph of a timed automaton.

Secondly, we show how the fact that commutation does not lead to incomparable zones is related to the "convex hull overapproximation" in tools like UppAal.

The case of $I = \emptyset$ For the purpose of comparison with classical timed automata constructions, we consider here a fully dependent alphabet.

Proposition 27. *Let $I = \emptyset$. Then $\lesssim_N = \lesssim_{IN} = \lesssim_I$.*

This fact is obvious and illustrates the generalisation arising from moving from the fully dependent case to the asynchronous case. An immediate consequence is:

Corollary 28. *$\lesssim_N$ is of finite index.*

Proof. $\lesssim_N$ is preserved under modifications of the independence relation, in particular also for the case $I = \emptyset$. Then we find that $\lesssim_C \subseteq \lesssim_{IN} = \lesssim_N$. Hence, $\lesssim_N$ has an index limited by the index of $\lesssim_C$. □

However, a much stronger property is true:

Theorem 29. *Let $I = \emptyset$. Then Algorithm 1 can be modified to yield a finite deterministic automaton for $L_N(\mathcal{A})$.*

Proof. We just give a sketch: Instead of using $\lesssim_C$ in Algorithm 1, use $\simeq_C$ for the red colouring. The modified construction will insert a transition from a black state to each of its children. Each time, a state is coloured red, redirect all ingoing transitions to the gray or black state that caused the recolouring. Accepting states of $\mathcal{A}$ become accepting states of the constructed automaton.

The red and the gray/black state causing its recolouring, are catchup equivalent, and hence equivalent with respect to $L(\mathcal{A})$ and $\simeq_{IN}=\simeq_N$, which means that they are equivalent with respect to $\simeq_{L_N(\mathcal{A})}$. The induces right congruence of the constructed automaton is then easily seen to refine $\simeq_{L_N(\mathcal{A})}$. □

In fact, it seems that $\lesssim_C$ itself becomes a right precongruence for the case of $I = \emptyset$. However, the algebraic argument shows that we do not really need this property to construct an automaton for $L_N(\mathcal{A})$. It is sufficient that we have $\lesssim_{EZ} \subseteq \lesssim_C \lesssim_N$, where $\lesssim_{EZ}$ and $\lesssim_N$ are right precongruences and $\lesssim_C$ is a preorder of finite index! States are still constructed using the infinite index $\simeq_{EZ}$, yet $\simeq_C$ is used to have a bounded number of representatives of the classes of $\simeq_N$ only.

Relation of $\lesssim_{IN}$ and the "convex hull overapproximation" UppAal has an option to join incomparable clock zones into a "convex hull[8] In terms of $\lesssim_N$, this is similar to finding for two incomparable zones Z_1, Z_2 a zone Z such that $Z_1 \lesssim_N Z$ and $Z_2 \lesssim_N Z$, where of course we would want to be Z as small as possible with respect to $\lesssim_N$. This "convex hull" is then used to replace the two states $(s, Z_1), (s, Z_2)$ by (s, Z) in the black set of a zone automaton. This operation results in an overapproximation, i.e. a path to an accepting state need not be realisable. But if the aim is to prove language emptiness, this need not disturb: if the automaton with convex hull overapproximation has an empty language, then so has the original automaton.

We will now indicate, in which way our approach is related to the convex hull overapproximation:

Proposition 30. $\omega_1\$\omega_2 \lesssim_{EZ} \omega_1\omega_2$

Proof. Since $\lesssim_{EZ}$ is a precongruence, it is sufficient to prove that $\omega_1\$ \lesssim_{EZ} \omega_1$. For the two event zones $Z_{\omega_1\$} = (V_1, E_1, Last_1)$ and $Z_{\omega_1} = (V_2, E_2, Last_2)$, assume that $V_1 = V_2 \cup \{x_\$\}$ where $x_\$$ is the variable for the occurrence time of \$ and that E_1 restricted to V_2 coincides with V_2. The only additional nontrivial constraints in E_1 are some $E(y, x_\$)$ between some $y \in V_2$ and $x_\$$, notably $E(x_\$, y) = (+\infty, <)$ for all $y \in V_2$.

Then for pairs of clocks C, D it holds that $Last_1(C) = Last_2(C) \in V_2$ and $Last_1(D) = Last_2(D) \in V_2$ and consequently $E_1(Last_1(C), Last_1(D)) = E_2(Last_2(C), Last_2(D))$. But for pairs of a clock C and some component $i \in Comp$, it holds that

[8] Not convex in the geometric sense, but rather the smallest difference constraint set including two incomparable difference constraint sets.

$$E_1(Last_1(C), Last_1(i)) = E_1(Last_1(C), x_\$) \leq E_1(L_2(C), L_2(i)) \oplus E_1(L_2(i), x_\$)$$
$$= E_1(L_2(C), L_2(i)) \oplus (0, \leq)$$
$$= E_1(L_2(C), L_2(i))$$

□

Corollary 31. *For every $\alpha_1 \dots \alpha_n \in [\omega]$ it holds for $\omega' = \alpha_1\$\alpha_2\$ \dots \$\alpha_n$ that $\omega' \lesssim_C \omega$, $\omega' \lesssim_I \omega$, $\omega' \lesssim_{IN} \omega$.*

Interpreted in terms of "convex hulls", consider $Z_{\alpha_1\$\alpha_2\$\dots\$\alpha_n}$ as a classical zone and observe that for each interleaving of a trace we obtain that the classical zone is "included" in the event zone with respect to $\lesssim_{EZ}$: $Z_{\alpha_1\$\alpha_2\$\dots\$\alpha_n} \lesssim_{EZ} Z_\omega$. This means that the "convex hull" of the classical zones of all interleavings is included in the event zone, and of course that this inclusion also holds on the level of $\lesssim_C$ and $\lesssim_{IN}$.

We can conclude that the convex hull closure is exact (and not an overapproximation) when applied to zones reached by trace equivalent interleavings only. However, in the event zone automaton, a single interleaving already yields this convex hull! It is therefore to be expected that our construction will work well where the convex hull overapproximation works well, but without possible error.

8 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 states (where feasable for the prototype) and did not compare execution times. Also, since we do not include static analysis improvements for less clock zones, comparison with state numbers obtained by the last version of for example UppAal is not meaningful. We chose to compare the two modi of the same base implementation. Where there are gains, they should be complementary to gains by better static analysis.

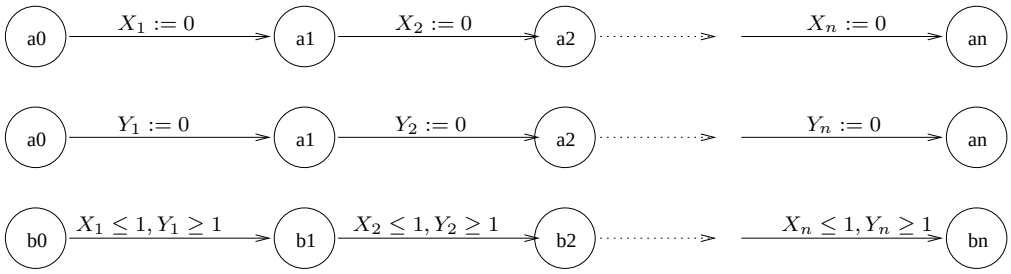


Fig. 6. The diamond example with 2n clocks

We consider three examples The first – artificial – example is the *diamond example* of Figure 6: Two automata just reset clocks in a fixed order and when both are done, an observer tests some properties of the interleavings. The clock zone automaton has just one maximal run (trace), with a quadratic number of prefixes. Clock zone automata however have to distinguish all possible shuffles of the resets of clocks X_i and Y_j. 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 [AL94] is a very unfavourable 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, even if just a modest one.

The experimental results are summarized in Figure 7, 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).

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	198	711	2596	9607	35923	135407	–	–	–	–
Philosophers, EZC	13	48	153	478	1507	4791	15369	49662	161393	–
Philosophers, CZ	66	393	2772	23103	223052	–	–	–	–	–
Fischer, EZC	48	887	17672	380632	–	–	–	–	–	–
Fischer, CZ	49	919	18751	417249	–	–	–	–	–	–

Fig. 7. Experimental results

9 Conclusions and Future Work

We have established a novel formal framework for partial order reductions of timed automata and developed a new kind of finite symbolic state automata based on event zones.

A particular difficulty in the partial order setting is to obtain a finite automaton. Technically different from zone widening approaches which exploit bisimulation properties of individual states, we use an equivalence and preorder relation on symbolic states without applying any widening. The event zones are thus

always exact, but we chop certain states of the event zone automaton without loss of reachable states, but resulting in a finite sub automaton.

We have implemented this approach in a prototype, the ELSE tool. While we have not yet been able to do experiments allowing a conclusive evaluation, we have designed academic examples in which our approach results in exponential savings compared to standard timed automata approaches.

On the theoretical side we are working on the integration of urgency into our framework, which will make it applicable to arbitrary Alur-Dill automata.

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 for his helpful constructive critique.

References

[AD94] R. Alur and D. Dill, *A theory of timed automata*, Theoretical Computer Science **126(2)** (1994), 183–235.

[AL94] Martín Abadi and Leslie Lamport, *An old-fashioned recipe for real time*, ACM Transactions on Programming Languages and Systems **16** (1994), no. 5, 1543–1571.

[BJLY98] J. Bengtsson, B. Jonsson, J. Lilius, and W. Yi, *Partial order reductions for timed systems*, Proceedings, Ninth International Conference on Concurrency Theory, Lecture Notes in Computer Science, vol. 1466, Springer-Verlag, 1998, pp. 485–500.

[BLP+99] G. Behrmann, K. Larsen, J. Pearson, C. Weise, W. Yi, and J. Lind-Nielsen, *Efficient timed reachability analysis using clock difference diagrams*, International Conference on Computer Aided Verification (CAV), Lecture Notes in Computer Science, vol. 1633, 1999, pp. 341–353.

[Bou02] P. Bouyer, *Timed automata may cause some troubles*, Tech. report, LSV, July 2002.

[CLR90] Th. Cormen, Ch. Leiserson, and R. Rivest, *Introduction to algorithms*, MIT Press, 1990.

[DGKK98] D. Dams, R. Gerth, B. Knaack, and R. Kuiper, *Partial-order reduction techniques for real-time model checking*, Formal Methods for Industrial Critical Systems (Amsterdam), no. 10, May 1998, pp. 469–482.

[DR95] V. Diekert and G. Rozenberg (eds.), *The book of traces*, World Scientific, 1995.

[DT98] D. D'Souza and P.S. Thiagarajan, *Distributed interval automata: A subclass of timed automata*, 1998, Internal Report TCS-98-3.

[DY96] C. Daws and S. Yovine, *Reducing the number of clock variables of timed automata*, IEE Real-Time Systems Symposium, December 1996, pp. 73–81.

[God96] 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-Verlag Inc., New York, NY, USA, 1996.

[LPY95] K. Larsen, P. Pettersson, and W. Yi, *Model-checking for real-time systems*, Fundamentals of Computation Theory, Lecture Notes in Computer Science, August 1995, Invited talk, pp. 62–88.

[Min99] Marius Minea, *Partial order reduction for verification of timed systems*, Ph.D. thesis, Carnegie Mellon University, 1999.

[NHZL01] P. Niebert, M. Huhn, S. Zennou, and D. Lugiez, *Local first search – a new paradigm in partial order reductions*, International Conference on Concurrency Theory (CONCUR), LNCS, no. 2154, 2001, pp. 396–410.

[NMA+02] P. Niebert, M. Mahfoudh, E. Asarin, M. Bozga, N. Jain, and O. Maler, *Verification of timed automata via satisfiability checking*, Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS, vol. 2469, 2002, pp. 225–244.

[NSY92] X. Nicollin, J. Sifakis, and S. Yovine, *Compiling real-time specifications into extended automata*, IEE Transactions on Software Engineering, vol. 18, September 1992, pp. 794–804.

[Pel93] D. Peled, *All from one, one for all: On model checking using representatives*, International Conference on Computer Aided Verification (CAV), Lecture Notes in Computer Science, vol. 697, 1993, pp. 409–423.

[Val89] A. Valmari, *Stubborn sets for reduced state space generation*, 10th International Conference on Application and Theory of Petri Nets, vol. 2, 1989, pp. 1–22.

[YS97] Tomohiro Yoneda and Bernd-Holger Schlingloff, *Efficient verification of parallel real-time systems*, Formal Methods in System Design **11** (1997), no. 2, 197–215.