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

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

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. 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, the 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, thus fully avoiding state splitting, yet avoiding problems of previous similar approaches with "maximal bounds abstraction". We achieve this goal by (1) lifting the theory of Mazurkiewicz traces to timed words and symbolic state exploration, (2) examining symbolic path exploration from a formal language point of view, and (3) by splitting the concerns of (abstraction free) successor computation and zone comparison by a new abstraction related to maximal bounds. The theory results in data structures and algorithms that we have experimentally validated, finding good reductions.

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.

Among the approaches for improve the efficiency is the transfer of "partial order reduction methods" [God96] from the discret setting (where they are known to give good reductions) to the timed setting. Partial order methods basically

K. Jensen and A. Podelski (Eds.): TACAS 2004, LNCS 2988, pp. 296–311, 2004.

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 *incomparable* states.

Previous works nevertheless trying to transfer partial order methods to the timed automata setting [DGKK98,BJLY98,Min99] have have tried to overcome this problem, 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. [BJLY98] therefore restrict 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.

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 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.

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 Myhill-Nerode right congruence (as well as a corresponding preorder 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 Myhill-Nerode congruence, and attempts to avoid incomparable states (by better abstractions, etc.) aim to get closer to the actual Myhill-Nerode congruence. For the framework with commutation, the Myhill-Nerode congruence is typically of infinite index (see Proposition 8), whereas the classical interleaving approaches prove its finiteness for the interleaving case.

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 Myhill-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 with maximal bounds abstra ction* 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 surprising fact 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) 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.

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 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. Due to lack of space, all proofs had to be omitted. A long version with all proofs is available online as technical report [LNZ04].

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 $w = a_1 \ldots a_n$ 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 $(a_1, t_1), (a_2, t_2) \ldots$ 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 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 finite 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 $(1 \leq m \leq n)$). We define $last_C(\omega) = 0$ if no such position exists (i.e. we assume that all clocks are reset at time position 0).

Definition 1. *A timed word (w, t) is a* normal realisation *of a clocked word $\omega = \alpha_1 \ldots \alpha_m$ 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 $i \in \{1, \ldots, m\}$, (ii) (w, t) is normal, and (iii) for all prefixes $\alpha_1 \ldots \alpha_{k-1}$ and all clocks C with $l = last_C(\alpha_1 \ldots \alpha_{k-1})$, $t(k) - t(l) \in c_k(C)$, i.e. the time elapsed since the last reset of clock C before position k meets the interval constraint at position k.*

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 $\alpha\gamma\beta \in L_N(\mathcal{A})$.

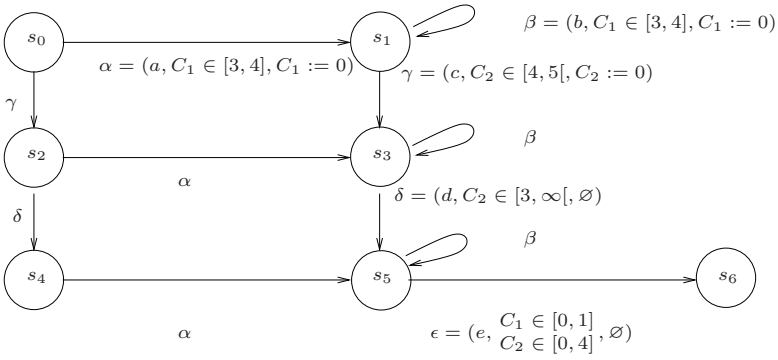


Fig. 1. A timed automaton

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]

The theoretical foundation of many partial order reduction approaches relies on the fact that the languages of asynchronous automata are closed with respect to equivalent words.Intuitively, two words are equivalent with respect to $\simeq_M$ iff they can be obtained from each other by repeatedly exchanging adjacent independent letters, as stated by the following lemma:

Lemma 3. *Let (Σ, I_Σ) be a partially commutative alphabet and $a_1 \dots a_n \simeq_M b_1 \dots b_n$ be two equivalent words. There exists a uniquely determined permutation $\pi : \{1, \dots, n\} \to \{1, \dots, n\}$, 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 $\pi(0) = 0$.*

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').*

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 (Δ, I_Δ). A timed word (w,t) I_Δ-*realises *ω iff (i) (*same length and actions*) $|\omega| = |w|$, 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 $\alpha_1 \dots \alpha_n$.*

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 straightforward 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 $\omega' \in L_N(\mathcal{A})$.*

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 we need more sophisticated methods to solve this 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-Myhill-Nerode right-congruence which yields the minimal automaton accepting the language (the states are the equivalence classes of the congruence)[1]. In our case the relevant congruence would be [2] $\omega_1 \simeq_I \omega_2$ `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 $\omega = \beta^i\gamma$ make $\alpha^i\beta^i\gamma$ I-realisable whereas $\alpha^j\beta^i\gamma$ is not I-realisable. □

This ruins our primary goal and explains why we 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.*
- *$\omega_1 \lesssim_{IN} \omega_2$ 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 $\lesssim_{EZ} \subseteq \lesssim_I$.*

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

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

– $\lesssim_C$ *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

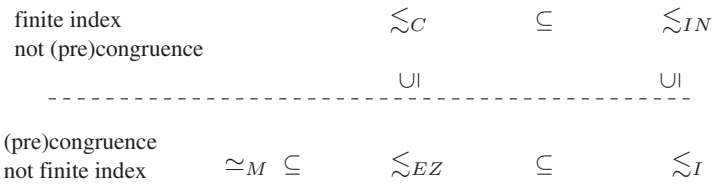


Fig. 2. Right preorders for clocked traces

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 17) 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 that 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 19 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 will show how to decide emptiness of the accepted language.

Difference Constraint Sets. 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). We represent a different constraint by a graph (the incidence matrix of which is a Difference Bounds Matrix, DBM): the variables are the vertices 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. A solution is a valuation from the set of variables to $\mathbb{R}$ which satisfies all the constraints. Since constraints are differences there is a solution iff there is a positive solution (all variables are ≥ 0). A difference constraint set is consistent iff it has one solution. Figure 3 gives a difference constraint set corresponding to the clocked word $\alpha\beta$ of the timed automaton of Figure 1.

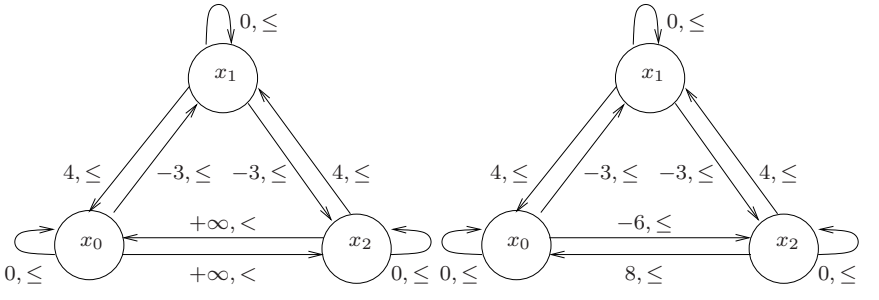
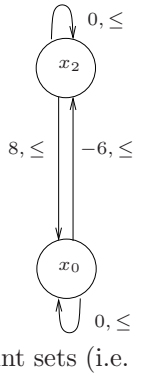


Fig. 3. A difference constraint set (left) and its closure (right).

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. The *closure* of a difference constraint set (V, E) is the difference constraint set $(V, E') = Cl(V, E)$ such that $E'(x, y) = min\{E(x_1, x_2) \oplus \ldots \oplus E(x_{p-1}, x_p) \mid x = x_1, \ldots, x_p = y \in V\}$, 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 (right part).



The closure can be computed by an all pairs shortest path algorithm such as Floyd-Warshall [CLR90]. The *projection* $\Pi_{V'}$ of (V, E) on $V' \subseteq V$ 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 graph of Figure 3. Projection is normally only a sensible operation on closed constraint sets (i.e. such that $Cl(V, E) = (V, E)$).

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 arrows 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 $lower([c_1, c_2)) = (-c_1, \leq)$ (note the $-$ sign)

Definition 10 (event zone). *The* event zone Z_ω *associated to a clocked word* ω *is a triple* $(V_\omega, E_\omega, Last_\omega)$ *where* $V_\omega = \{x_0, x_1, \ldots, x_{|\omega|}\}$, E_ω *is defined by* $E_\omega(x_i, x_j) = min\{(m, \prec) \mid x_i - x_j \prec m \in A_\omega\}$ *for all* $x_i, x_j \in V$ *with* A_ω *the following set of constraints:*

Respect clock constraints: *for* k, l *with* $l = last_C(\alpha_1 \ldots \alpha_{k-1})$ *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* $l = last_i(a_1 \ldots a_{k-1})$ *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* $x_i - x_j < +\infty \in A_\omega$.

and $Last_\omega : \mathcal{C} \cup Comp \to V_\omega$ *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* $j = last_i(\omega)$.

The difference constraint set associated to Z_ω is $S_\omega = (V_\omega, E_\omega)$. The closure of the 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 $\Pi_{V'}(Z_\omega) = (\Pi_{V'}(V_\omega, E_\omega), Last_\omega)$. A zone Z_ω is *consistent* iff its associated difference constraint set S_ω is consistent. By construction a zone Z_ω is consistent iff ω is I-realisable.

Definition 11 (event zone precongruence). *Let* ω_1, ω_2 *be two clocked words over* Δ *and* $Cl(Z_{\omega_1}) = (V_1, E_1, Last_1), Cl(Z_{\omega_2}) = (V_2, E_2, Last_2)$ *be the closure of their respective event zones. 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* $\xi_1, \xi_2 \in \mathcal{C} \cup Comp$, $E_1(Last_1(\xi_1), Last_1(\xi_2)) \leq E_2(Last_2(\xi_1), Last_2(\xi_2))$.

We define $\omega_1 \lesssim_{EZ} \omega_2$ iff $Z_{\omega_1} \lesssim_{EZ} Z_{\omega_2}$ and we get the following properties:

Proposition 12. *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* $\omega_1 \lesssim_I \omega_2$.

The Event Zone Automaton. For the construction of an automaton we define the notion of zone extension.

Definition 13. *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 $x_i \in V$, $x_i - x_{n+1} \leq +\infty, x_{n+1} - x_i \leq +\infty \in A_{\omega\odot\alpha}$,
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* $Last'(\xi) = Last(\xi)$.

By definition, we get $Z_\omega \odot \alpha \simeq_{EZ} Z_{\omega\alpha}$. Event zones have an unbounded number of variables but only the variables representing the last occurrences of events are relevant. 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). This projection behaves well with respect to extension and $\lesssim_{EZ}$:

Proposition 14. *Let* Z *be a consistent event zone and* α *be a clocked label. Then* $last(Z \odot \alpha) \simeq_{EZ} last(last(Z) \odot \alpha)$.

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 15 (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* $s'_0 = (s_0, [Z_0])$, *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 16. *The event zone automaton for an asynchronous timed automaton is an asynchronous timed automaton accepting exactly the clocked words having an* I*-realisation.*

6 Catchup Preorder for Language Emptiness Checking

In this section we introduce the *catchup preorder*, closely related to the maximal bounds abstraction used in classical timed automata algorithms and a very important aspect of our approach. Based on it, we then give an algorithm to decide the emptiness of timed automata languages.

Catchup preorder and equivalence. First we introduce a useful technical tool: the separator action \$. A separator \$ is an element of Δ such that the constraints are trivial (no conditions on any clocks), the reset set is empty, and for all $\alpha \in \Delta$ it holds that $\alpha \not I \$$. Any clocked alphabet Δ can be extended to a clocked alphabet Δ' containing a separator (either there is one already in Δ or we simply add one). This extension preserves the semantics: if $\omega_1 \lesssim_I \omega_2$ in Δ', then $\omega_1 \lesssim_I \omega_2$ in Δ. From now on, we assume that Δ contains such a separator \$. All previous results holds independently of the existence of \$, but it is used in the proof of the next proposition:

Proposition 17. *If Δ contains a separator \$, then $\lesssim_I \subseteq \lesssim_{IN}$.*

Definition 18 (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 $\xi_1 \in \mathcal{C}$, $\xi_2 \in \mathcal{C} \cup \{1\}$ ($1 \in Comp$[3]):*

- $E_1(Last_1(\xi_1), Last_1(\xi_2)) \leq E_2(Last_2(\xi_1), Last_2(\xi_2))$*; or*
- $E_1(Last_1(\xi_1), Last_1(1))$, $E_2(Last_2(\xi_1), Last_2(1))$ *(constraint between clock reset events and the separator) are both strictly smaller than $(-c, \leq)$ for the greatest non-trivial upper bound $(c, \leq)$ for ξ_1 in Δ (upper catchup); or*
- *both $E_1(Last_1(\xi_1), Last_1(\xi_2))$, $E_2(Last_2(\xi_1), Last_2(\xi_2))$ greater or equal to the opposite of the biggest lower bound for ξ_2 in Δ (lower catchup).*

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 is that we abstract event zone extensions that occur in the past of already present event (e.g. events that would have occurred before the separator in the second rule). We consider such events as *late* and catching up. The second rule addresses bounds relevant to upper bounds of clocks (*upper catchup*), the third rule addresses bounds relevant to lower bounds (*lower catchup*).

Theorem 19. *$\lesssim_C \subseteq \lesssim_{IN}$ and the index of $\simeq_C$ is smaller than $(4K+3)^{n(n+1)}$ where n is the number of clocks and K is the biggest constant mentioned in constraints.*

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

An algorithm to decide the emptiness of $L_I(\mathcal{A})$. The description of the algorithm uses traces for readability, but the actual implementation relies on event zones. The set of traces is partionned into *white traces* that are not visited yet, *gray traces* that await exploration, *black traces* that have been explored, and *red traces* that have been rejected because of catchup equivalence. This last set is convenient for the correctness proof only and is useless in the implementation. The algorithm is generic and doesn't rely on the particular method used to explore traces. The key point for ensuring termination is the finite index of the relation $\lesssim_C$. The actual implementation uses several technical improvments that we do not describe because of the lack of space.

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']}
    end if
  end for
end while
return "empty"
```

Theorem 20. *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".*

Comparison with clock zone automata

The zone automaton. If $I = \emptyset$, we are back to classical timed automata and $\lesssim_N = \lesssim_{IN} = \lesssim_I$. Zones and the relation $\lesssim_Z$ are the same and we can modify the algorithm to get a finite deterministic automaton for $L_N(\mathcal{A})$.

Relation of $\lesssim_{IN}$ and the convex hull overapproximation. UppAal has an option to join incomparable clock zones Z_1, Z_2 into a so-called convex hull, the least zone Z, such that containing both $Z_1, Z_2 \lesssim_N Z$. For the exploration algorithm this means replacing two state zone pairs (s, Z_i) into a single pair (s, Z). This is an overapproximation (additional states may become reachable), yet it can be used to prove language emptiness.

Given a word $\alpha_1\alpha_2 \ldots \alpha_n$, the corresponding classical zone is $Z_{\omega_\$}$ for $\omega_\$ = \alpha_1\$\alpha_2\$ \ldots \$\alpha_n$ (the separator forbids any interleaving). We can prove that for any $\alpha_1\alpha_2 \ldots \alpha_n \in [\omega]$, if $\omega_\$ = \alpha_1\$\alpha_2\$ \ldots \$\alpha_n$ then $\omega_\$ \lesssim_I \omega$, $\omega_\$ \lesssim_{IN} \omega$ and $\omega_\$ \lesssim_C \omega$. This means that all classical zones corresponding to interleavings of the same word are included in the same event zone, i.e. the convex hull approximation is

exact when applied to zones reached by equivalent interleavings only and in fact a single interleaving in our semantics already yields this convex hull!

Experiments. For practical evaluation, we have built a tool, *ELSE*, 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, absolute comparisons with existing tools like UppAal seem not meaningful at this stage. We chose to compare the two modi of the same base implementation to estimate the potential of passing from classical semantics to event zones and catchup preorder.

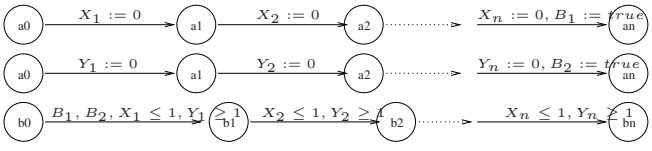


Fig. 4. The diamond example with 2n clocks

We consider three examples. The first – artificial – example is the *diamond example* of Figure 4: 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 may yield a fair reduction. The experimental results are summarized in Figure 5, where "EZC" stands for exploration with event zone automata and catchup preorder whereas "CZ" stands for clock zone automata. Each case concerns scalable examples with a parameter m (number of clock of each process in the diamond example, number of philosophers, number of processes Fischer protocol).

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. This work was supported

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

Fig. 5. Experimental results

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

References

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

[AL94] M. Abadi and L.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, LNCS, vol. 1466, Springer-Verlag, 1998, pp. 485–500.

[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.

[God96] P. Godefroid, *Partial-order methods for the verification of concurrent systems: an approach to the state-explosion problem*, LNCS, vol. 1032, Springer-Verlag Inc., New York, NY, USA, 1996.

[LNZ04] D. Lugiez, P. Niebert, and S. Zennou, *A Partial Order Semantics Approach to the Clock Explosion Problem of Timed Automata*, Rapport de Recherche, Laboratoire d'Informatique Fondamentale de Marseille, January 2004, available from `http://www.lif.univ-mrs.fr/Rapports`.

[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.

[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.