Partial Order Reduction for Model Checking of Timed Automata *

Marius Minea

Carnegie Mellon University, School of Computer Science
Pittsburgh, PA 15213-3891
marius+@cs.cmu.edu

Abstract. The paper presents a partial order reduction method applicable to networks of timed automata. The advantage of the method is that it reduces both the number of explored control states and the number of generated time zones. The approach is based on a local-time semantics for networks of timed automata defined by Bengtsson et al. [1998], and used originally for local reachability analysis. In this semantics, each component automaton executes asynchronously, in its own local time scale, which is tracked by an auxiliary reference clock. On communication transitions, the automata synchronize their time scales. We show how this model can be used to perform model checking for an extension of linear temporal logic, which can express timing relations between events. We also show how for a class of timed automata, the local-time model can be implemented using difference bound matrices without any space penalty, despite the need to represent local time. Furthermore, we analyze the dependence relation between transitions in the new model and give practical conditions for selecting a reduced set of transitions.

1 Introduction

Model checking [5] has emerged as a very successful automatic verification technique for finite-state systems. However, its application is still limited by the *state space explosion* problem. The number of possible states in a system grows exponentially with the number of component parts, quickly exceeding the current capabilities of verification tools. For timed systems, the complexity in the control space is increased by the timing information that needs to be maintained, since each untimed state can be reached at many different time instances.

Partial order reduction [8,14,15] is a well-established method to reduce the complexity of state space exploration in asynchronous systems. It explores a restricted number of interleavings for independent concurrent transitions, while preserving the verified property in the reduced model. However, in timed systems the implicit synchronization among transitions, caused by the passage of time, makes the application of this technique problematic. This paper shows how to perform partial order reduction for continuous-time systems modeled as timed

* This research was sponsored in part by the Semiconductor Research Corporation (SRC), the National Science Foundation (NSF), and the Defense Advanced Research Projects Agency (DARPA).

Jos C.M. Baeten, Sjouke Mauw (Eds.): CONCUR'99, LNCS 1664, pp. 431–446, 1999.

automata, while preserving properties specified in an extension of linear-time temporal logic augmented with explicit time constraints.

2 Timed Automata

2.1 Definition

Timed automata [1] are transition systems extended with real-valued clocks which advance at the same rate and can be reset on executing a transition. Both states and transitions are associated with temporal constraints on the clocks.

Definition 1. *A* clock *is a variable over the set* $\mathbb{R}^+$ *of nonnegative reals. A* clock valuation *for a set of clocks* $C = \{x_1, \cdots, x_n\}$ *is a function* $v : C \to \mathbb{R}^+$.

Definition 2. *An* atomic clock constraint *is an inequality of the form* $x \prec c$, $c \prec x$, *or* $x - y \prec c$, *where* x, y *are clocks,* $c \in \mathbb{Z}$ *is an integer and* $\prec \in \{<, \leq\}$. *A clock constraint is a conjunction of atomic clock constraints or the value* true. *The set of clock constraints over a set of clocks* C *is denoted by* $\mathcal{B}(C)$.

Definition 3. *A timed automaton is a tuple* $A = (S, S^0, C, E, I, \mu)$, *where*
- S *is a finite set of nodes (control states);* $S^0 \subseteq S$ *is the set of initial nodes*
- C *is a finite set of real-valued non-negative clocks*
- $E \subseteq S \times \mathcal{B}(C) \times 2^C \times S$ *is a finite set of edges. An edge* $e = \langle s, \psi, R, s' \rangle$ *has an* enabling condition ψ *and a set* R *of clocks that are reset on traversing the edge.*
- $I : S \to \mathcal{B}(C)$ *defines an* invariant condition *associated with each node*
- $\mu : S \to 2^P$ *labels each node with atomic propositions from a set* P

A satisfied enabling condition does not force the execution of a transition. An automaton can remain at the same node as long as the node invariant is satisfied.

We define a network of timed automata using a general parallel composition:

Definition 4. *Consider* n *timed automata* $A_i = (S_i, S_i^0, C_i, E_i, I_i, \mu_i)$, *and a synchronization function* $f : \prod_{i=1}^n (E_i \cup \{\epsilon\}) \to \{0, 1\}$ *(where* ϵ *is a symbol denoting a null edge). The network of timed automata* $A_1 \parallel A_2 \parallel \ldots \parallel A_n$ *is a timed automaton* $A = (S, S^0, C, E, I, \mu)$, *where:*
- $S = S_1 \times S_2 \times \ldots \times S_n$ *and* $S^0 = S_1^0 \times S_2^0 \times \ldots \times S_n^0$
- $C = C_1 \cup C_2 \cup \ldots \cup C_n$ *(assuming* $C_i \cap C_j = \emptyset$, *for* $i \neq j$*)*
- E *contains a family of edges (a transition) for each tuple with* $f(e_1, \cdots, e_n) = 1$. *For transition* a, *let* $e_i = \langle s_i, \psi_i, R_i, s_i' \rangle$ *if* $e_i \neq \epsilon$ *and* $active(a) = \{i \mid e_i \neq \epsilon\}$. *The edges of* a *have endpoints with* s_i *and* s_i' *given by* e_i *for* $i \in active(a)$, $s_j = s_j' \in S_j$ *arbitrary for* $j \notin active(a)$, $\psi = \bigwedge_{i \in active(a)} \psi_i$, *and* $R = \bigcup_{i \in active(a)} R_i$.
- $I(s) = \bigwedge_{i=1}^n I_i(s_i)$
- $\mu(s) = \bigcup_{i=1}^n \mu(s_i)$ *(assuming pairwise disjoint sets of atomic propositions* P_i*)*

A transition corresponds to the synchronous traversal of edges in several component automata. The synchronization function determines which automata execute (the *active set* of the transition) and which ones remain at their local state. This allows the modeling of many common synchronization paradigms, including pairwise communication. A transition with more than one automaton in its active set is called a *synchronization transition*, otherwise it is called *local*.

2.2 Semantics

Given a clock valuation v and $d \in \mathbb{R}^+$, $v+d$ is the valuation given by $(v+d)(x) = v(x) + d$, $\forall x \in C$. For $R \subseteq C$, $v[R \mapsto 0]$ is the clock valuation that is zero for clocks in R and agrees with v for all other clocks. The truth value of the clock constraint $\psi \in \mathcal{B}(C)$ for a clock valuation v is denoted by $\psi(v)$.

Definition 5. *A model of a timed automaton is a state-transition graph* $\mathcal{S}(A) = (\Sigma, \Sigma^0, \rightarrow)$, *where*
- $\Sigma = \{(s, v) \mid I(s)(v)\}$ *is the set of timed states satisfying the node invariant*
- $\Sigma^0 = \{(s^0, 0_C) \mid s^0 \in S^0\}$ *is the set of initial states, with* $0_C(x) = 0$, $\forall x \in C$
- $\rightarrow$ *is the transition relation defined as union of* delay *and* action *transitions:*
  - $(s, v) \overset{d}{\rightsquigarrow} (s, v + d)$ *if* $d \in \mathbb{R}^+$, *and for all* $0 \leq d' \leq d$, $I(s)(v + d')$ *holds*
  - $(s, v) \overset{a}{\rightarrow} (s', v[R \mapsto 0])$ *for* $a \in \mathcal{T}$ *(the set of transitions of A) if there exists an edge* $e = (s, \psi, R, s') \in a$, *such that* $\psi(v)$ *is true and* $I(s')(v[R \mapsto 0])$ *holds*

A *delay transition* models the elapse of time in the same control state, while maintaining the invariant. An *action transition* can be executed (instantaneously) if the clock valuation satisfies the enabling condition. Clocks in the set R are reset, the other clocks maintain their value.

We assume that node invariants contain only constraints of the form $x_i \prec c$, because the constraints $x_i - x_j \prec c$ or $c \prec x_i$ are not falsified by time passage, and can be incorporated into the enabling condition of incoming edges. Also, since clock constraints are convex, invariants must only be checked in the final state of a delay transition: $(s, v) \overset{d}{\rightsquigarrow} (s, v + d)$ if $d \in \mathbb{R}^+$ and $I(s)(v + d)$ holds.

Definition 6. *An* execution trace *of a timed automaton is a finite or infinite sequence* $\sigma = (s^0, 0_C) \rightarrow (s^1, v^1) \ldots \rightarrow (s^k, v^k) \ldots$ *starting from a state* $s^0 \in S^0$.

We denote by $\sigma(k) = (s^k, v^k)$ the k^{th} state on the trace σ, by σ_k the finite prefix of σ ending at (s^k, v^k) and by σ^k the suffix of σ starting at the same state.

2.3 The Model Checking Problem

Several model checkers for timed automata exist. The KRONOS tool is a model checker for TCTL and timed μ-calculus [9], and UPPAAL [11] verifies properties in a timed modal logic. However, partial order approaches have been so far restricted to less expressive properties: Pagani [12,13] performs deadlock detection, whereas Bengtsson et al. [2] check local reachability within one process.

We use an extension of LTL inspired from the timed temporal logic for nets (TNL) of [18], which has been used to verify time Petri nets. By allowing constraints on two clock differences, the logic permits reasoning about the time separation of two events, since the difference between two clocks corresponds to the difference between the execution times of the transitions that reset them.

The formulas of our logic, called LTL_Δ, are defined by the grammar:

$$\psi ::= \mathit{true} \mid p \mid x - y \prec c \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \,\mathcal{U}\, \varphi_2$$

where $p \in P$ is an atomic proposition, $x, y \in C$ are clocks, $c \in Z$ and $\prec \in \{<, \leq\}$.

Definition 7. *Consider an infinite execution trace* $\sigma = (s^0, v^0) \to (s^1, v^1) \to \ldots \to (s^k, v^k) \to \ldots$ *The semantics of an* LTL_Δ *formula is defined as follows:*
- $(s, v) \models p$ *iff* $p \in \mu(s)$
- $(s, v) \models x - y \prec c$ *iff* $v(x) - v(y) \prec c$.
- $\sigma \models \varphi_a$ *iff* φ_a *is an atomic formula and* $(s^0, v^0) \models \varphi_a$
- $\sigma \models \neg\varphi$ *iff* $\sigma \models \varphi$ *does not hold*
- $\sigma \models \varphi_1 \wedge \varphi_2$ *iff* $\sigma \models \varphi_1$ *and* $\sigma \models \varphi_2$
- $\sigma \models \varphi_1 \,\mathcal{U}\, \varphi_2$ *iff* $\exists k \geq 0$ *such that* $\sigma^k \models \varphi_2$ *and* $\sigma^j \models \varphi_1$ *for all* $0 \leq j < k$
- $\mathcal{S}(A) \models \varphi$ *iff* $\sigma \models \varphi$ *for any infinite execution trace* σ *of* $\mathcal{S}(A)$.

Since control state and clock differences are preserved by time passage, all intermediate states traversed by a delay transition have the same truth value for any atomic subformulas in LTL_Δ. Thus, the given semantics based on transition endpoints corresponds to the intuitive meaning of continuous execution.

3 The Model Checking Approach

3.1 Effect of Transition Interleavings

The traditional reachability analysis algorithm for networks of timed automata explores all possible transition interleavings among the individual components. Partial order methods choose a representative from each set of equivalent interleavings, exploring only a reduced portion of the state space. However, in our model of time, clocks advance simultaneously in all automata, and different interleavings may produce different assignments to clock values. The independence of transitions in the underlying untimed system may not be preserved.

Consider the system of two automata in Fig. 1 and its exploration using timed zones [9]. From the initial state $\langle (s_1, s_2), x = y \rangle$, transition a leads to the state $\langle (s'_1, s_2), x \leq y \rangle$ (since clock x is reset). Next, on executing b, clock y is reset, leading to state $\langle (s'_1, s'_2), x \geq y \rangle$. If b is executed before a, the system reaches first the state $\langle (s_1, s'_2), x \geq y \rangle$, and then the state $\langle (s'_1, s'_2), x \leq y \rangle$.

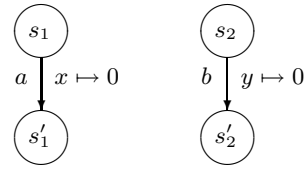

Fig. 1. Effect of transition interleavings

The two interleavings lead to the same control state, but to distinct clock zones and thus distinct states in the zone automaton. Hence, the transitions are not independent and usual partial order reduction techniques cannot be applied.

For a property insensitive to the ordering of x and y, both interleavings are still equivalent, leading to a timed state in the union of the two zones, $\langle (s'_1, s'_2), x \geq y \vee x \leq y \rangle = \langle (s'_1, s'_2), \mathit{true} \rangle$. Our goal is a partial order reduction method that produces a zone containing the timed states reachable by all transition interleavings, while exploring only one interleaving, and thus fewer states.

3.2 Related Work

Partial order reduction has been investigated by Yoneda and Schlingloff [18] for time Petri nets, which have earliest and latest firing times associated with transitions and are thus less expressive than timed automata. The logic used for specifications is similar to LTL_{Δ}, but the dependency relation between transitions uses run-time information about the time component of the current state. Lilius [10] improves on this technique by not storing the transition firing order in the timing constraints and reducing branching in the generated graph.

For timed automata, Pagani [12,13] shows that in many cases timing introduces dependencies and reduces the amount of partial order reduction. The analysis is limited to deadlock detection. Dams et al. [6] handle some of these cases, generalizing the notion of independence and selecting at a state those transitions whose executions cover the result of exploring other interleavings.

Belluomini and Myers [3] use an event model with lower and upper time bounds associated to transitions. Timing information is represented in the form of partially ordered sets, reducing the number of generated time zones. However, their analysis does not reduce the number of explored transition interleavings.

The method from which we draw most is that of Bengtsson, Jonsson, Lilius and Wang [2]. They define a local-time semantics based on desynchronized execution of the component automata and local time delays, with additional reference clocks to model synchronization. In this model the same independence conditions as in the untimed case apply, and an algorithm is given to decide the reachability of a local control state.

3.3 Local-Time Model

We revisit the local-time model of Bengtsson et al. [2] using somewhat different notations and prove several results underlying its use in model checking.

Consider the interaction of action and delay transitions. The enabling of an action transition and the resulting state change depend only on the state of the participating automata. Hence, two action transitions with disjoint active sets are independent. On the other hand, a delay transition changes the state in *all* automata by incrementing the values of all clocks. It is therefore dependent on any action transition that also changes clock values (specifically, resets clocks).

However, one can view a global delay transition as a set of simultaneous transitions with equal delay in all component automata. This suggests that time-induced dependencies can be removed by separating a global delay transition into individual transitions for each component automaton, without requiring their simultaneity. To this effect, local passage of time is introduced as follows:

For a clock valuation v, $d \in \mathbb{R}$ and $i \in \overline{1,n}$, define the clock valuation $v +_i d$ by: $(v +_i d)(x) = v(x) + d$ for $x \in C_i$ and $(v +_i d)(x) = v(x)$ otherwise.

A *local delay transition* $\overset{d}{\rightsquigarrow}_i$ increments only the clocks in automaton A_i. We identify it with a pair $(d,i) \in \mathcal{T}_{\Delta} = \mathbb{R}^+ \times \overline{1,n}$, define $active(\overset{d}{\rightsquigarrow}_i) = \{i\}$ and denote $\mathcal{T}_l = \mathcal{T} \cup \mathcal{T}_{\Delta}$. For $i \in \overline{1,n}$, define the functions $delay_i : \mathcal{T}_l \mapsto \mathbb{R}^+$ as

follows: $delay_i(\overset{d}{\rightsquigarrow}_i) = d$, $delay_i(\overset{d}{\rightsquigarrow}_j) = 0$ for $i \neq j$, and $delay_i(\overset{a}{\rightarrow}) = 0$ for $a \in \mathcal{T}$. They indicate the delay caused by a transition in a component automaton.

Definition 8. *The local-time model $\mathcal{L}(A)$ for a network of timed automata $A = A_1 \parallel A_2 \parallel \ldots \parallel A_n$ is a state-transition graph with state set Σ, initial state set Σ^0 and execution traces $\sigma = (s^0, v^0) \overset{\tau_1}{\rightarrow} (s^1, v^1) \ldots \overset{\tau_k}{\rightarrow} (s^k, v^k) \ldots$ starting from a state $(s^0, v^0) \in \Sigma^0$ and satisfying one of the following conditions for any $k \geq 1$:*
• *$\tau_k = (d, i) \in \mathcal{T}_\Delta$, $s^k = s^{k-1}$, $v^k = v^{k-1} +_i d$ and $\forall d' \in [0, d]. I_i(s_i^k)(v^k + d')$, or*
• *$\tau_k \in \mathcal{T}$, $(s^{k-1}, v^{k-1}) \overset{\tau_k}{\rightarrow} (s^k, v^k)$ and $\sum_{l=1}^{k-1} delay_i(\tau_l) = \sum_{l=1}^{k-1} delay_j(\tau_l)$ for all $i, j \in active(\tau_k)$*

The first case is a local delay transition $(s^{k-1}, v^{k-1}) \overset{d}{\rightsquigarrow}_i (s^k, v^k)$ in automaton A_i. In the second case, an action transition $(s^{k-1}, v^{k-1}) \overset{\tau_k}{\rightarrow} (s^k, v^k)$ is executed, under the additional constraint that the elapsed time (the sum of delays) is identical for all automata in the active set. (For a local action transition, this additional constraint is void). In both cases, the transition τ_k is said to be *enabled* after the execution of σ_{k-1}. Denote by $enabled(\sigma)$ and $enabled^*(\sigma)$ the set of transitions and transition sequences, respectively, that can follow a finite trace σ.

For a finite execution trace $\sigma = (s^0, v^0) \overset{\tau_1}{\rightarrow} (s^1, v^1) \ldots \overset{\tau_k}{\rightarrow} (s, v)$, define $time_i(\sigma) = t_0 + \sum_{l=1}^{k} delay_i(\tau_l)$, where $t_0 \in \mathbb{R}^+$ is an arbitrary timepoint at which the execution of σ starts. Then, $time_i(\sigma)$ (or $time_i$, when σ is implicit) denotes the timepoint reached in A_i after executing σ. The *local configuration* of A_i reached by σ is the tuple $cfg_i(\sigma) = (s_i, v_i, time_i)$, where v_i denotes the restriction of v to the clocks of A_i. The *global configuration* of A is the tuple $cfg(\sigma) = (cfg_1(\sigma), cfg_2(\sigma), \cdots, cfg_n(\sigma))$, also denoted $cfg(\sigma) = (s, v, time)$ with $time = (time_1, time_2, \cdots, time_n)$. The set of configurations is $\Sigma_C = \Sigma \times (\mathbb{R}^+)^n$.

Note that the enabling of an action transition is defined in terms of the trace executed so far. The following result shows that a configuration determines completely the subsequently enabled transitions. The proof follows directly from the definitions of parallel composition and the local-time model.

Proposition 1. *The following properties hold in the local-time model $\mathcal{L}(A)$ for finite execution traces σ and σ' and transition $\tau \in enabled(\sigma)$:*
• *if $cfg_i(\sigma) = cfg_i(\sigma')$ for all $i \in active(\tau)$, then $\tau \in enabled(\sigma')$ and $cfg_i(\sigma\tau) = cfg_i(\sigma'\tau)$ for all $i \in active(\tau)$*
• *$cfg_j(\sigma\tau) = cfg_j(\sigma)$ for all $j \notin active(\tau)$, where $\sigma\tau$ denotes the trace obtained by extending σ with the transition τ.*

Consequently, two finite execution traces leading to the same configuration have the same enabled transitions. For a configuration $\gamma \in \Sigma_C$ one can thus define $enabled(\gamma) = enabled(\sigma)$, where σ is an execution trace such that $cfg(\sigma) = \gamma$. Likewise, the successor configuration of γ by a transition $\tau \in enabled(\sigma)$ is defined as the configuration reached when extending the trace σ by transition τ: $succ_\tau(\gamma) = cfg(\sigma\tau)$. This is again independent of σ and we write $\gamma \overset{\tau}{\rightarrow} succ_\tau(\gamma)$.

We now prove the desired independence properties for transitions in $\mathcal{L}(A)$. In general, two transitions are called independent if neither disables the execution of the other, and the same state is reached by executing them in either order:

Definition 9. *Two transitions τ_1 and τ_2 are independent iff for any finite execution trace σ such that $\tau_1, \tau_2 \in enabled(\sigma)$ the following two conditions hold:*
• Enabledness: $\tau_2 \in enabled(\sigma\tau_1) \wedge \tau_1 \in enabled(\sigma\tau_2)$
• Commutativity: $fin(\sigma\tau_1\tau_2) = fin(\sigma\tau_2\tau_1) \wedge enabled^*(\sigma\tau_1\tau_2) = enabled^*(\sigma\tau_2\tau_1)$
where $fin(\sigma)$ denotes the last state on the trace σ.

Theorem 1. *Two (action or local delay) transitions $\tau_1, \tau_2 \in \mathcal{T}_l$ that involve disjoint sets of automata ($active(\tau_1) \cap active(\tau_2) = \emptyset$) are independent.*

Proof. For all $j \in active(\tau_2)$, we have $j \notin active(\tau_1)$, hence $cfg_j(\sigma\tau_1) = cfg_j(\sigma)$. Therefore, $\tau_2 \in enabled(\sigma) \Rightarrow \tau_2 \in enabled(\sigma\tau_1)$, and symetrically for τ_1. Also, since $active(\tau_1) \cap active(\tau_2) = \emptyset$, each local configuration is changed at most once, either by τ_1 or by τ_2, irrespective of their ordering. Therefore, $cfg(\sigma\tau_1\tau_2) = cfg(\sigma\tau_2\tau_1)$ and $fin(\sigma\tau_1\tau_2) = fin(\sigma\tau_2\tau_1)$. Since the enabled transitions are determined by the reached configuration, $enabled^*(\sigma\tau_1\tau_2) = enabled^*(\sigma\tau_2\tau_1)$. □

A finite trace σ in $\mathcal{L}(A)$ is called *synchronized* if $time_i(\sigma) = time_j(\sigma)$ for all $i, j \in \overline{1,n}$, i.e., if all automata have executed for the same amount of time, denoted by $time(\sigma)$. The following theorem relates the reachable state spaces of the standard and local-time models (cf. [2]):

Theorem 2. *Each state reachable in $\mathcal{S}(A)$ is also reachable in $\mathcal{L}(A)$. Moreover, each state reached by a synchronized trace in $\mathcal{L}(A)$ is also reachable in $\mathcal{S}(A)$.*

Proof. First, any trace in $\mathcal{S}(A)$ yields a trace in $\mathcal{L}(A)$ by replacing each global delay transition $\overset{d}{\rightsquigarrow}$ with the sequence of local delay transitions $\overset{d}{\rightsquigarrow}_1 \ldots \overset{d}{\rightsquigarrow}_n$.

The reverse implication follows by induction on the number of action transitions in the trace σ_l of $\mathcal{L}(A)$. For the base case, if σ_l is synchonized and contains only local delay transitions, they sum up to the same total delay d. Then, $fin(\sigma_l)$ is reachable in $\mathcal{S}(A)$ by executing the global delay transition $\overset{d}{\rightsquigarrow}$.

For the induction step, let a be the action transition in σ_l executed at the latest timepoint, $t_a \leq t = time(\sigma_l)$. In every automaton, σ_l ends with local delay transitions totaling at least $t - t_a$. Removing this delay in every automaton yields a synchronized trace σ'_l with $time(\sigma'_l) = t_a$. In σ'_l, a is the last transition in all participating automata. Its removal yields a synchronized execution trace σ''_l with fewer action transitions. By the induction hypothesis, $fin(\sigma''_l)$ is reachable in $\mathcal{S}(A)$, and $fin(\sigma_l)$ is reachable from it by executing $\overset{a}{\rightarrow}$ followed by $\overset{t-t_a}{\rightsquigarrow}$. □

3.4 Local-Time Zone Automaton

An analogue of the zone automaton [9], which represents sets of timed states using clock constraints can be derived for the local-time model [2]. A *local-time zone* is a convex set of configurations $z \in \Sigma_C$ with the same control state. A transition is enabled in a zone iff it is enabled in some configuration in the zone: $enabled(z) = \{\tau \in \mathcal{T}_l \mid \exists \gamma \in z.\, \tau \in enabled(\gamma)\}$. The successor of a zone z by a transition $\tau \in enabled(z)$ is $succ_\tau(z) = \{succ_\tau(\gamma) \mid \gamma \in z \wedge \tau \in enabled(\gamma)\}$.

For the standard zone automaton, an exploration step consists of an action transition followed by a delay transition of arbitrary amount. For the local-time model, we combine an action transition with subsequent delay transitions in all automata belonging to its active set, and show:

Proposition 2. *For any finite execution trace σ, there exists a trace σ' with the same final configuration, which starts with a local delay transition in each component automaton, after which every subsequent action transition is followed by local delay transitions in all participating automata.*

Proof. A delay transition $\overset{d}{\rightsquigarrow}_i$ commutes with any other delay transition, and with action transitions a for which $i \notin \mathit{active}(a)$. Thus, a delay transition can be moved towards the beginning of the execution trace σ (merging consecutive delay transitions in the same automaton) until the preceding action transition involves the same automaton, or there are no preceding action transitions. □

Based on this result, we define the zone successor operation as follows:

$$\mathit{succ}_l^Z(z,a) = \{\gamma_k \in \Sigma_C \mid \exists \gamma \in z, \exists d_{i_1}, \cdots, d_{i_k} \in \mathbb{R}^+ . \gamma \xrightarrow{a} \gamma' \overset{d_{i_1}}{\rightsquigarrow} \gamma_1 \ldots \overset{d_{i_k}}{\rightsquigarrow} \gamma_k\}$$

where $\mathit{active}(a) = \{i_1, i_2, \cdots, i_k\}$. An initial local-time zone is the set of all configurations reachable from an initial state by a sequence of delay transitions:

$$\mathit{init}_l^Z(s^0) = \{\mathit{cfg}(\sigma) \mid \exists d_{i_1}, \cdots, d_{i_n} \in \mathbb{R}^+ . \sigma = (s^0, 0_C) \overset{d_{i_1}}{\rightsquigarrow} (s^0, v^1) \ldots \overset{d_{i_n}}{\rightsquigarrow} (s^0, v^n)\}$$

If $\mathit{succ}_i^\Delta(z) = \{\gamma' \mid \exists \gamma \in z, \exists d \in \mathbb{R}^+ . \gamma \overset{d}{\rightsquigarrow}_i \gamma'\}$ is the successor by an arbitrary local delay, then $\mathit{init}_l^Z(s^0) = (\mathit{succ}_n^\Delta \circ \ldots \circ \mathit{succ}_1^\Delta)(\gamma^0(s^0))$ and $\mathit{succ}_l^Z(z,a) = (\mathit{succ}_{i_k}^\Delta \circ \ldots \circ \mathit{succ}_{i_1}^\Delta \circ \mathit{succ}_a)(z)$, where $\circ$ denotes function composition.

Definition 10. *The local-time zone automaton $\mathcal{Z}_l(A)$ for a network of automata A is a tuple $(Z_l, Z_l^0, \mathit{succ}_l^Z)$, with $Z_l^0 = \{\mathit{init}_l^Z(s^0) \mid s^0 \in S^0\}$ the set of initial local-time zones, succ_l^Z the successor relation defined above, and Z_l the set of local-time zones reachable by successive application of succ_l^Z from an initial zone.*

Together with Prop. 2, this definition implies directly the following:

Theorem 3. *A state is reachable in the model $\mathcal{L}(A)$ iff it belongs to a zone z which is reachable in the local-time zone automaton $\mathcal{Z}_l(A)$.*

3.5 Representation of Local-Time Zones

In [2], it is shown how local-time zones can be represented by difference bound matrices [7] using one additional variable per automaton. For a class of timed automata, we derive an improved representation which does not need additional space compared to the standard zone automaton.

The difference between two clocks is invariant to global delay transitions, but in the local-time model, it may be changed by a local delay transition if the clocks belong to different automata. However, since a transition $\overset{d}{\rightsquigarrow}_i$ increments both time_i and the clocks in C_i, the value $\mathit{time}_i - v_i(x)$ is invariant to local delay transitions. Indeed, it represents the timepoint at which clock x was last reset.

Consider the new variables t_i for $i \in \overline{1,n}$ (the reference time in A_i) and t_x for all clocks $x \in C$ (the last reset time of x). Denote $T_i = \{t_x \mid x \in C_i\}$ for $i \in \overline{1,n}$, $T_i^+ = T_i \cup \{t_i\}$, $T = \{t_x \mid x \in C\} = \bigcup_{i=1}^n T_i$, and $T^+ = \bigcup_{i=1}^n T_i^+$. For a configuration $(s, v, time)$, define the valuation $\bar{v} : T^+ \to \mathbb{R}^+$ by $\bar{v}(t_i) = time_i$ for $i \in \overline{1,n}$ and $\bar{v}(t_x) = time_i - v(x)$ for $x \in C_i$. Conversely, $\bar{v}$ uniquely determines v and $time$, and $(s, \bar{v})$ is an alternate representation for a configuration.

Any atomic clock constraint appearing in the description of A can be rewritten as a difference constraint over T^+. In a difference constraint $x - y \prec c$, both clocks belong to the same automaton A_i, and $x-y = (t_i-t_x)-(t_i-t_y) = t_y-t_x$. Likewise, $x \prec c$ and $c \prec x$ are rewritten as $t_i - t_x \prec c$ and $t_x - t_i \prec -c$.

A *local-time clock zone* is the set of valuations belonging to a local-time zone. A zone is written as $\langle s, \psi_l \rangle$ with s the control state and ψ_l the clock zone.

Proposition 3. *A local-time clock zone can be written as a difference constraint over the variables in T^+: $\psi_l = \bigwedge_{t_u, t_w \in T^+} t_u - t_w \prec c_{uw}$.*

Proof. Initially, $t_x = t_i = t_0$, $\forall x \in C_i$, $i \in \overline{1,n}$. Thus, $\psi_l = \bigwedge_{t_u,t_w \in T^+}(t_u = t_w)$. For an action transition $(s, \bar{v}) \xrightarrow{a} (s', \bar{v}')$, we have $\bar{v}'(t_u) = \bar{v}(t_u)$ for $u \notin R_a$ and $\bar{v}'(t_x) = t_{i_x}$ for $x \in R_a$ (with $x \in C_{i_x}$). We denote this by $\bar{v}' = \bar{v}[t_x \mapsto t_{i_x}]_{x \in R_a}$ and extend the notation to clock zones. Also, the enabling condition ψ_a holds for $\bar{v}$ and the reference times in $T_a = \{t_i \mid i \in active(a)\}$ are equal. Thus, $succ_a(\psi_l) = \{\bar{v}' \mid (s, \bar{v}) \xrightarrow{a} (s', \bar{v}')\} = (\psi_l \wedge \psi_a \wedge \bigwedge_{t_i,t_j \in T_a} t_i = t_j)[t_x \mapsto t_{i_x}]_{x \in R_a} = [\exists_{X_a}. \psi_l \wedge \psi_a \wedge \bigwedge_{t_i,t_j \in T_a} t_i = t_j] \wedge \bigwedge_{x \in R_a} t_x = t_{i_x}$, with $X_a = \{t_x \mid x \in R_a\}$ and $\exists_{X_a}$ denoting quantification over all variables in X_a. Since difference constraints are closed under conjunction and quantification, $succ_a(\psi_l)$ is a difference constraint.

For a local delay transition $(s, \bar{v}) \overset{d}{\rightsquigarrow}_i (s, \bar{v}')$, we have $\bar{v}'(t_i) = \bar{v}(t_i) + d$ and $\bar{v}'(t_u) = \bar{v}(t_u)$ for all $t_u \in T^+ \setminus \{t_i\}$. Denote this by $\bar{v}' = \bar{v} +_i d$ and the successor of ψ_l after an arbitrary delay $\overset{d}{\rightsquigarrow}_i$ as $\psi_l \Uparrow^i = \{\bar{v}' \mid \exists \bar{v} \in \psi_l, \exists d \in \mathbb{R}^+. \bar{v}' = \bar{v} +_i d\}$. We have $\psi_l \Uparrow^i = \exists d \in \mathbb{R}^+. \psi_l[t_i - d/t_i] = \exists t_i' \in \mathbb{R}^+. \psi_l[t_i'/t_i] \wedge t_i' - t_i \leq 0$, where $e[y/x]$ denotes substitution of y for x in e. Since $(s, \bar{v}) \overset{d}{\rightsquigarrow}_i (s, \bar{v}')$ iff $\bar{v}' = \bar{v} +_i d$ and $I_i(s_i)(\bar{v}')$ holds, we have $succ_i^A(\psi_l) = \psi_l \Uparrow^i \wedge I_i(s_i)$, again a difference constraint.

Combining action and delay steps, we obtain the relation: $succ_l^Z(\psi_l, a) = ([\exists_{X_a}. \psi_l \wedge \psi_a \wedge \bigwedge_{t_i,t_j \in T_a} t_i = t_j] \wedge \bigwedge_{x \in R_a} t_x = t_{i_x}) \Uparrow^{i_1} \ldots \Uparrow^{i_k} \wedge \bigwedge_{i \in active(a)} I_i(s_i')$.

This representation of a local-time zone is monolithic and relates reset times of clocks to reference times in *all* automata, using n auxiliary reference times. For a certain class of networks, we prove the following simpler representation:

Proposition 4. *If each synchronization transition in a network of automata A resets at least one clock in each participating automaton, a local-time clock zone has the form $\psi_l = \psi_\Delta(T) \wedge \bigwedge_{i=1}^n \psi_i(T_i, t_i)$, where:*
- $\psi_\Delta(T) = \bigwedge_{t_x \neq t_y \in T} t_x - t_y \prec c_{xy}$, *with* $c_{xy} \in \mathbb{Z}$
- $\psi_i(T_i, t_i) = \bigwedge_{t_x \in T_i}(t_i - t_x \prec c_{ix} \wedge t_x - t_i \prec c_{xi})$ *with* $c_{ix}, c_{xi} \in \mathbb{Z}$

We call A a *sync-reset* network of automata. The term $\psi_\Delta(T)$ relates pairs of two reset times, while $\psi_i(T_i, t_i)$ relates t_i to the reset times in automaton A_i.

Proof. The initial zone is written as: $init_l^Z(s^0) = \bigwedge_{x,y \in C}(t_x = t_y) \wedge \bigwedge_{i=1}^n I_i(s_i^0)$. For $succ_l^Z$, the term $\psi_l \wedge \psi_a \wedge \bigwedge_{t_i,t_j \in T_a} t_i = t_j$ from Prop. 3 has the required form, save for $t_i = t_j$. Quantification over X_a adds constraints between t_i and t_z, for $i \in active(a), t_z \in T$. By assumption, for every $i \in active(a)$, a clock $x \in R_a \cap C_i$ is reset, yielding $t_x = t_i$. Hence, constraints on $t_i - t_z$ can be included in ψ_Δ as constraints on $t_x - t_z$. Finally, executing $\Uparrow^i$ for $i \in active(a)$ removes the equalities $t_i = t_j$, and adds constraints on $t_z - t_j$ with $z \notin C_j$. Likewise, these can be replaced with $t_z - t_y$ for $y \in R_a \cap C_j$, which are in the desired form. □

Clock constraints are usually represented as *difference-bound matrices* [7], which are indexed by clock variables and whose elements are *bounds*, i.e., pairs $(\prec, c)$ corresponding to an atomic clock constraint. The component ψ_Δ of a local-time zone can be represented as a DBM of dimension $|C|$ (the total number of clocks). Each constraint ψ_i requires $2 * |C_i|$ time bounds, for a total of $2 * |C|$, i.e., an additional row and column. Thus, ψ_l can be represented by a matrix of dimension $|C| + 1$, the same size as the DBM used in the standard algorithm. However, only the submatrices corresponding to individual automata (with reference time) and the submatrix for ψ_Δ (without reference times) are subject to DBM operations. The successor computation is done first on the submatrix corresponding to the active automata (after enforcing the synchronization constraints $t_i = t_j$). Strengthened constraints may lead to the recanonicalization of ψ_Δ and possibly of submatrices for other individual automata.

If an automaton in the network has synchronization transitions that do not reset clocks, an additional clock can be inserted into the automaton for this purpose. This transforms any network of automata into a sync-reset network, with potentially fewer than n additional time variables.

3.6 Preservation of LTL_Δ Formulas

Since in the local-time model $\mathcal{L}(A)$ the execution order of transitions is relaxed, $\mathcal{L}(A)$ accepts a richer set of behaviors than $\mathcal{S}(A)$. This section establishes restrictions on the local-time model which ensure that each of its traces is equivalent to a trace of the standard model with respect to a given LTL_Δ formula φ.

We extend LTL_Δ to the local-time model by defining the satisfaction of an atomic time constraint in a configuration: $(s, \bar{v}) \models x - y \prec c$ iff $\bar{v}(t_y) - \bar{v}(t_x) \prec c$. For $x \in C_i$ and $y \in C_j$ we have $\bar{v}(t_y) - \bar{v}(t_x) = (time_j - v(y)) - (time_i - v(x))$. Thus, in a synchronized configuration, the semantics is the same as in $\mathcal{S}(A)$. The transitions which affect the truth value of a formula are identified as follows:

Definition 11. *(Visibility) A transition $(s, v) \to (s', v')$ is* invisible *with respect to a specification φ if every atomic subformula of φ that has the same truth value in (s, v) and (s', v'). A transition which is not* invisible *is called* visible.

A transition in $\mathcal{L}(A)$ is visible if it connects two states which differ by at least one atomic proposition in the specification or it resets at least one clock in the specification, affecting the truth value of a difference constraint. Delay transitions are invisible, since they don't change the control state and don't reset clocks.

For a network of timed automata A and a formula φ in LTL_Δ denote by $\mathcal{F}^\varphi(A)$ the set of those traces of $\mathcal{L}(A)$ which satisfy the following properties:
• *Ordering* (**O**): Visible transitions occur in increasing order of their execution times. That is, in any trace $\sigma \in \mathcal{F}^\varphi(A)$, for visible transitions τ_k and τ_l with $k < l$, we have $time(\tau_k) \leq time(\tau_l)$ (where $time(\tau)$ is the execution time of τ).
• *Fairness* (**F**): Time progress is unbounded in all automata. That is, for any trace $\sigma \in \mathcal{F}^\varphi(A)$, $i \in \overline{1,n}$ and $M \in \mathbb{R}^+$, there exists $k \in \mathbb{N}$ with $time_i(\sigma_k) > M$.

Theorem 4. *Given an LTL_Δ formula φ, for any execution trace in $\mathcal{S}(A)$ there exists an execution trace in $\mathcal{F}^\varphi(A)$ with the same truth value for φ and vice versa.*

Proof. The direct implication is straightforward: from a trace σ in $\mathcal{S}(A)$ construct a trace σ_l in $\mathcal{L}(A)$ by replacing each global delay transition $\overset{d}{\rightsquigarrow}$ with the sequence of local delay transitions $\overset{d}{\rightsquigarrow}_1 \ldots \overset{d}{\rightsquigarrow}_n$. The trace σ_l satisfies **O**, since no action transitions are reordered, and **F**, since the same delay transitions are executed in each automaton. Because delay transitions are invisible, this transformation preserves the truth value of φ, and $\sigma \models \varphi$ iff $\sigma_l \models \varphi$.

For the reverse implication, we construct σ from σ_l by reordering all transitions in increasing order of their timepoints. The ordering condition **O** guarantees that no visible transitions are reordered, and the truth value of the formula is not changed. Delay transitions may be split and reordered so every action transition is preceded by equal delays in all automata. The fairness condition **F** guarantees that for all automata, local delay transitions with the needed amount exist in σ_l. Finally, all local delay transitions between two consecutive action transitions are merged into a global delay transition, resulting in a trace σ of $\mathcal{S}(A)$. □

Based on the above theorem, we proceed as follows: We first define a restricted local-time model $\mathcal{L}^\varphi(A)$ whose traces satisfy the ordering condition **O**. Next, we construct a zone automaton $\mathcal{Z}_l^\varphi(A)$ whose states are local-time *atoms*, i.e., sets of configurations with the same truth value for all atomic subformulas of φ. We show a correspondence between the traces of $\mathcal{L}^\varphi(A)$ and $\mathcal{Z}_l^\varphi(A)$, and then impose a fairness condition corresponding to **F** to ensure equivalence with the standard model. Finally, we apply a *maximization* to the atoms in $\mathcal{Z}_l^\varphi(A)$ to obtain an automaton $\mathcal{M}_l^\varphi(A)$ which is finite and therefore amenable to model checking.

To preserve the ordering of visible transitions, we introduce a new reference variable t_v, denoting the timepoint of the last visible transition executed. The domain of the valuation $\bar{v}$ is extended to include t_v. In the initial configuration, $\bar{v}(t_v) = 0$. The model $\mathcal{L}^\varphi(A)$ is defined in the same way as $\mathcal{L}(A)$, but with the additional restriction $\bar{v}(t_v) \leq time(a)$ for executing a visible transition a, and $\bar{v}'(t_v) = time(a)$ in the resulting configuration. Thus, each visible transition is executed at a later timepoint than the previous one, and condition **O** holds.

The zone successor formula for a visible transition becomes: $succ_a^v(\psi_l) = [\exists_{X_a} \exists t_v . \psi \wedge \psi_a \wedge \bigwedge_{t_i, t_j \in T_a} t_i = t_j \wedge \bigwedge_{t_i \in T_a} t_v \leq t_i] \wedge \bigwedge_{t_i \in T_a} t_v = t_i \wedge \bigwedge_{x \in R_a} t_x = t_{i_x}$, For invisible transitions, the successor operation remains the same.

The ordering condition **O** can also be ensured without a new variable by a stronger condition on the traces of $\mathcal{L}^\varphi(A)$. This requires a visible transition to

precede in time *all* action transitions which follow it in the execution trace and is enforced by the conjunct $\bigwedge_{j \notin active(a)} time(a) \leq t_j$.

In this case, the zone successor formula for visible transitions is written: $succ_a^v(\psi_l) = [\exists_{X_a}.\psi \wedge \psi_a \wedge \bigwedge_{t_i,t_j \in T_a} t_i = t_j \wedge \bigwedge_{t_i \in T_a, t_j \notin T_a} t_i \leq t_j] \wedge \bigwedge_{x \in R_a} t_x = t_{i_x}$.

To perform model checking, we consider zones in which every configuration satisfies the same atomic subformulas of the specification φ (cf. [18]):

Definition 12. *(Atom) Given a timed automaton A and a LTL_Δ formula φ, an* atom *is a zone $\langle s, \psi_l \rangle$ such that $\bar{v}_1(t_y) - \bar{v}_1(t_x) \prec c \Leftrightarrow \bar{v}_2(t_y) - \bar{v}_2(t_x) \prec c$ for all $\bar{v}_1, \bar{v}_2 \in \psi_l$ and any constraint $x - y \prec c$ in φ.*

For each atomic clock constraint in φ, consider a new atomic proposition $q_k = t_{y_k} - t_{x_k} \prec_k c_k$. Thus, φ is reduced to a next-time free LTL formula φ_q. All configurations in an atom have the same truth value for all propositions q_k. The atoms comprising a zone $\langle s, \psi_l \rangle$ are given by the nonempty intersections between ψ_l and all constraints $t_{y_k} - t_{x_k} \prec_k c_k$, either in positive or negated form:

$atoms^\varphi(\langle s, \psi_l \rangle) = \{\langle s, \phi \rangle \mid \phi = \psi_l \wedge \bigwedge_{k=1}^m q'_k,\ \phi \neq false,\ q'_k = q_k \text{ or } q'_k = \neg q_k\}$.

Define transitions between atoms as follows: $z \stackrel{a}{\Rightarrow} z'$ if $a \in enabled(z)$ and $z' \in atoms^\varphi(succ_l^Z(z, a))$, and $z \stackrel{\epsilon}{\Rightarrow} z$ if at least one local state of z has the invariant $I_i(s_i) = true$. We obtain an *atom graph* for A and the formula φ:

Definition 13. *(Atom graph) The atom graph $\mathcal{A}^\varphi(A)$ of a timed automaton A with respect to formula φ is a state-transition graph $(Z_l^\varphi, Z_l^0, \Rightarrow)$, with Z_l^0 the set of initial local-time zones, $\Rightarrow$ the atom transition relation and Z_l^φ the set of atoms reachable from Z_l^0 by repeated application of $\Rightarrow$.*

Then, our problem reduces to LTL model checking:

Proposition 5. *For each execution trace σ_l of $\mathcal{L}^\varphi(A)$, there is an atom sequence in $\mathcal{A}^\varphi(A)$ that has the same truth value for φ_q as σ_l has for φ and vice versa.*

Proof. The proof is based on reordering transitions as in Prop. 2 (cf. also [18]), with $\stackrel{a}{\Rightarrow}$ transitions corresponding to series of action-delay transitions in $\mathcal{L}^\varphi(A)$. In addition, $\stackrel{\epsilon}{\Rightarrow}$ transitions correspond to delay transitions in automata which remain indefinitely at a state with the invariant *true*. Again, the ordering condition **O** ensures that the truth value of the formula is preserved. □

We now restrict the zone execution sequences such that the execution traces included herein satisfy the fairness condition **F**. Otherwise, the local-time model may contain traces that stop executing some automata and do not correspond to any trace in the standard model. The fairness condition **F** is violated if the execution trace does not make infinite time progress in some automaton, i.e., if the growth of a clock is always restricted by a state invariant. This cannot happen if any clock which is infinitely often limited by an invariant is reset infinitely often, allowing time to diverge. The fairness constraint can thus be written in terms of the structure of the automaton, $\bigwedge_{x \in C} \mathbf{GF}x.bounded \Rightarrow \mathbf{GF}x.reset$. The model checking problem on the initial network of automata is thus reduced to LTL model checking of a finite Kripke structure with a set of fairness constraints.

A stronger fairness constraint restricts the atom graph $\mathcal{A}^{\varphi}(A)$ to zones that are *synchronizable*, i.e., contain at least one *synchronized* configuration (with $\bar{v}(t_i) = \bar{v}(t_j)$ for all $i, j \in \overline{1,n}$). This ensures that no more zones are explored in the local-time zone automaton than in the standard zone automaton, and the reduction is applied to a state space which is not larger than the original one. This guarantee comes at the expense of an additional check for the enabledness of transition $\stackrel{a}{\Rightarrow}$ in a given atom z, namely that $succ_l^Z(z,a)$ be synchronizable.

3.7 Building a Finite Model

The local-time zone automaton can be infinite, since difference bounds on clocks can become arbitrarily large. In [2], a finite quotient is shown to exist, but no method to compare local-time zones for equivalence is given. We show that, just as for the standard zone automaton, the actual value of time bounds does not affect the enabledness of transitions, once a certain value is exceeded. Hence, each local-time zone can be normalized to obtain a finite model.

We adapt the *maximization* (rounding) operation described, e.g., in [17] to the local-time model. Let $c_{\max}$ be the maximum absolute value of all constants in the automaton A and the formula φ. Adapting the region graph construction of [1], two valuations $\bar{v}$ and $\bar{v}'$ are called *region-equivalent* (written $\bar{v} \simeq_{\mathrm{reg}} \bar{v}'$) if for any time variables $t_u, t_v \in T^+$, either $\lfloor \bar{v}(t_u) - \bar{v}(t_v) \rfloor = \lfloor \bar{v}'(t_u) - \bar{v}'(t_v) \rfloor$ or both differences have the same sign and are greater in absolute value than $c_{\max}$. Region equivalence extends to configurations by defining $(s, \bar{v}) \simeq_{\mathrm{reg}} (s', \bar{v}')$ iff $s = s'$ and $\bar{v} \simeq_{\mathrm{reg}} \bar{v}'$. *Regions* are the equivalence classes induced by $\simeq_{\mathrm{reg}}$ on the set of configurations Σ_C. It is straightforward to show:

Lemma 1. *Let $\bar{v} \simeq_{\mathrm{reg}} \bar{v}'$. Then:*
1. If ψ is any constraint in A or in the specification φ, then $\bar{v} \in \psi$ iff $\bar{v}' \in \psi$.
2. For any clock set R, $\bar{v}[R \mapsto 0] \simeq_{\mathrm{reg}} \bar{v}'[R \mapsto 0]$.
3. For $i \in \overline{1,n}$ and $d \geq 0$ there exists $d' \geq 0$ such that $\bar{v} +_i d \simeq_{\mathrm{reg}} \bar{v}' +_i d'$.

Since Lemma 1 covers all operations involved in executing a transition, the following property follows (cf. [1]):

Proposition 6. *Let $\gamma \simeq_{\mathrm{reg}} \gamma'$ be two region-equivalent configurations in Σ_C.*
1. If $\gamma \stackrel{a}{\rightarrow} \gamma_1$, there exists $\gamma_1' \simeq_{\mathrm{reg}} \gamma_1$ such that $\gamma' \stackrel{a}{\rightarrow} \gamma_1'$.
2. If $\gamma \stackrel{d}{\rightsquigarrow}_i \gamma_1$, there exists $d' \in \mathbb{R}^+$ and $\gamma_1' \simeq_{\mathrm{reg}} \gamma_1$ such that $\gamma' \stackrel{d'}{\rightsquigarrow}_i \gamma_1'$.

The maximization of a zone z is the set of configurations which have some region-equivalent configuration in z: $\max(z) = \{\gamma' \in \Sigma_C \mid \exists \gamma \in z.\ \gamma \simeq_{\mathrm{reg}} \gamma'\}$. A maximized zone is therefore a convex union of regions. It is easily seen that a maximized zone is obtained from the canonical representation of a zone by modifying all constraints involving constants $\pm c'$ with $c' > c_{\max}$: $t_u - t_v \prec -c'$ becomes $t_u - t_v < -c_{\max}$ and $t_u - t_v \prec c'$ becomes $t_u - t_v < \infty$ (trivially true). Furthermore, by point (1) of Lemma 1, a maximized atom is in turn an atom. Define $succ_l^M(z,a) = \max(succ_l^Z(z,a))$ and let $\mathcal{M}_l^{\varphi}(A)$ be the atom graph

induced by $succ_l^M$ through repeated application from an initial zone. Since the constants in a maximized zone are bounded, it follows that $\mathcal{M}_l^\varphi(A)$ is finite.

By Prop. 6, the same transitions are enabled in every point of a region. Since a maximized atom is the closure of an atom with respect to region equivalence, this implies that the atom graph $\mathcal{A}^\varphi(A)$ and the maximized atom graph graph $\mathcal{M}^\varphi(A)$ are bisimilar. Putting the previous results together, we obtain the following theorem, which reduces our initial problem to LTL model checking with fairness constraints on a finite model:

Theorem 5. *The model $\mathcal{M}_l^\varphi(A)$ with the fairness constraint* $\mathbf{F}$ *is equivalent to the standard model $\mathcal{S}(A)$ with respect to the formula φ.*

3.8 Partial Order Reduction

Partial order reduction constructs only a representative part of the state space of a model, while preserving the verified property. This is done by exploring a subset of the enabled transitions at each states, instead of the entire set. Several criteria for choosing the subset of explored transitions have been developed. We follow the approach of Peled [14], in which the selected transitions are denoted as an *ample set* and have to satisfy the following conditions:
C0 *Emptiness:* $ample(s) = \emptyset$ iff $enabled(s) = \emptyset$.
C1 *Ample decomposition:* On any path from any state s, a transition in $ample(s)$ appears before the first transition dependent on a transition in $ample(s)$.
C2 *Invisibility:* If $ample(s) \neq enabled(s)$, all transitions in $ample(s)$ are invisible.
C3 *Cycle closing:* A transition enabled in every state of a cycle in the reduced state graph belongs to the ample set of some state on that cycle.

Having established the visible transitions in the model $\mathcal{M}_l^\varphi(A)$, one needs to determine the transition dependence relation. Bengtsson et al. [2] give a purely structural dependence relation, identical to that for untimed parallel composition: two transitions are independent if the two sets of automata involved in each of them are disjoint. This condition is sufficient for the local-time model $\mathcal{L}(A)$, as shown by Theorem 1. Since transitions in the zone automaton are composed of action and local delay transitions in the local-time model, the commutativity relation also follows for the zone automaton:

$$succ_l^Z(succ_l^Z(z,a),b) = succ_l^Z(succ_l^Z(z,b),a) \text{ if } active(a) \cap active(b) = \emptyset$$

However, in the local-time zone automaton, just like in the standard zone automaton, transitions which are both enabled in a zone may actually be enabled in different sets of configurations belonging to that zone.

Let A_1 and A_2 be two automata with clock sets $\{x, u\}$ and $\{y, v\}$, and consider a zone that is reached after executing two synchronization transitions, one resetting x and y, and the second resetting u and v. Thus, we have $t_x = t_y$ and $t_u = t_v$. Assume now that transition a in A_1 has enabling condition $x - u = t_u - t_x < 2$ and transition b in A_2 requires $y - v = t_v - t_y > 3$. Since $t_u - t_x = t_v - t_y$ due to the previous synchronizations, the two conditions cannot be satisfied simultaneously. Exploring either of $\stackrel{a}{\Rightarrow}$ and $\stackrel{b}{\Rightarrow}$ restricts the current local-time zone to a fragment where the other transition is no longer enabled.

Consequently, when selecting an ample set of transitions, one needs to check, just as for full state exploration, whether for every configuration in the current zone each of the explored automata is either be forced to execute an action transition or allows indefinite time progress. Otherwise, a potential deadlock exists. For a local transition, this check can be made statically by analyzing the invariant of the originating state together with the guard condition of the transition. This gives us a practical condition for the selection of an ample set:

Proposition 7. *In a sync-reset network of automata, a local transition in a process with a single clock does not disable transitions in other automata.*

Proof. Given local transition a in automaton A_i with a single clock x, the constraints in the enabling condition of a can be of the form $t_x - t_i \prec c$ and $t_i - t_x \prec c$. In a sync-reset network of automata, the representation of a local-time constraint links t_i only to clocks in the same automaton, i.e., to t_x. Therefore, the conjunction $\psi_l \wedge \psi_a$ does not induce stronger constraints on the other time variables and does not affect the enabledness of transitions in other automata. □

Based on the above results, we can use the ample set approach [14] to construct a reduced model for the automaton $\mathcal{M}_l^{\varphi}(A)$, and perform model checking by composing it with the tableau for the LTL formula [16].

4 Conclusions and Future Work

We have presented a method that allows the application of partial order reduction to systems modeled as a composition of timed automata. The method results in reduction in the state space, as well as in the number of clock zones that are generated for each control state. Compared to previous related work, this paper shows that partial order reduction can be used for model checking of properties described in a timed extension of linear temporal logic, rather than just for local reachability analysis. Furthermore, for a certain class of automata, we show that the local-time zones can be represented as efficiently as standard clock zones. We also analyze the dependence relation between transitions in the new model and give practical conditions for selecting an ample set.

An implementation of the presented algorithm is in progress, and we expect it to support the theoretical claims for efficiency improvement with experimental results. We also plan to extend the technique to models with other variants of synchronization, such as timed automata with deadlines. Of particular interest is a detailed comparison of the present approach with techniques developed for other timed models, such as time Petri nets and timed event level structures, and possible improvements that can result from here. Finally, we plan to explore how partial order reduction can be used for other finite quotient representations of timed automata, such as the region graph construction.

Acknowledgements

The author wishes to thank his advisor, Ed Clarke, for careful reading and suggestions on improving earlier drafts of this paper.

References

1. R. Alur and D. Dill. Automata for modeling real-time systems. In *Automata, Languages, and Programming.* 17^{th} *Int. Colloquium Proc.*, *LNCS* v. 443, pp. 322-35, Coventry, UK, July 1990. Springer.
2. J. Bengtsson, B. Jonsson, J. Lilius, and Wang Yi. Partial order reductions for timed systems. In *CONCUR'98: Concurrency Theory.* 8^{th} *Int. Conf. Proc.*, *LNCS* v. 1466, pp. 485-500, Nice, France, September 1998. Springer.
3. W. Belluomini and C. J. Myers. Verification of timed systems using POSETs. In *Computer Aided Verification.* 10*th Int. Conf., CAV'98. Proc.*, *LNCS* v. 1427, pp. 403-15, Vancouver, BC, Canada, June 1998. Springer.
4. *Computer Aided Verification.* 2^{nd} *Int. Conf., CAV'90. Proc.*, *LNCS* v. 531, New Brunswick, NJ, USA, June 1990. Springer.
5. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. *Logic of Programs: Workshop, Yorktown Heights, NY*, *LNCS* v. 131, pp. 52-71. Springer, May 1981.
6. D. Dams, R. Gerth, B. Knaack, and R. Kuiper. Partial-order reduction techniques for real-time model checking. In *Proc. 3rd Int. Workshop on Formal Methods for Industrial Critical Systems*, pp. 157-69, Amsterdam, The Netherlands, May 1998.
7. D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. *Proc. Int. Workshop Automatic Verification Methods for Finite State Systems.*, *LNCS* v. 407, pp. 197-212, Grenoble, June 1989. Springer.
8. P. Godefroid. Using partial orders to improve automatic verification methods. In [4], pp. 176-85.
9. T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. In *Proc. Seventh Ann. IEEE Symp. on Logic in Computer Science*, pp. 394-406, Santa Cruz, CA, USA, June 1992. IEEE Comp. Soc. Press.
10. J. Lilius. Efficient state space search for time Petri nets. In *Proc. MFCS'98 Workshop on Concurrency*, Brno, Czech Republic, August 1998. Elsevier.
11. K. G. Larsen, P. Pettersson, and Wang Yi. Compositional and symbolic model-cheking of real-time systems. In *Proc.* 16*th IEEE Real-Time Systems Symp.*, pp. 76-87, Pisa, Italy, Dec. 1995. IEEE Comp. Soc. Press.
12. F. Pagani. Partial orders and verification of real-time systems. In *Formal Techniques in Real-Time and Fault-Tolerant Systems.* 4^{th} *Int. Symp. Proc.*, *LNCS* v. 1135, pp. 327-46, Uppsala, September 1996. Springer.
13. F. Pagani. *Ordres partiels pour la vérification de systèmes temps réel (Partial orders for verification of real-time systems).* PhD thesis, Centre d'Études et de Recherches de Toulouse, September 1997.
14. D. Peled. All from one, one for all: on model checking using representatives. In *Computer Aided Verification.* 5*th Int. Conf., CAV'93. Proc.*, *LNCS* v. 697, pp. 409-23, Elounda, Greece, June 1993. Springer.
15. A. Valmari. A stubborn attack on state explosion. In [4], pp. 156-65.
16. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. Symp. on Logic in Computer Science*, pp. 332-44, Cambridge, MA, USA, June 1986. IEEE Comp. Soc. Press.
17. H. Wong-Toi. *Symbolic Approximations for Verifying Real-Time Systems.* PhD thesis, Stanford University, December 1994.
18. T. Yoneda and B.-H. Schlingloff. Efficient verification of parallel real-time systems. *Formal Methods in System Design*, 11(2):197-215, August 1997.