

Layered Composition for Timed Automata*

Ernst-Rüdiger Olderog and Mani Swaminathan

Department of Computing Science
University of Oldenburg, Germany

{olderog,mani.swaminathan}@informatik.uni-oldenburg.de

Abstract. We investigate layered composition for real-time systems modelled as (networks of) timed automata (TA). We first formulate the principles of layering and transition independence for TA, and demonstrate the validity of the communication closed layer (CCL) laws in such a setting, by means of an operator for layered composition that is intermediate between parallel and sequential composition. Next, we introduce the principles of input/output (i/o) and partial-order (po) equivalences, and show that such equivalences are preserved when the layered composition operator is replaced by sequential composition within the expressions appearing in the CCL laws. Finally, we proceed to show that such layering (together with equivalences obtained through the CCL laws) can be useful in the design and verification of dense real-time systems that consist of a network of interacting components, by bringing about a reduction of the state-space through the exploitation of transition independence. This is illustrated by considering a collision avoidance protocol developed for an audio/video system of Bang and Olufsen.

1 Introduction and Related Work

Real-time systems have strict timing requirements and are used within many safety critical applications. Such systems are becoming increasingly complex, and often consist of multiple parallel interacting components, with the interaction taking place by means of shared variables or by message passing along common channels. Reasoning about such systems is much easier when the execution of the components is viewed *sequentially*, as opposed to corresponding *distributed or concurrent representations*. This is because the *physical structure* of the system is often in the form of multiple *parallel* interacting components, each executing a (sequential) program, while the system's *logical structure* is in the form of a complex protocol consisting of a sequence of *layers*, wherein each layer consists of many actions distributed across the whole system [1].

Such a distinction between the physical and logical structures of a distributed system has motivated research into techniques such as *communication closedness* [2] for transforming concurrent/distributed representations of the system into

* This work has been partially funded by the German Research Council (DFG) as part of the Transregional Collaborative Research Center “Automatic Verification and Analysis of Complex Systems” (SFB/TR 14 AVACS, www.avacs.org).

appropriate (and equivalent) *layered* ones, in order to enable easier reasoning about system properties.

Layering for real-time systems was first investigated in [3], where systems are expressed in a process language that allows for the specification of events of a (sub-)process within a given time interval (expressed by means of a timing operator), with interaction among the (sub-)processes being via shared variables. The timing operator is used here to enforce a temporal order between events, which is then exploited in the layer transformation rules. A process algebra with a new operator for *layered composition* (intermediate between sequential and parallel composition) based on hierarchical graphs is presented in [4]. This layered composition operator is then used to formalize *equivalences* between distributed and layered representations of a system, by exploiting *independence* between events across multiple system components, through the *communication closed layer (CCL) laws*. Section 9.1 of [4] discusses real-time extensions of the process algebra, with real-time behaviour being embedded into the algebra by means of explicit constructs (such as a delay construct) that model the passage of time. Such real-time extensions to the process algebra are elaborated further in [1], wherein an assertional proof system is presented that is then used for checking real-time side conditions for (extended) CCL laws.

However, the assertional proof system presented in the above works is not amenable to efficient and automatic tool-based verification. There has since been extensive study of the formalism of *timed automata (TA)* as an effective means for the modelling and (automatic) verification of real-time systems. TA [5] extend (finite) automata by augmenting them with non-negative real-valued clocks, in order to quantitatively capture the timing behaviour of the system. The TA model has been shown to be very amenable to automatic verification of large real-time systems, with TA model checkers such as UPPAAL [6] and KRONOS [7] having been successfully used in industrial case studies (such as [8]). We therefore propose here to investigate notions of layering and communication closedness in the context of TA. Our approach thus differs from that of previous related works on layering (particularly [1,4]) in the following aspects:

- Previous works on layering present assertional proof systems for validating (real-time) side conditions for the applicability of the CCL laws for process algebras based on hierarchical graphs, under notions of process independence, and the techniques therein are not amenable to efficient and automatic verification. Our approach however investigates the conditions for applicability of the CCL laws on timed automata based models that come equipped with extensive tool-support for automatic verification.
- The layered composition operator used in all cases relies crucially on notions of *independence* between actions, transitions, and processes across a system. Previous works (particularly [1,4]) rely on explicitly postulated notions of (in-)dependence, based on *dependency graphs*. Our work additionally uses a semantic notion of transition independence (involving the conditions of *enabledness* and *commutativity*), similar to the ones that have been extensively

studied in the context of *partial order reduction* approaches [9] (for tackling the state-space explosion problem in model checking).

- The approach in previous works on layering is mainly motivated by a design process that starts with a sequential algorithm (corresponding to the system’s logical structure) and then gradually transforms it into a distributed version (corresponding to the system’s actual implementation), by successively applying the layering operator. The approach in our work, on the other hand, is motivated by the necessity of transforming a distributed representation of a system into an equivalent layered one, in order to permit easier verification of system properties. Such easier verification is made possible by means of the notion of *input/output (i/o) and partial order (po) equivalences* between the sequential and layered compositions of processes [4]. We establish in this paper the validity of such i/o and po equivalences also between sequential and layered compositions of timed automata, and exploit their use in reducing the state space of large real time systems for easier verification.

Partial order reduction for timed automata has been investigated in many papers [10,11,12,13,14,15,16,17], with a view towards reducing the system’s state space by exploiting transition independence, with efficient symbolic techniques being integrated within the UPPAAL model checker, by means of the UPPAAL PORT extension [16]. Our goal here is to exploit the notions of transition independence used within such partial order reduction techniques for timed automata, in order to bring about a further reduction of a large system’s state space by transforming its distributed representation into an equivalent layered version, by investigating conditions for the applicability of the CCL laws. It should be noted here that our layering approach is *complementary* to partial order reduction based approaches, while exploiting similar (in-)dependence notions at different levels. More specifically, our contributions here are the following:

1. We first formulate the principles of layering and communication closedness for (networks of) timed automata, by exploiting notions of transition independence in order to formulate the CCL laws. We then proceed to show the validity of such CCL laws in the absence of “cross-dependencies” between system components, corresponding to [1,4].
2. Next, we formulate the notions of i/o and po equivalences, and show that these equivalences are preserved when one replaces layered composition by sequential composition within the expressions in the CCL laws.
3. We then proceed to show that such layering can be useful in the design and verification of dense real-time systems that consist of a network of interacting components, by exploiting (in-)dependencies in order to bring about a reduction of the state-space, through the CCL laws, together with i/o and po equivalences. Such a reduction is shown to complement partial order reduction approaches applied to (parallel compositions of) timed automata. This is illustrated by considering a collision avoidance protocol that was developed for an audio/video system of Bang and Olufsen [8].

The rest of the paper is structured as follows. Section 2 reviews timed automata and their semantics. Section 3 discusses composition techniques for

timed automata, and introduces our operator \bullet for layered composition, and the consequent CCL laws. Section 4 introduces the notions of i/o and po equivalences, and demonstrates their validity in the context of the CCL laws. Extension of these results to TA enriched with data (widely used within the model checker UPPAAL) and a methodology of using these results for state-space reduction is also discussed. Section 5 provides preliminary ideas on the application of our results by considering a collision avoidance protocol that was developed for an audio/video system of Bang and Olufsen [8]. Section 6 concludes the paper along with directions for future research.

2 Preliminaries

Given a finite set C of *clocks*, a *clock valuation* over C is a map $v : C \rightarrow \mathbb{R}_{\geq 0}$ that assigns a non-negative real value to each clock in C . If $|C| = n$, a clock valuation is identified with a point in $\mathbb{R}_{\geq 0}^n$, which we henceforth denote by $\vec{u}, \vec{v}, \vec{x}, \vec{y}$ etc. By $\vec{0}$ we denote the clock valuation where all clocks are set to 0.

A *zone* over a set of clocks C is a constraint defined by the following grammar $g ::= x \triangleright d \mid g \wedge g$, where $x \in C$, $d \in \mathbb{N}$, and $\triangleright \in \{<, \leq, >, \geq\}$. The set of zones over C is denoted $Z(C)$. The subset of zones having only upper bounds $<, \leq$ is denoted by $Z_U(C)$. In the sequel we shall identify zones with the set of clock valuations satisfying them, so that set-theoretic operations may be applied on zones.

Definition 1 (Timed automaton). A timed automaton (TA) is a tuple $A = (L, \Sigma, C, l_0, l_F, Inv, E)$, where

- L is a finite set of locations, Σ a finite alphabet, and C a finite set of clocks,
- $l_0 \in L$ is the initial location, and $l_F \in L$ the final location, with $l_0 \neq l_F$,
- $Inv : L \rightarrow Z_U(C)$ assigns a clock invariant to each location,
- $E \subseteq L \times \Sigma \times Z(C) \times 2^C \times L$ is a finite set of directed edges between locations. An edge $e = (l, a, g, r, l')$ from l to l' involves an action $a \in \Sigma$, a guard $g \in Z(C)$, and a reset set $r \subseteq C$.

For a clock valuation \vec{x} , its *time-passage* is $timepass(\vec{x}) = \{\vec{x} + d \mid d \geq 0\}$, where $\vec{x} + d$ denotes the addition of a scalar $d \in \mathbb{R}_{\geq 0}$ to each component of \vec{x} . The *k-region-equivalence relation* \approx_k on clock valuations \vec{x} and \vec{y} is defined by

$$\vec{x} \approx_k \vec{y} \text{ iff } \forall i \leq n : \left(\begin{array}{l} (x_i > k) \wedge (y_i > k) \\ \vee (int(x_i) = int(y_i) \wedge (fr(x_i) = 0 \Leftrightarrow fr(y_i) = 0) \wedge) \\ \quad \forall j \leq n : (fr(x_i) \leq fr(x_j) \Leftrightarrow fr(y_i) \leq fr(y_j)) \end{array} \right),$$

where, for a clock valuation $\vec{x} \in \mathbb{R}_{\geq 0}^n$, x_i denotes its i -th component, i.e., the value of the i -th clock, and $int(x_i)$ and $fr(x_i)$ denote the integer and fractional parts of x_i , respectively. By $[\vec{x}]_k$ we denote the k -region containing \vec{x} , which is the equivalence class induced by \approx_k .

The semantics of a TA is given in terms of its underlying *timed transition system*, which consists of an infinite set of states of the form (l, \vec{x}) , where $l \in L$ and

$\vec{x} \in \mathbb{R}_{\geq 0}^n$, with the transitions between states resulting in the formation of *canonical paths* through the transition system.

Definition 2 (Canonical path). A canonical path π through such a timed transition system is a (possibly infinite) sequence $\langle (l_0, \vec{x}_0), (l_1, \vec{x}_1), \dots \rangle$ of states, subject to the following initiation and consecution conditions:

1. *Initiation:* l_0 is the initial location and $\vec{x}_0 = \vec{0}$.
2. *Consecution (time-passage):* for even i we require $l_{i+1} = l_i$ and $\vec{x}_{i+1} \in \text{Inv}(l_i) \cap \text{timepass}(\vec{x}_i)$.
3. *Consecution (edges):* for odd i we require $\exists e = (l_i, a, g, r, l_{i+1}) \in E$: $\vec{x}_i \in \text{Inv}(l_i) \cap g$ and $\vec{x}_{i+1} \in \text{Inv}(l_{i+1}) \cap r(\vec{x}_i)$.¹

Let Π denote the set of all such canonical paths. A *transition* between two consecutive states in such a path thus corresponds either to time-passage within a location, or to an edge-traversal between discrete locations. The reachable state space of the TA is then given by the set of states that are reachable from the initial state, through the transitions of all possible canonical paths, and is defined as follows.

Definition 3 (Reachable state space). $\text{Reach}(A) \subseteq L \times (C \rightarrow \mathbb{R}_{\geq 0})$ is the reachable state space of a TA A , consisting of an infinite set of states of the form (l, \vec{x}) , where $l \in L$ and $\vec{x} \in \mathbb{R}_{\geq 0}^n$. It is defined inductively as follows, with $\text{Reach}_i(A)$ denoting the reach-set under $i \in \mathbb{N}$ steps, starting from the initial state $(l_0, \vec{0})$ and alternating between time-passage and discrete transitions:

- $\text{Reach}_0(A) = \{(l_0, \vec{0})\}$,
- $\text{Reach}_{i+1}(A) = \text{Reach}_i(A) \cup \text{Succ}(\text{Reach}_i(A))$, where
 - if $i \geq 0$ even, $\text{Succ}(\text{Reach}_i(A)) = \left\{ (l, \vec{x}) \mid \begin{array}{l} \exists \vec{u} \in \text{Inv}(l) : (l, \vec{u}) \in \text{Reach}_i(A) \\ \wedge \vec{x} \in \text{timepass}(\vec{u}) \cap \text{Inv}(l) \end{array} \right\}$
 - if $i \geq 0$ odd, $\text{Succ}(\text{Reach}_i(A)) = \left\{ (l, \vec{x}) \mid \begin{array}{l} \exists e = (l', a, g, r, l) \in E \\ \exists \vec{u} \in \text{Inv}(l') \cap g : \\ (l', \vec{u}) \in \text{Reach}_i(A) \\ \wedge \vec{x} \in \text{Inv}(l) \cap r(\vec{u}) \end{array} \right\}$
- $\text{Reach}(A) = \bigcup_{i \in \mathbb{N}} \text{Reach}_i(A)$.

This leads to the following notion of *reachability equivalence* denoted by \equiv . Given two TA A_1 and A_2 , we define

$$A_1 \equiv A_2 \quad \text{iff} \quad \forall i \in \mathbb{N} : \text{Reach}_i(A_1) = \text{Reach}_i(A_2).$$

Thus the equivalence requires equal sets of reachable states after every iteration of the transition relation.

¹ To simplify subsequent proofs, we use even- and odd-numbered steps to distinguish between time-passage and taking edges between discrete locations. Here $r(\vec{x})$ denotes the valuation obtained from \vec{x} after resetting all the clocks in r .

3 Layered Composition and CCL Laws

We have thus far considered the semantics of timed automata that operate in isolation. However, real-time systems in practice *communicate* with each other and with the environment, and this results in a *composite* system consisting of communicating components. The communication between components is often through synchronization actions drawn from a shared alphabet. We now define three operators for constructing such composite systems: *sequential*, *parallel*, and *layered* composition, where the latter is new for timed automata.

In the sequel we consider timed automata $A_i = (L_i, C_i, l_{0_i}, l_{F_i}, \text{Inv}_i, E_i, \Sigma_i)$, $i = 1, 2$, with disjoint locations and clocks: $L_1 \cap L_2 = C_1 \cap C_2 = \emptyset$.

Definition 4 (Sequential composition). *Let $l_{0_1} \neq l_{F_1}$ and $l_{0_2} \neq l_{F_2}$. Then the sequential composition of A_1 and A_2 is defined as the timed automaton*

$$A_1; A_2 = (L_1 \cup L_2 \cup \{\widetilde{l}_{F_1}\}, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, l_{0_1}, l_{F_2}, \text{Inv}_1 \cup \text{Inv}_2, E),$$

where \widetilde{l}_{F_1} is a copy of l_{F_1} disjoint from $L_1 \cup L_2$, with $\text{Inv}(\widetilde{l}_{F_1}) = \text{Inv}(l_{F_1}) \wedge \text{Inv}(l_{0_2})$, and E is given by:

$$\begin{aligned} E = & (E_1 \setminus \{(l_1, a_1, g_1, r_1, l_{F_1}) \mid (l_1, a_1, g_1, r_1, l_{F_1}) \in E_1\}) \\ & \cup \{(l_1, a_1, g_1, r_1, \widetilde{l}_{F_1}) \mid (l_1, a_1, g_1, r_1, l_{F_1}) \in E_1\} \\ & \cup (E_2 \setminus \{(l_{0_2}, a_2, g_2, r_2, l_2) \mid l_{0_2}, a_2, g_2, r_2, l_2 \in E_2\}) \\ & \cup \{(\widetilde{l}_{F_1}, a_2, g_2, r_2, l_2) \mid (l_{0_2}, a_2, g_2, r_2, l_2) \in E_2\}. \end{aligned}$$

Thus $A_1; A_2$ is obtained by first performing the actions in A_1 and then performing the actions in A_2 . The final location l_{F_1} of A_1 and the initial location l_{0_2} of A_2 are amalgamated to a new location \widetilde{l}_{F_1} . It is assumed here that invariants of l_{F_1} and l_{0_2} are mutually consistent². This definition is adapted from [18,19].

Definition 5 (Parallel composition). *The parallel composition is defined by*

$$A_1 \parallel A_2 = (L_1 \times L_2, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, (l_{0_1}, l_{0_2}), (l_{F_1}, l_{F_2}), \text{Inv}, E),$$

where $\forall (l_1, l_2) \in L_1 \times L_2 : \text{Inv}(l_1, l_2) = \text{Inv}_1(l_1) \wedge \text{Inv}_2(l_2)$ and E given by:

- *Synchronization:* If $e_i = (l_i, a_i, g_i, r_i, l'_i) \in E_i$ for $i = 1, 2$ with $a_1 = a_2$ then $((l_1, l_2), a_1, g_1 \wedge g_2, r_1 \cup r_2, (l'_1, l'_2)) \in E$.
- *Interleaving:* (1) If $e_1 = (l_1, a_1, g_1, r_1, l'_1) \in E_1$ with $a_1 \notin \Sigma_2$ then $\forall l_2 \in L_2 : ((l_1, l_2), a_1, g_1, r_1, (l'_1, l_2)) \in E$.
- (2) *Conversely,* if $e_2 = (l_2, a_2, g_2, r_2, l'_2) \in E_2$ with $a_2 \notin \Sigma_1$ then $\forall l_1 \in L_1 : ((l_1, l_2), a_2, g_2, r_2, (l_1, l'_2)) \in E$.

² Our example in Section 5 satisfies this condition.

Note that parallel composition is *symmetric*, it involves a CSP-style *synchronization* on common actions and *interleaving* on disjoint actions. This composition does not respect the *dependencies* between the individual components³. As mentioned in the introduction, a real-time distributed system often consists of (sequential) phases that execute in parallel on multiple platforms, wherein an action (resp. transition) within a given phase can execute only after all *dependent* actions (resp. transitions) in each preceding phase have been executed.

Motivated by this, we now proceed to define *layered composition* between two TA. As in [1,4], we assume in this definition an explicitly postulated notion of *dependencies* between the *actions* of the timed automata that constitute the layered composition, as part of the overall system specification. The edges of the timed automata are then dependent iff they execute dependent actions.

Actions/edges that are not dependent are termed *independent*. Two independent actions need to satisfy the conditions of *enabledness* and *commutativity*. Enabledness here implies that they do not disable each other, while commutativity implies that they can be executed in either order starting from a given input state, and yet result in the same (or an “equivalent”) output state. We refer to [10,16] for formal definitions of independent actions in timed automata. The dependency between two actions a and b is denoted $a \rightsquigarrow b$, their independence by $a \not\rightsquigarrow b$. We stipulate that the dependency relation \rightsquigarrow is reflexive. Two TA A_1 and A_2 are said to be *independent*, denoted $A_1 \not\rightsquigarrow A_2$, iff every action of A_1 is independent of every action of A_2 .

Definition 6 (Layered composition). *The layered composition is defined by*

$$A_1 \bullet A_2 = (L_1 \times L_2, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, (l_{01}, l_{02}), (l_{F1}, l_{F2}), Inv, E),$$

where Inv and E are as in the parallel composition $A_1 \parallel A_2$, except that part (2) of the interleaving case is now different:

- *Synchronization:* If $e_i = (l_i, a_i, g_i, r_i, l'_i) \in E_i$ for $i = 1, 2$ with $a_1 = a_2$ then $((l_1, l_2), a_1, g_1 \wedge g_2, r_1 \cup r_2, (l'_1, l'_2)) \in E$.
- *Interleaving:* (1) If $e_1 = (l_1, a_1, g_1, r_1, l'_1) \in E_1$ with $a_1 \notin \Sigma_2$ then $\forall l_2 \in L_2 : ((l_1, l_2), a_1, g_1, r_1, (l'_1, l_2)) \in E$.
- (2) If $e_2 = (l_2, a_2, g_2, r_2, l'_2) \in E_2$ with $a_2 \notin \Sigma_1$ and $\forall l_1, l_1^* \in L_1 : l_1 \xrightarrow{*} l_1^* \forall e_1 = (l_1^*, a_1, g_1, r_1, l'_1) \in E_1 : a_1 \not\rightsquigarrow a_2$, then $((l_1, l_2), a_2, g_2, r_2, (l_1, l'_2)) \in E$,

where $l_1 \xrightarrow{*} l_1^*$ expresses that l_1^* is reachable from l_1 in the syntactic structure of A_1 through an arbitrary sequence of edges.

Thus only part (2) of the interleaving case differs from parallel composition: an interleaving edge of the second automaton A_2 is allowed to execute only after all

³ We assume *local time semantics* as in [10,16] to obtain fewer dependencies induced by timing.

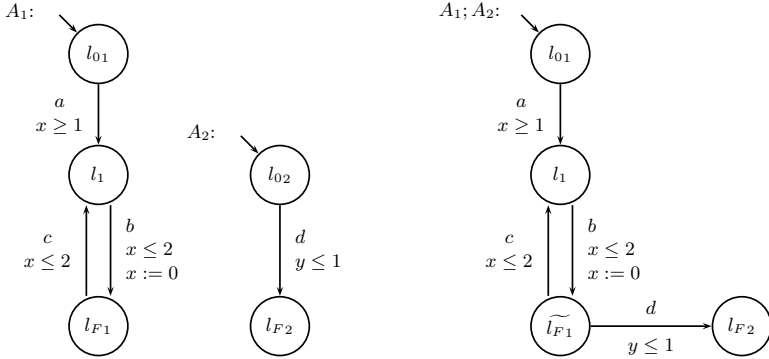


Fig. 1. *Left:* timed automata A_1 and A_2 with the stipulated dependency $a \rightsquigarrow d$; *right:* sequential composition $A_1; A_2$

dependent edges of the first automaton A_1 have been executed. Figures 1 and 2 illustrate the three composition operators for two simple timed automata.

We now proceed to formulate and validate the CCL laws (equivalences) of [1,4] for layered composition of timed automata.

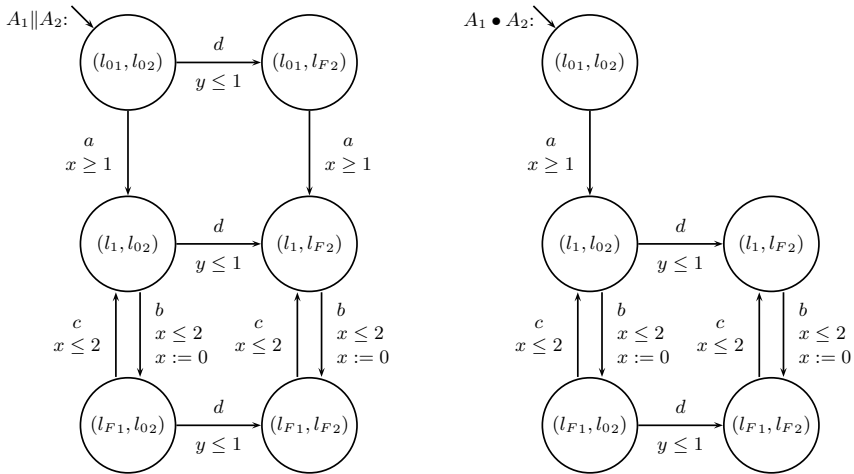


Fig. 2. *Left:* parallel composition $A_1 || A_2$; *right:* layered composition $A_1 \bullet A_2$

Theorem 1 (CCL laws for timed automata). *For all timed automata $A_1, A_2, B_1,$ and $B_2,$ with $A_1 \rightsquigarrow B_2$ and $A_2 \rightsquigarrow B_1,$ the following communication closed layer equivalences (CCL laws) hold for the reachability equivalence \equiv :*

1. $A_1 \bullet B_2 \equiv A_1 || B_2$ (Indep)
2. $(A_1 \bullet A_2) || B_2 \equiv A_1 \bullet (A_2 || B_2)$ (CCL-L)

3. $(A_1 \bullet A_2) \parallel B_1 \equiv (A_1 \parallel B_1) \bullet A_2$ (CCL-R)
4. $(A_1 \bullet A_2) \parallel (B_1 \bullet B_2) \equiv (A_1 \parallel B_1) \bullet (A_2 \parallel B_2)$ (CCL)

Proof. Owing to space limitations, we provide below only a brief sketch of the proof of the law CCL-L. The proofs of the other CCL laws are similar.

Given TA A_1, A_2, B_2 , with $A_1 \rightsquigarrow B_2$, we show $(A_1 \bullet A_2) \parallel B_2 \equiv A_1 \bullet (A_2 \parallel B_2)$, i.e., the sets of reachable states are equal at every iteration i of their transition relations.⁴ The proof is by induction over i .

The containment $Reach_i(A_1 \bullet (A_2 \parallel B_2)) \subseteq Reach_i((A_1 \bullet A_2) \parallel B_2)$ is not hard to see intuitively, as the parallel composition operator dominates on the right-hand side and the layered composition operator on the left-hand side. The dominance of layered composition induces fewer interleavings on the basis of the respective dependencies, as seen from the definition earlier.

We show $Reach_i((A_1 \bullet A_2) \parallel B_2) \subseteq Reach_i(A_1 \bullet (A_2 \parallel B_2))$ by induction over i . *Induction Basis.* This case $i = 0$ is obvious.

Assume that the containment holds for some i .

Induction Step. Consider $((l_{A_1}, l_{A_2}, l_{B_2}), (\vec{u}, \vec{v}, \vec{w})) \in Reach_{i+1}((A_1 \bullet A_2) \parallel B_2)$. For the case where $((l_{A_1}, l_{A_2}, l_{B_2}), (\vec{u}, \vec{v}, \vec{w})) \in Reach_i((A_1 \bullet A_2) \parallel B_2)$ the proof is again immediate from the induction hypothesis. We now examine the cases where $((l_{A_1}, l_{A_2}, l_{B_2}), (\vec{u}, \vec{v}, \vec{w})) \in Succ(Reach_i((A_1 \bullet A_2) \parallel B_2))$.

If i is even, the preceding transition corresponds to time-passage, which is possible also in $A_1 \bullet (A_2 \parallel B_2)$, and the proof then follows immediately.

If i is odd, using the fact that A_1 and B_2 are independent (thereby ruling out synchronization between A_1 and B_2), we see that the preceding transition corresponds to an action which could have been performed (a) either by A_1, A_2, B_2 individually, (b) or as a synchronization action involving A_1 and A_2 , (c) or as a synchronization action involving A_2 and B_2 .

The cases corresponding to the preceding transition having been executed individually either by A_1 or by A_2 are relatively straightforward, as are the cases (b) and (c). We now consider the case where the preceding transition corresponds to an action performed by B_2 alone. This means that there exist $((l_{A_1}, l_{A_2}, l'_{B_2}), (\vec{u}', \vec{v}', \vec{w}')) \in Reach_i((A_1 \bullet A_2) \parallel B_2)$ and $e = (l'_{B_2}, a, g, r, l_{B_2}) \in E_{B_2}$ such that $\vec{u}' \in Inv(l_{A_1}) \cap g$, $\vec{v}' \in Inv(l_{A_2}) \cap g$, $\vec{w}' \in Inv(l_{B_2}) \cap g$, with $\vec{u} = r(\vec{u}')$, $\vec{v} = r(\vec{v}')$, $\vec{w} = r(\vec{w}')$. The proof is then immediate from the induction hypothesis, and using the fact that A_1 and B_2 are independent. \square

4 Equivalences

We now formalize the notions of *input/output (i/o)* and *partial order (po) equivalences* as a means of relating sequential and layered composition. In particular, we show that, for two timed automata A_1 and A_2 , it is *always* the case that $A_1; A_2$ and $A_1 \bullet A_2$ are i/o and po equivalent, irrespective of any (in-) dependence relation between the actions of A_1 and A_2 .

⁴ We identify nested pairs $((x, y), z)$ and $(x, (y, z))$ of locations with tuples (x, y, z) .

We elaborate on these equivalences, through the following definitions and lemmas.

Definition 7 (i/o equivalence of paths). *Given two TA A_1 and A_2 with Π_1 and Π_2 denoting the corresponding sets of finite, canonical paths ending in l_{F_1} and l_{F_2} , respectively, and given a relation \approx relating the locations of A_1 and A_2 , a path $\pi_1 \in \Pi_1$ is i/o equivalent to a path $\pi_2 \in \Pi_2$ (relative to \approx), denoted $\pi_1 \equiv_{i/o} \pi_2$, where $\pi_i = \langle (l_{0_i}, \vec{0}), \dots, (l_{F_i}, \vec{x}_i) \rangle, i = 1, 2$, iff $l_{0_1} \approx l_{0_2}$, $l_{F_1} \approx l_{F_2}$, and $\vec{x}_1 \approx_k \vec{x}_2$, where k is the maximum of all constants in A_1 and A_2*

Definition 8 (Layered normal form). *A (finite) canonical path π of $A_1 \bullet A_2$ is in layered normal form (LNF) if it consists of consecutive transitions from E_1 passing through l_{F_1} , followed by consecutive transitions from E_2 ending in l_{F_2} (i.e., no transition from E_2 precedes a transition from E_1).*

Definition 9 (po equivalence of paths). *Let A_1 and A_2 be two TA sharing a common alphabet Σ , with Π_1 and Π_2 denoting the corresponding sets of finite, canonical paths. Let \approx be a relation between the locations of A_1 and A_2 . A path $\pi_1 \in \Pi_1$ is po equivalent to $\pi_2 \in \Pi_2$, denoted $\pi_1 \equiv_{po} \pi_2$, relative to \approx on the corresponding locations, and region-equivalence on the corresponding clock-valuations (w.r.t the maximum constant of A_1 and A_2) if π_i can be obtained from π_{3-i} by repeated permutation of adjacent independent transitions separated by only one time-passage.*

Thus, two po equivalent paths π_1 and π_2 (relative to region-equivalence on their clock valuations and \approx on their locations) differ only in the (permutative) ordering of *independent* transitions. This definition has been adapted for TA from [20].

Lemma 1. *Let A_1 and A_2 be two TA, let Π denote the set of all finite, canonical paths of $A_1 \bullet A_2$, and $\Pi_L \subseteq \Pi$ the subset of these paths that are in LNF. It then holds that $\forall \pi \in \Pi \exists \pi' \in \Pi_L : \pi \equiv_{i/o} \pi' \wedge \pi \equiv_{po} \pi'$.*

The proof follows from the definitions of layered composition of TA, and of i/o and partial order equivalences between paths of a TA. Thus, every path of a layered composition can be rewritten into an i/o and po equivalent path that is in layered normal form.

The notions of i/o and po equivalence are then lifted to TA as follows:

Definition 10 (i/o and po equivalence of TA). *Let A_1 and A_2 be two TA sharing a common alphabet Σ , with Π_1 and Π_2 denoting the corresponding sets of finite, canonical paths that end in their respective final states. Then A_1 and A_2 are i/o (resp. po) equivalent, denoted $A_1 \equiv_{i/o} A_2$ (resp. $A_1 \equiv_{po} A_2$), iff $\forall \pi_1 \in \Pi_1 \exists \pi_2 \in \Pi_2 : \pi_1 \equiv_{i/o} \pi_2$ (resp. $\pi_1 \equiv_{po} \pi_2$), and conversely, $\forall \pi_2 \in \Pi_2 \exists \pi_1 \in \Pi_1 : \pi_1 \equiv_{i/o} \pi_2$ (resp. $\pi_1 \equiv_{po} \pi_2$).*

We then have the following theorem that establishes the i/o and po equivalence between sequential and layered compositions of timed automata

Theorem 2. *For any two TA A_1 and A_2 we have $A_1 \bullet A_2 \equiv_{i/o} A_1; A_2$ and $A_1 \bullet A_2 \equiv_{po} A_1; A_2$.*

Proof. We provide a brief sketch of the proof here owing to space limitations.

We first introduce a relation \succ that relates locations of $A_1 \bullet A_2$ with those of $A_1; A_2$. $\forall l_1 \in L_1$ with $l_1 \neq l_{F1} : (l_1, l_{02}) \succ l_1$, $\forall l_2 \in L_2$ with $l_2 \neq l_{02} : (l_{F1}, l_2) \succ l_2$, and $(l_{F1}, l_{02}) \succ \widetilde{l_{F1}}$ (where $\widetilde{l_{F1}}$ is as defined for sequential composition).

Let Π (resp. Π^\bullet) be the set of all finite, canonical paths of $A_1; A_2$ (resp. $A_1 \bullet A_2$) ending in l_{F2} . Then

- $\forall \pi \in \Pi: \exists \pi' \in \Pi^\bullet : \pi' \equiv_{i/o} \pi \wedge \pi' \equiv_{po} \pi$, such that π' is in LNF.
- $\forall \pi \in \Pi^\bullet \exists \pi' \in \Pi : \pi \equiv_{i/o} \pi' \wedge \pi \equiv_{po} \pi'$, where π is either in LNF, or is i/o and po equivalent to another path in Π^\bullet that is in LNF (cf. Lemma 1).

The i/o and po equivalence between π and π' above is relative to \succ between their respective locations, and to region equivalence (w.r.t the maximum of all constants of A_1 and A_2) between the clock valuations. The i/o and po equivalence between $A_1; A_2$ and $A_1 \bullet A_2$ then follows as an immediate consequence. \square

Corollary 1. *Replacing \bullet by $;$ within the expressions appearing in Theorem 1 yields i/o and partial order equivalences.*

Proof. The proof follows from Theorems 1 and 2. \square

A consequence of Corollary 1 is the preservation of (timed) LTL and CTL properties without the *next* operator (see [12] and Chapter 8 of [21]).

Extensions to TA with Data. We have thus far considered (simple) TA that communicate by means of synchronization actions drawn from a shared alphabet, and an explicitly postulated notion of a dependency relation between actions. TA models used within model checkers such as UPPAAL are often extended with *data variables* that range over finite subsets of integers. We do not provide a formal definition of such extended TA, but instead refer the reader to Section 4.4 of [22] for the details concerning their syntax and semantics.

For such extended TA, one may now *infer* the dependencies from the syntactic structure of the given TA, based on the *Read* and *Write* sets associated with the actions. Two actions are dependent in such a setting if one of the two writes a variable that is read or written by the other action (see Section 4 of [1] for formal details).

The complementary *independence* relation then respects the commutativity condition necessary for partial order reduction. I/O and po equivalence for paths of such extended TA are defined as earlier (i.e., relative to a \approx -relation on their locations and appropriate region-equivalence on their clock valuations), together with *identity on their data valuations*.

Theorems 1 and 2 and Corollary 1 then carry over to extended TA under such modified notions of i/o and po equivalences.

Methodology. We now outline the intended methodology on a schematic example. Suppose we want to verify for TA A_1, A_2, B_1 that the system $(A_1; A_2) \parallel B_1$ satisfies a temporal formula φ without the *next* operator. If $A_2 \leftrightarrow B_1$ we can reduce the state space of the TA system by applying to it the equivalences stated above, as shown on the right-hand side. Thus it suffices to check that $(A_1 \parallel B_1); A_1$ satisfies φ . If each of A_1, A_2, B_1 has 10 locations then original TA system has 200 locations whereas the transformed system has 110 locations.

$$\begin{aligned}
 & (A_1; A_2) \parallel B_1 \\
 \equiv_{po} & \{ \text{Corollary 1} \} \\
 & (A_1 \bullet A_2) \parallel B_1 \\
 \equiv & \{ \text{CCL-R} \} \\
 & (A_1 \parallel B_1) \bullet A_2 \\
 \equiv_{po} & \{ \text{Corollary 1} \} \\
 & (A_1 \parallel B_1); A_2.
 \end{aligned}$$

We proceed to apply such a methodology for easier automatic verification of a large system consisting of a network of data-enriched TA.

5 Example: Audio/Video Collision Avoidance Protocol

We present in this section preliminary ideas on the application of the techniques discussed so far (in particular, the CCL laws and the corresponding i/o and po equivalences) towards easier automatic verification of large real-time systems modelled as networks of timed automata. For this purpose, we consider a collision avoidance protocol that was developed for an audio/video system of Bang and Olufsen, whose formal modelling and analysis is considered in detail in [8], using the UPPAAL tool for modelling the protocol as a network of timed automata. The treatment of the protocol in this section is brief, and is only intended to illustrate a possible application of the layering and partial order equivalences discussed hitherto for easier verification of networks of TA.

The UPPAAL model of the collision avoidance protocol presented in [8] consists of a network of nine timed automata communicating in parallel. The protocol schematic is described in Figure 3. This is a simplified version of the schematic found in [8], omitting the names of shared variables and channels. The schematic essentially describes the communication between two (sender) systems A and B that send data frames via a shared *Bus*.⁵ Each (sender) system consists of a corresponding Sender (S_A, S_B), Detector (Det_A, Det_B), Frame Generator (FG_A, FG_B), and Observer (Obs_A, Obs_B).

The protocol is given by the Sender and Detector, where the Sender transmits frames over the shared bus, while the Detector performs collision detection. The Frame Generator and Observer in a sense constitute the *environment* in which the protocol operates. The protocol together with its environment is then represented by a parallel composition of nine (data-enriched) timed automata:

$$System = S_A \parallel Obs_A \parallel Det_A \parallel FG_A \parallel Bus \parallel S_B \parallel Obs_B \parallel Det_B \parallel FG_B.$$

This system is required to satisfy the following informal correctness properties

1. Any frame transmitted by a Sender X (where X is A or B) that is destroyed due to collision is (eventually) detected by X .
2. Collision detection must be simultaneous across all senders.

⁵ Receiver systems are not relevant for the analysis.

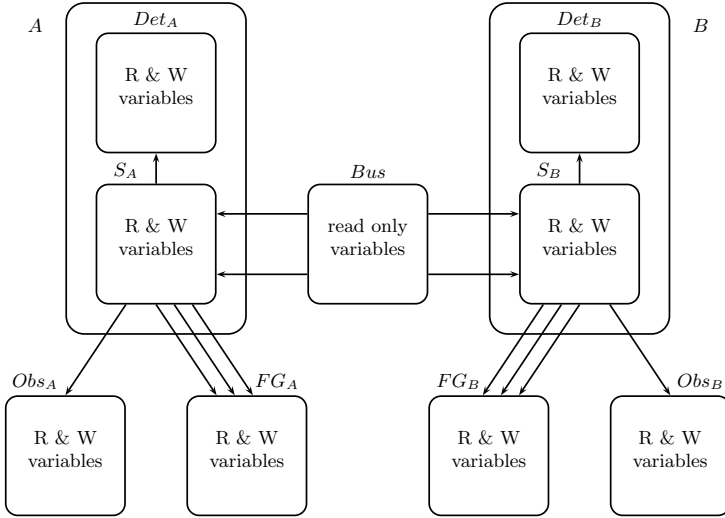


Fig. 3. Audio/video protocol as a network of timed automata, adapted from [8]

These properties may be formally expressed as a CTL formula (without the *next* operator) involving conditions on the shared data variables (see [8]).

Based on the detailed timed automata models given in [8], we find that each Sender System comprises 19 discrete locations, each Detector 8 locations, each Frame Generator 7 locations, each Observer 1 location, while the Bus contains 2 discrete locations. Thus the total number of discrete locations in the parallel composition is $19^2 \times 8^2 \times 7^2 \times 1^2 \times 2$ (i.e., over 2 million discrete locations). The timing behaviour of the system is governed by a single clock A_c per sender that runs locally, consistent with the *local time semantics* introduced in [10].

We however note that the execution of each Sender System described in [8] consists of *three sequential phases* corresponding to *initialization* (*Init*) (comprising 8 discrete locations), *transmission* (*Tx*) (comprising 5 discrete locations), and *collision response* (*Coll*) (comprising 6 discrete locations). For simplicity, we consider only a single run of the protocol and thus ignore cross-over edges between phases. Each Sender (as given in Figure 10 of [8]) may then be described by the following sequential composition of timed automata

$$S_X = Init_X; Tx_X; Coll_X,$$

where X denotes either A or B .

Exploiting the fact that certain “cross-dependencies” do not exist in the system (for instance, between Tx_A and $Init_B$), and the consequent application of the CCL laws and the corresponding partial order equivalences, we may rewrite the system’s composition as follows:

$$\begin{aligned}
System' = Bus \parallel & ((Init_A \parallel Init_B) \\
& ; \\
& (FG_A \parallel FG_B \parallel TX_A \parallel TX_B) \\
& ; \\
& (Det_A \parallel Det_B \parallel Obs_A \parallel Obs_B \parallel Coll_A \parallel Coll_B))
\end{aligned}$$

Based on the methodology described at the end of Section 3, it follows that *System* and *System'* are po equivalent, and thus the satisfaction of the desired correctness property (expressed in CTL without *next*) is preserved when transforming *System* into *System'*. An advantage of such a transformation is that *System'* is much easier to reason about than *System* (given that ; dominates in the former, as opposed to || in the latter). In fact, it may be seen that the number of discrete locations in *System'* is a little over 7000, yielding a state space reduction by a factor of over 300.

Such a layered transformation may be seen as being complementary to the well-studied partial order reduction approach to the model checking of networks of timed automata, given that exactly the same class of system properties is preserved. In fact, such a layered transformation performs, in a certain sense, partial order reduction on the system *a priori*.

6 Conclusion

We have presented a framework for layered reasoning of complex real-time systems modelled as networks of timed automata. This was achieved by means of a layered composition operator that enables a combination of both parallel execution and sequential verification, by appropriately exploiting (in-)dependence conditions across components. The approach complements the partial order reduction approach in the verification of real-time systems, in the sense that layered transformation using the CCL laws and the resulting i/o and po equivalences bring about an *a priori* (partial order) reduction of the state space to be explored. Preliminary ideas on the application of the approach have been illustrated on a realistic example. Future work includes the extension of these techniques to more complex models of real-time systems such as Phase Event Automata [23], and their application to detailed analysis on realistic examples.

Acknowledgements. We wish to thank the reviewers for useful feedback.

References

1. Janssen, W., Poel, M., Xu, Q., Zwiers, J.: Layering of Real-Time Distributed Processes. In: Langmaack, H., de Roever, W.-P., Vytupil, J. (eds.) FTRTFT 1994 and ProCoS 1994. LNCS, vol. 863, pp. 393–417. Springer, Heidelberg (1994)
2. Elrad, T., Francez, N.: Decomposition of Distributed Programs into Communication Closed Layers. *Science of Computer Programming* 2(3), 155–173 (1982)
3. Zwiers, J.: Layering and Action Refinement for Timed Systems. In: Huizing, C., de Bakker, J.W., Rozenberg, G., de Roever, W.-P. (eds.) REX 1991. LNCS, vol. 600, pp. 687–723. Springer, Heidelberg (1992)
4. Janssen, W.: Layered Design of Parallel Systems, PhD Dissertation, Universiteit Twente (1994)

5. Alur, R., Dill, D.: A Theory of Timed Automata. *Theoretical Computer Science* 126(2), 183–235 (1994)
6. Behrmann, G., David, A., Larsen, K.: A Tutorial on Uppaal. In: Bernardo, M., Corradini, F. (eds.) *SFM-RT 2004*. LNCS, vol. 3185, pp. 200–236. Springer, Heidelberg (2004)
7. Bozga, M., Daws, C., Maler, O., Olivero, A., Tripakis, S., Yovine, S.: KRONOS: A Model-Checking Tool for Real-Time Systems. In: Ravn, A.P., Rischel, H. (eds.) *FTRTFT 1998*. LNCS, vol. 1486, pp. 298–302. Springer, Heidelberg (1998)
8. Havelund, K., Skou, A., Larsen, K.G., Lund, K.: Formal modeling and analysis of an audio/video protocol: an industrial case study using UPPAAL. In: *RTSS'97*, pp. 2–13. IEEE CS Press, Los Alamitos (1997)
9. Godefroid, P.: Using Partial Orders to Improve Automatic Verification Methods. In: Clarke, E., Kurshan, R.P. (eds.) *CAV 1990*. LNCS, vol. 531, pp. 176–185. Springer, Heidelberg (1991)
10. Bengtsson, J., Jonsson, B., Lilius, J., Yi, W.: Partial Order Reductions for Timed Systems. In: Sangiorgi, D., de Simone, R. (eds.) *CONCUR 1998*. LNCS, vol. 1466, pp. 485–500. Springer, Heidelberg (1998)
11. Dams, D., Gerth, R., Knaack, B., Kuiper, R.: Partial-order Reduction Techniques for Real-time Model Checking. *Formal Aspects of Computing* 10(5-6), 469–482 (1998)
12. Minea, M.: Partial Order Reduction for Model Checking of Timed Automata. In: Baeten, J.C.M., Mauw, S. (eds.) *CONCUR 1999*. LNCS, vol. 1664, pp. 431–436. Springer, Heidelberg (1999)
13. Zhao, J., Xu, H., Li, X., Zheng, T., Zheng, G.: Partial Order Path Technique for Checking Parallel Timed Automata. In: Damm, W., Olderog, E.-R. (eds.) *FTRTFT 2002*. LNCS, vol. 2469, pp. 417–432. Springer, Heidelberg (2002)
14. Lugiez, D., Niebert, P., Zennou, S.: A Partial Order Semantics Approach to the Clock Explosion Problem of Timed Automata. *Theoretical Computer Science* 345(1), 27–59 (2005)
15. Ben Salah, R., Bozga, M., Maler, O.: On Interleaving in Timed Automata. In: Baier, C., Hermanns, H. (eds.) *CONCUR 2006*. LNCS, vol. 4137, pp. 465–476. Springer, Heidelberg (2006)
16. Haakansson, J., Petterson, P.: Partial Order Reduction for Verification of Real-Time Components. In: Raskin, J.-F., Thiagarajan, P.S. (eds.) *FORMATS 2007*. LNCS, vol. 4763, pp. 211–226. Springer, Heidelberg (2007)
17. Malinowski, J., Niebert, P.: SAT Based Bounded Model Checking with Partial Order Semantics for Timed Automata. In: Esparza, J., Majumdar, R. (eds.) *TACAS 2010*. LNCS, vol. 6015, pp. 405–419. Springer, Heidelberg (2010)
18. Bouyer, P., Petit, A.: Composition and Decomposition of Timed Automata. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.) *ICALP 1999*. LNCS, vol. 1644, pp. 210–219. Springer, Heidelberg (1999)
19. Dong, J.S., Hao, P., Qin, S., Sun, J., Yi, W.: Timed Automata Patterns. *IEEE Transactions on Software Engineering* 34(6), 844–859 (2008)
20. Alur, R., Brayton, R.K., Henzinger, T.A., Qadeer, S., Rajamani, S.K.: Partial-Order Reduction in Symbolic State Space Exploration. In: Grumberg, O. (ed.) *CAV 1997*. LNCS, vol. 1254, pp. 340–351. Springer, Heidelberg (1997)
21. Baier, C., Katoen, J.-P.: *Principles of Model Checking*. MIT Press, Cambridge (2008)
22. Olderog, E.-R., Dierks, H.: *Real-Time Systems: Formal Specification and Automatic Verification*. Cambridge University Press, Cambridge (2008)
23. Hoenicke, J., Olderog, E.-R.: CSP-OZ-DC: A Combination of Specification Techniques for Processes, Data and Time. *Nordic Journal of Computing* 9(4), 301–334 (2002)