Structural Transformations for Data-Enriched Real-Time Systems *

Ernst-Rüdiger Olderog and Mani Swaminathan

Department of Computing Science
University of Oldenburg, Germany
{olderog, mani.swaminathan}@informatik.uni-oldenburg.de

Abstract. We investigate *structural transformations* for easier verification of real-time systems with shared data variables, modelled as networks of *extended timed automata* (ETA). Our contributions to this end are: (1) An operator for *layered composition* of ETA that yields *communication closed* equivalences under certain *independence* and / or *precedence* conditions. (2) Two *reachability preserving* transformations of *separation* and *flattening* for reducing (under certain cycle conditions) the number cycles of the ETA. (3) The interplay of the three structural transformations (separation, flattening, and layering), illustrated on an enhanced version of Fischer's *real-time mutual exclusion* protocol.

1 Introduction

Reasoning about networks of (real-time) systems is much easier when the execution of the system components is viewed sequentially, as opposed to corresponding distributed or concurrent representations. Transformations of distributed system representations to equivalent *layered* (i.e., sequential) representations were first explored in [10] through a notion of *communication closedness* between system components. Such a layered transformation was subsequently investigated in [12] for a process algebra based on hierarchical graphs, with an operator for *layered composition* (intermediate between sequential and parallel composition) that formalized *equivalences* between the distributed and layered system representations through the *Communication Closed Layer* (CCL) laws, by exploiting *independence* between system components. Real-time extensions to this process algebra were presented in [13], where CCL laws were shown to hold under certain *timing conditions*, even in the absence of cross-component independence.

The layered transformation used in the *assertion-based* reasoning techniques of the above works was recently adapted in [20] for *automatic verification* of real-time systems modelled as *timed automata* (TA) [1]. Our layered transformation in [20] aimed for state-space reduction in TA networks, based on *transition independence* as studied for *partial order reduction* of TA [4, 17, 15, 11]. We enhance

* 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).

here the layered transformation in [20] for TA, and complement this by the transformation techniques of *separation* and *flattening*. The structure of this paper and the enhancements therein w.r.t [20] are as follows :

(1) We considered in [20] networks of TA under *local time semantics*, as in many works on partial order reduction for TA (cf. [4, 17, 15, 11]), where the clocks of each constituent TA evolve independently so as to induce greater timing-based independence, but at the expense of extra *reference clocks* for resynchronization (cf. [4]). In this paper, we instead work with networks of TA extended with shared data variables (termed extended timed automata (ETA)), having *synchronous clocks* across the constituent ETA, as supported by the well-established ETA model-checker UPPAAL [3]. Dependencies in such ETA networks arise due to (a) the read-write interference of the variables shared across the ETA (b) the global timing constraints induced by synchronous clocks.

Section 2 of this paper reviews ETA and their compositional constructs, and establishes communication closed equivalences that exploit the absence of dependencies due to (a) and (b) above. Notions of non-interference are introduced for dealing with (a), while (b) is handled by *wrapped* ETA that mimic local time semantics in a network even with synchronous clocks.

Section 3 of this paper establishes communication closed equivalences for ETA with synchronous clocks that exploit global-time-induced *precedence relations*, in the presence of shared variable and clock dependencies between ETA.

(2) The explicit passage of control (from one sequential phase of the system to the next) necessary for applying (non-interference- or precedence-based) layered transformations may however not be directly apparent from the system's structure, owing to multiple nested cycles that often arise while modelling reactive distributed real-time systems as ETA networks. We therefore introduce in Section 4 the transformations of *separation* and *flattening* as (reachability preserving) pre-processing steps that (under certain cycle conditions on the ETA) reduce the nesting depth and the number of cycles in ETA networks. Communication closedness (via appropriate non-interference and/or precedence conditions) may then be easily investigated on such separated and flattened ETA, such that the verification of reachability properties may be rendered almost trivial.

(3) The interplay of the three structural transformations (separation, flattening, and layering) is illustrated in Section 5 on an enhanced version of Fischer's real-time mutual exclusion protocol for two critical sections.

Section 6 concludes the paper with further comparisons to related work. A full version of this paper is available at [19], whose Appendix A details the syntax and semantics of ETA and their compositional constructs, whose Appendix B includes proofs of the results stated in this paper, and whose Appendix C discusses the generality of these results.

2 ETA, Compositions, and CCL

We briefly review the *Extended Timed Automata* (ETA) model for (networks of) real-time systems enriched with *shared data variables*. Various compositional

operators for ETA are also introduced, along with the notion of transition independence yielding communication closed equivalences for suitably wrapped ETA in the presence of synchronous clocks.

Extended Timed Automata. Let $(\alpha, \beta, \cdots \in)\ \Sigma$ be a finite *alphabet* of *channels*. For each channel $\alpha \in \Sigma$ there are two *actions*: $\alpha?$ denotes *input* on α and $\alpha!$ *output* on α, where $\alpha?, \alpha! \notin \Sigma$. We consider two different *internal* actions $\tau, \varepsilon \notin \Sigma$, where τ results only from *synchronization*. The set of all actions over Σ is denoted by $(a, b, \cdots \in)\ \Sigma_{?!} = \{\alpha? \mid \alpha \in \Sigma\} \cup \{\alpha! \mid \alpha \in \Sigma\} \cup \{\tau, \varepsilon\}$. In the context of parallel composition, input and output are *complementary* actions that can synchronize yielding τ. For an action $a \in \Sigma_{?!} \setminus \{\tau, \varepsilon\}$, its complementary action is denoted by $\overline{a}$, i.e., $\overline{\alpha?} = \alpha!$, and vice-versa.

Let $(v \in)\ V$ be a finite set of *data variables* ranging over a finite value set D. The set $(\psi_D \in)\ \Psi(V)$ of *data expressions* over V is the set of expressions involving variables of V and the usual arithmetic operators $+, -, \cdots$. The set $(\phi_D \in)\ \Phi(V)$ of *data constraints* over V is the set of Boolean constraints over variables in V involving the usual arithmetic $(+, -, \cdots)$ and relational $(<, \leq, >, \geq)$ operators. A *data valuation* assigns to each data variable in V a value in D. If $|V| = m$, a data valuation is identified with a point in D^m, denoted typically by $\vec{u}, \vec{v}$ etc.

Let $(x, y, \cdots \in)\ C$ be a finite set of *clocks*. The set $(\phi \in)\ \Phi(C)$ of *clock constraints* over C has the following syntax: $\phi ::= x \bowtie k \mid \phi_1 \wedge \phi_2$, where $x \in C$, $k \in \mathbb{N}$, and $\bowtie \in \{<, \leq, \geq, >\}$. The set $(g \in)\ G(C, V)$ of *guards* has the syntax: $g ::= \phi \mid \phi_D \mid g_1 \wedge g_2$, where $\phi \in \Phi(C)$ and $\phi_D \in \Phi(V)$.

A *clock valuation* assigns a non-negative real value to each clock in C. If $|C| = n$, a clock valuation is identified with a point in $\mathbb{R}^n_{\geq 0}$, which we typically denote by $\vec{x}, \vec{y}$ etc. By $\vec{0}$ we denote the clock valuation where all clocks are set to 0, while $\vec{v_0} \in D^m$ denotes a designated initial data valuation. A *reset* operation is an assignment $x := 0$ to a clock $x \in C$, or an assignment $v := \psi_D$ to a data-variable $v \in V$, involving a data expression $\psi_D \in \Psi(V)$.

We may then define an *extended timed automaton* (ETA) A as a tuple $A = (L, \Sigma, C, V, l_0, l_F, Inv, E)$ over a set of (finitely many) locations, channel names, clocks, data variables, initial and final locations, invariants, and edges. An edge $e \in E$ is of the form $e = (l, a, g, r, l')$ with $l, l' \in L, a \in \Sigma_{?!}, g \in G(C, V)$, and r a list of reset operations. Define $target(e) = l'$ as the target location of the edge e, $act(e) = a$ as the *action of* e and $edges_A(a) = \{e \in E \mid act(e) = a\}$ as the set of edges in A with action a. For a pair of clock valuations $\vec{x}$ and $\vec{y}$ and a constant $k \in \mathbb{N}$, we denote by $\vec{x} \approx_k \vec{y}$ the *k-region-equivalence* between $\vec{x}$ and $\vec{y}$. The corresponding equivalence class for a clock valuation $\vec{x}$ is denoted $[\vec{x}]_k$, cf. Definitions 11 and 12 in Appendix A of [19].

The semantics of an ETA is given in terms of its *timed transition system*, which consists of an infinite set of *states* of the form $(l, \vec{x}, \vec{v})$, where $l \in L$, $\vec{x} \in \mathbb{R}^n_{\geq 0}$, and $\vec{v} \in D^m$. The transitions between such states result in the formation of *paths* through the timed transition system. Such a *path* π is a (possibly infinite) sequence $\pi = \langle (l_0, \vec{0}, \vec{v_0}) \xrightarrow{d_0} (l_1, \vec{x}_1, \vec{v}_1) \xrightarrow{e_1} (l_2, \vec{x}_2, \vec{v}_2) \xrightarrow{d_2} (l_3, \vec{x}_3, \vec{v}_3) \xrightarrow{e_3} \ldots \rangle$ of states with delays $d_i \in \mathbb{R}_{\geq 0}$ and edges $e_i \in E$, subject to the *initiation* and *consecution* conditions (cf. Definition 13 in Appendix A of [19]). This path π

induces a *timed trace* $\xi = \langle (t_0, e_1), (t_2, e_3), \ldots \rangle$ with $t_0 = d_0$ and $\forall i \in \mathbb{N} : t_{2i+2} - t_{2i} = d_{2i+2}$, where ξ consists of pairs (t, e) indicating the *absolute* time instant $t \geq 0$ at which the edge $e \in E$ is executed by ETA A along path π. Note that all timestamps in π and ξ have even subscripts. The reachable state space of the ETA A, denoted $Reach(A)$, is then given by the set of states reachable from the initial state through transitions of all paths, with $Reach_i(A)$ denoting the set of reachable states of A after i iterations of its transition relation (cf. Definition 14 in Appendix A of [19]). This leads to the notion of *reachability equivalence* denoted by $\equiv$. Given two ETA A_1 and A_2, we define $A_1 \equiv A_2$ iff $\forall i \in \mathbb{N} : Reach_i(A_1) = Reach_i(A_2)$. Thus $\equiv$ requires equal sets of reachable states after every iteration of the transition relation.

ETA Compositions. So far we considered ETA operating in isolation. In practice, real-time systems *communicate* with each other and their environment. This results in *composite* systems with communicating components. The communication is via synchronizing actions drawn from a shared alphabet and via shared data variables. We now consider four operators for constructing composite systems: *sequential*, *step*, *parallel*, and *layered* composition (where the latter is new for (extended) timed automata), defined for ETA $A_i = (L_i, \Sigma_i, C_i, V_i, l_{0i}, l_{Fi}, Inv_i, E_i)$, $i = 1, 2$, with disjoint locations: $L_1 \cap L_2 = \emptyset$.

For modeling that the execution of A_1 is followed by that of A_2, it is convenient to have two composition operators at hand. *Sequential composition* $A_1; A_2$ *amalgamates* the final location l_{F1} of A_1 with the initial location l_{02} of A_2, the *step composition* $A_1 \rhd A_2$ links l_{F1} and l_{02} by an explicit *step transition* t. Formally, $A_1; A_2 = (L_1 \cup L_2 \cup \{\widetilde{l_{F1}}\}, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, V_1 \cup V_2, l_{01}, l_{F2}, Inv, E)$, where $\widetilde{l_{F1}}$ is a new location obtained by amalgamating l_{F1} with l_{02}. We require that there is no outgoing edge from l_{F1}, as it would otherwise be possible to reenter A_1 from A_2 via incoming edges to l_{02} and outgoing edges from l_{F1}. The set of edges E is obtained by appropriately assigning $\widetilde{l_{F1}}$ as the target or source location for edges in E_1 entering l_{F1} and for edges in E_2 entering or leaving l_{02}, cf. Definition 15 in Appendix A of [19]. In the *step composition* $A_1 \rhd A_2$ defined below, the stepping transition t allows for l_{F1} to have outgoing transitions, as no location of A_1 will be re-entered once t has been executed.

Definition 1 (Step Composition). *The* step composition $A_1 \rhd A_2$*, read* A_1 step A_2*, is defined by* $A_1 \rhd A_2 = (L, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, V_1 \cup V_2, l_{01}, l_{F2}, Inv, E)$*, where* $L = L_1 \cup L_2$*, with* $Inv(l_i) = Inv_i(l_i)$ *for* $l_i \in L_i$ *and* $i = 1, 2$*, and* $E = E_1 \cup E_2 \cup \{t\}$*, where* $t = (l_{F1}, \varepsilon, true, \emptyset, l_{02})$ *steps from* l_{F1} *to* l_{02}*.*

Alternative definitions of sequential and step compositions for timed automata may be found in [5, 9]. Parallel composition $\|$ of ETA is in the CCS-style [16], i.e., parallel ETA *synchronize* on common actions but also act autonomously on all actions – the latter is modelled by *interleaving*. In order to avoid any read-write and write-write conflicts w.r.t the shared variables in the parallel ETA, we require that edges with synchronizing actions are *non-interfering*, as defined below. For an edge $e = (l, a, g, r, l')$ of an ETA A its *write-set* $wr(e)$ is the set of all clocks and data variables appearing on the left-hand side of one of the reset

operations in r, while its *read-set* $rd(e)$ is the set of all clocks and data variables appearing in the guard g or on the right-hand side of a reset operation in r.

Definition 2 (Non-interfering edges). *Let E_1 and E_2 be sets of edges. The* non-interference relation $\not\frown \subseteq E_1 \times E_2$ *is defined for $e_1 \in E_1$ and $e_2 \in E_2$ by: $e_1 \not\frown e_2$ if $rd(e_1) \cap wr(e_2) = wr(e_1) \cap rd(e_2) = wr(e_1) \cap wr(e_2) = \emptyset$.*

The relation $\not\frown$ is canonically lifted to sets of edges (and consequently to ETA): $E_1 \not\frown E_2$ iff for all $e_1 \in E_1$ and $e_2 \in E_2$ we have $e_1 \not\frown e_2$. For two ETA A_1 and A_2 with respective edge-sets E_1 and E_2, we have that $A_1 \not\frown A_2$ when (1) $E_1 \not\frown E_2$, i.e., their edge-sets are non-interfering, and (2) $C_1 \cap C_2 = \emptyset$, i.e., their clock-sets are disjoint so as to eliminate timing-induced dependencies between A_1 and A_2 by the *wrapping* construction (cf. Definition 4). In the context of parallel composition $\|$, we require a more relaxed notion of *synchronized non-interference* on the constituent ETA A_1 and A_2 for $A_1 \| A_2$ to be well-formed.

Definition 3 (Synchronized non-interfering ETA). *ETA A_1 and A_2 over alphabets Σ_1 and Σ_2, respectively, are* synchronized non-interfering*, denoted $A_1 \not\frown_{sync} A_2$, if $\forall a \in \Sigma_{1?!} \setminus \{\tau, \varepsilon\} : \overline{a} \in \Sigma_{2?!} \implies edges_{A_1}(a) \not\frown edges_{A_2}(\overline{a})$.*

The relation $\not\frown_{sync}$ on ETA is only w.r.t synchronizing actions on common channels, and thus (unlike the more restrictive $\not\frown$ relation on ETA) does not preclude shared-variable and clock dependencies between actions on disjoint channels.

The *parallel composition* of two A_1 and A_2, with $A_1 \not\frown_{sync} A_2$, is defined by $A_1 \| A_2 = (L_1 \times L_2, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, V_1 \cup V_2, (l_{01}, l_{02}), (l_{F1}, l_{F2}), Inv, E)$, where $\forall (l_1, l_2) \in L_1 \times L_2 : Inv(l_1, l_2) = Inv_1(l_1) \wedge Inv_2(l_2)$, and E is constructed according to a CCS-style synchronization and interleaving, cf. Definition 16 of Appendix A of [19]. As mentioned in the introduction, a real-time distributed system often consists of (sequential) phases that execute in parallel on multiple platforms, wherein a transition (edge) within a given phase can execute only after all *dependent* transitions (edges) in each preceding phase have been executed. It is clear that the non-interference relation of Definition 2 is sufficient to ensure independence in the untimed setting, where dependencies are induced only by shared variables. In the timed setting of ETA, however, the clocks of the various system components evolve *synchronously*, resulting in *timing-induced* dependencies even in the presence of disjoint sets of clocks. In contrast to several works on the partial order reduction of TA (e.g., [4, 17, 15]) that deal with such timing-induced dependencies by imposing the semantic condition of *local time* (where, in addition to mutual disjointness, the clocks of the constituent components run entirely independent of each other), we retain here the synchrony between the clocks of the various components as in the UPPAAL model-checker, but eliminate timing-induced dependencies by *wrapping* the ETA with an initial location that admits idling for arbitrarily long periods before proceeding to its actual execution. The wrapping concept is defined as follows:

Definition 4 (Wrapped ETA). *An ETA $A = (L, \Sigma, C, V, l_0, l_F, Inv, E)$ is* wrapped *if $Inv(l_0) = true$ and every edge $e \in E$ leaving l_0 is of the form $e = (l_0, \varepsilon, true, r, l)$, where r resets all clocks in C to 0 and all data variables in V to their initial value $\vec{v_0}$. If A is wrapped, we denote this by writing $[A]$.*

The arbitrary idling permitted in l_0 mimics local time semantics when $[A]$ is considered in the context of a (parallel) composition. Intuitively, $[A]$ is protected against time influences from components working in parallel, as long as its stays in its initial location.

We now introduce an asymmetric layered composition operator $\bullet$ (intermediate between parallel and sequential composition) that involves the non-interference relation on edges of ETA. The *layered composition* of A_1 and A_2 is given by $A_1 \bullet A_2 = (L_1 \times L_2, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, V_1 \cup V_2, (l_{01}, l_{02}), (l_{F1}, l_{F2}), Inv, E)$, where $A_1 \not\frown_{sync} A_2$, and Inv is as in the parallel composition $A_1 \| A_2$, while E is a subset of the set of edges of $A_1 \| A_2$, as an edge of A_2 is allowed to execute in $A_1 \bullet A_2$ only after all *dependent* edges of A_1 have been executed, cf. Def. 17 and Fig. 5 in Appendix A of [19]. Theorem 1 states the CCL laws of [12] for ETA.

Theorem 1 (CCL laws for ETA). *For all ETA A_1, A_2 and B_1, B_2 with $A_1 \not\frown B_2$ and $B_1 \not\frown A_2$ the* communication closed layer *(CCL) laws hold:*

1. $A_1 \bullet B_2 \;=\; A_1 \| B_2$ *(Indep)*
2. $(A_1 \bullet A_2) \| B_2 \;=\; A_1 \bullet (A_2 \| B_2)$ *(CCL-L)*
3. $(A_1 \bullet A_2) \| B_1 \;=\; (A_1 \| B_1) \bullet A_2$ *(CCL-R)*
4. $(A_1 \bullet A_2) \| (B_1 \bullet B_2) \;=\; (A_1 \| B_1) \bullet (A_2 \| B_2)$ *(CCL)*

PO-Equivalence. We now formalize *partial order* (po) equivalence as a means of relating step and layered compositions. For this relationship, we have to address the fact that in a layered composition $A_1 \bullet A_2$ there may be τ-edges arising from synchronization of complementary actions, whereas such τ-edges do not arise in the step composition $A_1 \rhd A_2$. For this purpose, we introduce for a path π of $A_1 \bullet A_2$ the operation $split(\pi)$ that *splits* every synchronization edge of π (labelled with τ) into a sequence of its constituent input and output edges, which is possible owing to the synchronized non-interference assumed for edges labelled with complementary actions (see Definition 3). Note that this non-interference implies that the order in which synchronization edges are split into constituent input and output edges is irrelevant.

Consider a finite τ-labelled path π of $A_1 \bullet A_2$ with fragments π' and π'' of the form $\pi = \pi' \xrightarrow{d_0} (l_1, \vec{x}_1, \vec{u}_1) \xrightarrow{\tau} (l_2, \vec{x}_2, \vec{u}_2) \xrightarrow{d_2} \pi''$. We then have $split(\pi) = \pi' \xrightarrow{d'_0} (l'_1, \vec{x}'_1, \vec{u}'_1) \xrightarrow{a} (l'_2, \vec{x}'_2, \vec{u}'_2) \xrightarrow{d'_2} (l'_3, \vec{x}'_3, \vec{u}'_3) \xrightarrow{\overline{a}} (l'_4, \vec{x}'_4, \vec{u}'_4) \xrightarrow{d'_4} \pi''$, with $a \in \Sigma_i$ and $\overline{a} \in \Sigma_{3-i}$, $i \in \{1, 2\}$, where $l_1 = l'_1$, $\vec{x}_1 \approx_k \vec{x}'_1$, $\vec{u}_1 = \vec{u}'_1$, $l_2 = l'_4$, $\vec{x}_2 \approx_k \vec{x}'_4$, $\vec{u}_2 = \vec{u}'_4$, and where $d_0, d'_0, d_2, d'_2, d'_4 \in \mathbb{R}_{\geq 0}$, $d_0 = d'_0$, $d'_2 = 0$, $d_2 = d'_4$, and k is the maximum of the clock constants appearing in A_1 and A_2. Following such a splitting of τ-edges, we may now define po-equivalence on paths of ETA.

Definition 5 (po equivalence of paths). *Let A_1 and A_2 be two ETA sharing an alphabet Σ, with Π_1 and Π_2 denoting the corresponding sets of finite 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, $\approx_k$ on the corresponding clock-valuations (where k is the maximum constant of A_1 and A_2), and identity on the corresponding data valuations, if $split(\pi_2)$ can time-abstractedly be obtained from $split(\pi_1)$ by repeated permutation of adjacent independent edges separated by only one time-passage.*

For $\pi_1 = \langle (l_{01}, \vec{0}, \vec{v_0}) \xrightarrow{d_0} (l_1, \vec{x}_1, \vec{u}_1) \xrightarrow{e} (l_2, \vec{x}_2, \vec{u}_2) \xrightarrow{d_2} (l_3, \vec{x}_3, \vec{u}_3) \xrightarrow{f} (l_4, \vec{x}_4, \vec{u}_4) \rangle$ and $\pi_2 = \langle (l_{02}, \vec{0}, \vec{v_0}) \xrightarrow{d'_0} (l'_1, \vec{y}_1, \vec{v}_1) \xrightarrow{f} (l'_2, \vec{y}_2, \vec{v}_2) \xrightarrow{d'_2} (l'_3, \vec{y}_3, \vec{v}_3) \xrightarrow{e} (l'_4, \vec{y}_4, \vec{v}_4) \rangle$, where $d_0, d_2, d'_0, d'_2 \in \mathbb{R}_{\geq 0}$, we say that $\pi_1 \equiv_{po} \pi_2$ relative to $\approx$ iff $l_{01} \approx l_{02}$, and $\forall 1 \leq i \leq 4 : l_i \approx l'_i$, $\vec{x}_4 \approx_k \vec{y}_4, \vec{u}_4 = \vec{v}_4$, and $e \nearrow f$. Thus, two po equivalent paths π_1 and π_2 (relative to $\approx$ on their locations, region-equivalence on their clock valuations, and identity on their data valuations) differ only in the (permutative) ordering of *independent* transitions. This definition has been adapted for ETA from [2]. The notion of po equivalence is then lifted to ETA as follows: For ETA A_1 and A_2 sharing a common alphabet Σ, with Π_1 and Π_2 denoting the corresponding sets of finite paths ending in their respective final states, we write $A_1 \equiv_{po} A_2$ iff $\forall \pi_i \in \Pi_i \ \exists \pi_{3-i} \in \Pi_{3-i} : \pi_i \equiv_{po} \pi_{3-i}$, where $i \in \{1, 2\}$.

Definition 6 (Layered Normal Form). *A (finite) path π of $A_1 \bullet A_2$ is in* layered normal form (LNF) *if it consists of consecutive edges from E_1 passing through l_{F1}, followed by consecutive edges from E_2 ending in l_{F2}.*

In a path π of $A_1 \bullet A_2$ in LNF, the A_2-transitions are *delayed* until all A_1-transitions have occurred. This may be too late because the clock constraints of the A_2-transitions may not be satisfied any more. To avoid this issue, we wrap A_2 so that starting $[A_2]$ resets all clocks of A_2. This way, we mimic a local time semantics for A_2. Now Lemma 1 states that every path of $A_1 \bullet [A_2]$ can be rewritten into a po equivalent path in LNF, which leads to Theorem 2 establishing po equivalence between layered and step compositions of ETA.

Lemma 1. *Consider an ETA A_1 that can terminate in its final location l_{F1}, and a wrapped ETA $[A_2]$. Let Π denote the set of all finite paths of $A_1 \bullet [A_2]$, and $\Pi_L \subseteq \Pi$ the subset of the paths in LNF. Then $\forall \pi \in \Pi \ \exists \pi' \in \Pi_L : \pi \equiv_{po} \pi'$.*

Theorem 2 (po equivalence between $\bullet$ and $\triangleright$). *For an ETA A_1 that can terminate, and a wrapped ETA $[A_2]$, we have that $A_1 \bullet [A_2] \equiv_{po} A_1 \triangleright [A_2]$.*

Note that $A_1 \bullet [A_2] \equiv_{po} A_1 \triangleright [A_2]$ implies that local reachability of locations is preserved, as $Reachloc(A_1) \cup Reachloc([A_2]) = Reachloc(A_1 \triangleright [A_2])$, where $Reachloc(A)$ denotes the set of reachable locations of the ETA A. Theorems 1 and 2 lead to the following corollary.

Corollary 1 (CCL laws with step) *For all ETA A_1, B_1 and all wrapped ETA $[A_2], [B_2]$ such that $A_1 \nearrow B_2$ and $B_1 \nearrow A_2$, and such that when A_1 can terminate, then so can B_1, and vice-versa, with $P = (A_1 \triangleright [A_2]) \parallel (B_1 \triangleright [B_2])$ and $S = (A_1 \parallel B_1) \triangleright ([A_2] \parallel [B_2])$, we then have that $P \equiv_L S$.*

$\equiv_L$ between P and S here is the *layered reachability equivalence* satisfying: $Reachloc(S) \subseteq Reachloc(P)$, and for any $(l_a, l_b) \in Reachloc(P)$

- if $l_a \in L_{A_i}, l_b \in L_{B_i}$, then $(l_a, l_b) \in Reachloc(S)$,
- if $l_a \in L_{A_i}, l_b \in L_{B_{3-i}}$, then $\exists\, l'_a \in L_{A_{3-i}}, l'_b \in L_{B_i} : (l_a, l'_b) \in Reachloc(S) \ \wedge$ $(l'_a, l_b) \in Reachloc(S)$,

where $i \in \{1, 2\}$.

3 Time Precedence and Timed CCL

The non-interference conditions used in the CCL laws of the preceding section may be *syntactically* inferred from the structure of the ETA. We now introduce a semantic condition termed *time precedence* and demonstrate its use in establishing equivalences analogous to Theorem 1 and Corollary 1 for ETA networks that do not respect the non-interference conditions discussed in the previous section. This time precedence relation is defined below for ETA.

Definition 7 (Time Precedence in ETA). *For ETA A_1 and A_2 we say that A_1* precedes *A_2, denoted $A_1 \prec A_2$ if $A_1 \| A_2 \equiv A_1; A_2$.*

Though $\prec$ is a semantic condition, it may be easily verified by inspecting the ETA guards, as will be shown in Section 5. For the specific case of acyclic ETA, where there is no location that is syntactically reachable from itself, we may refine Definition 7 to relate time-stamps of individual edges as follows:

Definition 8 (Time Precedence of Edges). *Consider two acyclic ETA A_1 and A_2 with corresponding edge sets E_1 and E_2, and the set $\Xi(A_1 \| A_2)$ of timed traces induced by paths of $A_1 \| A_2$. Then an edge $e_1 \in E_1$ is said to* precede *an edge $e_2 \in E_2$, denoted $e_1 \prec e_2$, if*

$$\forall \xi \in \Xi(A_1 \| A_2) \quad \begin{pmatrix} \exists t_2 \in \mathbb{R}_{\geq 0} : (t_2, e_2) \in \xi \\ \Rightarrow \quad \exists t_1 \in \mathbb{R}_{\geq 0} : [(t_1, e_1) \in \xi \quad \wedge \quad t_1 < t_2] \end{pmatrix}.$$

The following Lemma then follows as an immediate consequence:

Lemma 2 (Edge Precedence in acyclic ETA). *Given two acyclic ETA A_1 and A_2, we have that $A_1 \prec A_2 \iff \forall e_1 \in E_1 \ \forall e_2 \in E_2 : e_1 \prec e_2$.*

The CCL laws discussed previously may then be reformulated as follows, using this semantic notion of time-precedence as an alternative side-condition in the presence of syntactic dependencies.

Theorem 3 (Timed CCL laws for ETA). *For ETA A_1, A_2 and B_1, B_2 with $A_1 \prec B_2$ and $B_1 \prec A_2$ the following* timed communication closed layer *(Timed CCL) laws hold for the reachability equivalence $\equiv$:*

1. $A_1 \bullet B_2 \ \equiv \ A_1 \| B_2$ *(Timed Indep)*
2. $(A_1 \bullet A_2) \| B_2 \ \equiv \ A_1 \bullet (A_2 \| B_2)$ *(Timed CCL-L)*
3. $(A_1 \bullet A_2) \| B_1 \ \equiv \ (A_1 \| B_1) \bullet A_2$ *(Timed CCL-R)*
4. $(A_1 \bullet A_2) \| (B_1 \bullet B_2) \ \equiv \ (A_1 \| B_1) \bullet (A_2 \| B_2)$ *(Timed CCL)*

Theorem 3 then implies the reachability equivalence $\equiv$ when $\bullet$ is replaced by $;$ within the expressions of Theorem 3, as stated in the following corollary:

Corollary 2 *Replacing $\bullet$ by $;$ within the expressions appearing in Theorem 3 yields the reachability equivalence $\equiv$.*

Unlike the CCL laws in Theorem 1, the Timed CCL laws of Theorem 3 do not hold for the equality $=$. Note also that Corollary 2 does not require the ETA to the right of the $\bullet$ to be wrapped, in contrast to Corollary 1. As $\equiv$ is not a congruence w.r.t. parallel composition, the Timed CCL laws do not yield equivalences in an arbitrary parallel context.

4 Separation and Flattening

We consider in this section two transformations on cycles in ETA. *Separation* reduces the nesting of cycles, while *flattening* reduces the number of cycles. These transformations are sound (in the sense of reachability preservation) under the assumption that the ETA involved have *memoryless cycles*. Such an assumption is justified for protocols where each cycle performs some service, and there is no need to carry over some information from one service cycle to the next. Separation was studied in [7] in the abstract setting of Kleene algebras, where, under certain conditions, a nondeterministic iteration of the form $(a + b)^*$ could be separated into a sequence a^*b^* of iterations, with a and b being regular expressions for programs in a Kleene algebra. This is in essence what we will prove for ETA in the Separation Theorem of this section. The role of the nondeterministic choice $+$ on regular expressions is played by a *union* operator $\cup$ on ETA [5].

Definition 9 (Union). *Consider ETA* $A_i = (L_i, \Sigma_i, C_i, V_i, l_0, l_0, Inv_i, E_i)$, $i \in \{1, 2\}$, *with an identical initial and final location* l_0 *such that* $L_1 \cap L_2 = \{l_0\}$. $A_1 \cup A_2 = (L_1 \cup L_2, \Sigma_1 \cup \Sigma_2, C_1 \cup C_2, V_1 \cup V_2, l_0, l_0, Inv_1 \cup Inv_2, E_1 \cup E_2)$ *is then the* union *of* A_1 *and* A_2.

Whereas in the union $A_1 \cup A_2$ possible cycles of A_1 and A_2 are glued together in their initial location, in $A_1 \rhd A_2$ the new transition t separates the A_1 cycles from the A_2 cycles so that all A_1 cycles are performed before the A_2 cycles. In contrast to the other composition operators of the previous sections requiring the location disjointness condition $L_1 \cap L_2 = \emptyset$, the operator $\cup$ in this section instead requires that A_1 and A_2 have a common *memoryless* initial location, where the notion of a location being memoryless is as defined below:

Definition 10 (Memoryless locations in ETA). *A location* l *of an ETA* A *is said to be* memoryless *if* l *is always entered with the initial valuations of the clocks and the data variables.*

A sufficient syntactic condition for a location l to be memoryless is that all cycles through l have strong resets. A cycle of an ETA A through a location l is said to have *strong resets* if every transition entering l resets all clocks and all data variables. To simplify the reasoning about cycles, we wish to transform the union of ETA with memoryless cycles into their step composition. The Separation Theorem shows that this transformation respects a weak reachability equivalence $\equiv_r$ sufficient for the preservation of safety properties, as seen next.

Definition 11 (Weak Reachability Equivalence). *For ETA* A *and* A', *with* $A = (L, \Sigma, C, V, l_0, l_F, Inv, E)$ *and* $A' = (L', \Sigma', C', V', {l_0}', {l_F}', Inv', E')$, *with* $|C| = |C'| = n$, *and* $|V| = |V'| = m$, *we define the* weak reachability equivalence *between* A *and* A', *denoted by* $A \equiv_r A'$, *(relative to a relation* $\approx \subseteq L \times L'$*) if* $\forall\, l \in L\, \forall\, l' \in L'\, \forall \vec{x} \in \mathbb{R}^n_{\geq 0}\ \forall \vec{v} \in D^m$:

1. $(l, \vec{x}, \vec{v}) \in Reach(A) \ \Rightarrow\ \exists\, l' \in L' : l \approx l' \wedge (l', \vec{x}, \vec{v}) \in Reach(A')$.
2. $(l', \vec{x}, \vec{v}) \in Reach(A') \ \Rightarrow\ \exists\, l \in L : l \approx l' \wedge (l, \vec{x}, \vec{v}) \in Reach(A)$.

Theorem 4 (Separation). *Consider ETA A_1 and A_2 as in Definition 9 with memoryless initial locations $l_{01} = l_{02}$. Then $A_1 \cup A_2 \equiv_r A_1 \rhd A_2$.*

In general, $\equiv_r$ is not a congruence w.r.t. parallel composition. Nevertheless, the following theorem states its preservation by parallel instances of separation under the non-interference conditions similar to those of the CCL laws.

Theorem 5 (Separation in parallel context). *For ETA A_1, B_1, A_2, B_2 with memoryless initial locations, with A_2 and B_2 wrapped and satisfying $A_1 \not\frown B_2$ and $B_1 \not\frown A_2$, it holds that $(A_1 \cup [A_2]) \| (B_1 \cup [B_2]) \equiv_r (A_1 \rhd [A_2]) \| (B_1 \rhd [B_2])$.*

The next theorem states that an ETA with a memoryless location l can be *flattened* into one that contains fewer cycles through l, while preserving $\equiv_r$.

Theorem 6 (Flattening). *Consider an ETA $A^* = (L, \Sigma, C, V, l_0, l_F, Inv, E^*)$ with a memoryless location $l \in L$. Then $A_l = (L, \Sigma, C, V, l_0, l_F, Inv, E_l)$, where $E_l = E^* \backslash \{e \mid target(e) = l$ and there is no syntactic path from l_0 to l in E^* such that e is the first edge with $target(e) = l$ on this path$\}$, satisfies $A^* \equiv_r A_l$.*

If $E_l \subset E^*$ then A_l is a *flattened* version of A^* with a reduced number of cycles through l. Note that flattening at l_0 results in A_{l_0} being cycle-free at l_0. Next, we consider flattening of A^* in the context of a parallel composition $A^* \| B$ and state sufficient conditions for the preservation of *location reachability*.

Theorem 7 (Flattening in parallel context). *Suppose that for A^* and B, where A^* is memoryless at $l_a \in L_A$, the following holds within $A^* \| B$:*

1. *Every location of A^* is reachable from its initial location l_{0A} without visiting l_a more than once, while B stays in its initial location l_{0B}.*
2. *Every location of B is reachable from l_{0B}, while A^* stays in l_a.*
3. *If a transition entering l_a enables a transition of B with target l_b then every location of A^* is reachable from l_a without visiting l_a again, while B is in l_b.*

Then $Reachloc(A^* \| B) = Reachloc(A_{l_a} \| B)$.

In contrast to all the other transformations, flattening in a parallel context does not easily generalize to multiple parallel instances (cf. Appendix C of [19]), and the three itemized conditions of Theorem 7 above require an exploration of the reachable state space. Such an exploration however does not entail a complete resolution of the $\|$ operator in $A^* \| B$, as each of the above conditions reduces to a *local reachability check of A^* resp. B*, with control residing at a fixed location of B resp. A^*. For the case where B is itself composed of multiple parallel ETA, a limited resolution of $\|$ within B may be necessary in order to verify the second reachability condition of Theorem 7. The above reachability checks may nonetheless be *localized within a layer* if the flattening is performed subsequent to separation and layering, as will be shown in the next section.

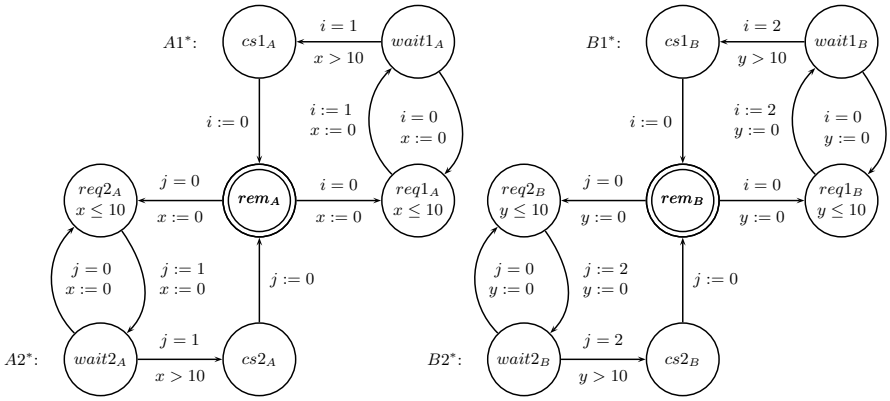

Fig. 1. Double Fischer protocol $DF = (A1^* \cup [A2^*]) \parallel (B1^* \cup [B2^*])$ for processes A and B accessing two critical sections $cs1$ and $cs2$. *Left*: ETA $A1^* \cup [A2^*]$; *right*: ETA $B1^* \cup [B2^*]$. In this and all subsequent figures, we omit the ε-labels of all edges.

5 Example : Real-Time Mutual Exclusion

Consider two processes A and B competing for two critical sections $cs1$ and $cs2$. We safeguard these sections by a *double Fischer protocol* DF obtained by taking for each process the union of two copies of Fischer's real-time protocol.

We represent DF by the following composition of ETA:

$$DF = (A1^* \cup [A2^*]) \parallel (B1^* \cup [B2^*]),$$

where $A1^*, A2^*$ are the two copies of Fischer's protocol used by process A, and $B1^*, B2^*$ the two copies used by process B, cf. Fig. 1. The stars * indicate the presence of cycles. In locations $cs1_A$ and $cs2_A$ process A accesses the critical sections $cs1$ and $cs2$, respectively, and in $cs1_B$ and $cs2_B$ process B does so. Initially, A and B need not access the critical sections and are busy with *remaining* activities in the initial locations rem_A and rem_B, whose conditions permit the wrapping of all ETA, and in particular $A2^*$ and $B2^*$. In $req1_A, req2_A$ and $req1_B, req2_B$ the processes A and B *request* access to $cs1, cs2$. The locations $wait1_A, wait2_A$ and $wait1_B, wait2_B$ represent *waiting* of A and B for $cs1, cs2$. The parallel ETA in DF use disjoint (but synchronous) clocks x and y, while sharing the data variables i and j that range over $0, 1, 2$ and initialized with 0. These values indicate whether (0) none of the processes, (1) process A, or (2) process B wants to access $cs1$ or $cs2$, respectively. In these ETA, all edges are labelled by ε-actions. Synchronization between the ETA takes place via guards checking the values of the shared variables i and j.

We wish to prove that DF satisfies the *double mutual exclusion* property

$$DMX = \Box\neg(cs1_A \wedge cs1_B) \wedge \Box\neg(cs2_A \wedge cs2_B).$$

We simplify the verification task now by a series of structural transformations so that it finally becomes almost trivial.

1. Separation. We apply the separation transformation to DF and obtain two single versions of Fischer's protocol separated by an extra transition, cf. Fig. 2. This is possible due to $A1^* \not\frown B2^*$ and $B1^* \not\frown A2^*$. The result is

$$SDF = (A1^* \rhd [A2^*]) \parallel (B1^* \rhd [B2^*]).$$

By the Separation Theorem 5, we have $DF \equiv_r SDF$.

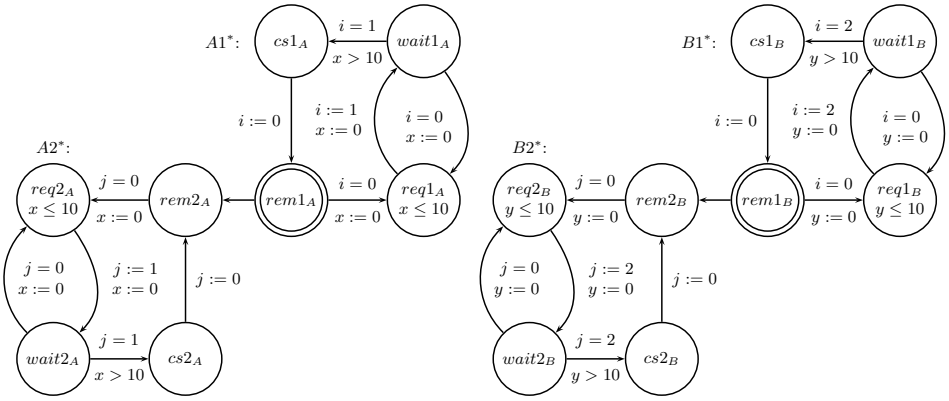

Fig. 2. Separated version of double Fischer: $SDF = (A1^* \rhd [A2^*]) \parallel (B1^* \rhd [B2^*])$. The layered version LDF is obtained from SDF by cutting the two component ETA of SDF at $rem2_A$ and $rem2_B$, yielding $LDF = (A1^* \parallel B1^*) \rhd ([A2^*] \parallel [B2^*])$.

2. Layering. To apply layering to SDF, we calculate:

$$\begin{aligned} SDF = \quad & (A1^* \rhd [A2^*]) \parallel (B1^* \rhd [B2^*]) \\ \equiv_L \quad & \{ \text{ Corollary 1, using } A1^* \not\frown B2^* \text{ and } B1^* \not\frown A2^* \} \\ & (A1^* \parallel B1^*) \rhd ([A2^*] \parallel [B2^*]) \quad = \quad LDF \end{aligned}$$

LDF stands for layered double Fischer. The component ETA of LDF are obtained from the ETA shown in Fig. 2 by cutting these at $rem2_A$ and $rem2_B$.

To prove that DF satisfies DMX, it suffices to do this for LDF. Since step composition is the top operator in LDF, it suffices to show that both step components, $A1^* \parallel B1^*$ and $[A2^*] \parallel [B2^*]$, individually satisfy DMX.

3. Flattening. We remove all cycles in $A1^*, B1^*, A2^*, B2^*$ and show this in detail for $A1^*$ in Fig. 3. Flattening $A1^*$ at $req1_A$ is possible, since whenever $req1_A$ is entered, $i = 0$ and $x = 0$ holds. Flattening the resulting $A1_{req1_A}$ at $rem1_A$ does not seem possible at first sight because only the data variable i is reset to 0. However, we may safely add $x := 0$ because this clock reset occurs when $rem1_A$ is left. Note also that the additional three conditions of Theorem 7 hold for $A1^*$ at the locations $req1_A$ and $rem1_A$. Hence, we arrive at $A1 = A1_{req1_A, rem1_A}$ without any cycles. We may similarly flatten $B1^*$ at

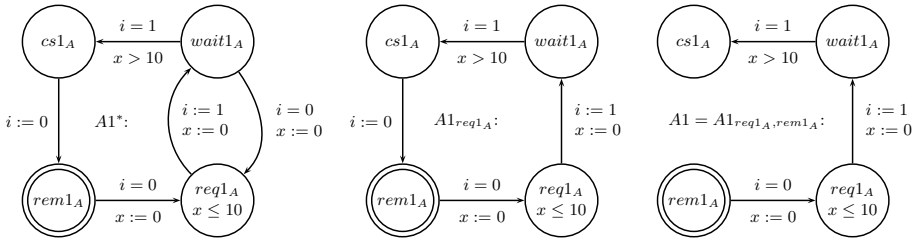

Fig. 3. Flattening $A1^*$ at location $req1_A$ yields $A1_{req1_A}$ and flattening this ETA at location $rem1_A$ yields the cycle-free ETA $A1 = A1_{req1_A, rem1_A}$.

$req1_B$ and $rem1_B$, yielding a corresponding cycle-free ETA $B1$. It thus remains to show that two cycle-free versions of Fischer's protocol, $A1 \parallel B1$ and $A2 \parallel B2$, individually satisfy *DMX*, where we have, for $i \in \{1, 2\}$, $Reachloc(Ai \| Bi) = Reachloc(Ai^* \| Bi^*)$ by Theorem 7, which is sufficient for preserving *DMX*.

4. Timed Layering. We prove that $A1 \| B1$ satisfies *DMX* (and symmetrically for $A2 \| B2$). Consider the ETA $A_{01}, A_{11}, A_{21}, B_{01}, B_{11}, B_{21}$ shown in Fig. 4. As before, x and y are clocks, and i is a shared data variable ranging over 0, 1, 2, initialized with 0. The ETA A_{01}, A_{11}, A_{21} represent three phases of $A1$ and B_{01}, B_{11}, B_{21} those of $B1$, such that $A1 \| B1 = (A_{01}; A_{11}; A_{21}) \parallel (B_{01}; B_{11}; B_{21})$.

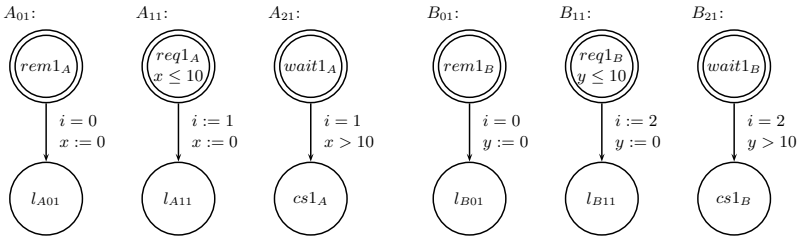

Fig. 4. Six ETA for building Fischer's protocol for single mutual exclusion of two processes: $A1 = A_{01}; A_{11}; A_{21}$ and $B1 = B_{01}; B_{11}; B_{21}$.

We explore the interleavings of $A_{01}; A_{11}; A_{21}$ with $B_{01}; B_{11}; B_{21}$ by (partially) *expanding* (as in CCS, cf. [16]) the parallel composition in $A1 \| B1$. After A_{11} neither B_{01} nor B_{21} can occur due to the i-guard, and vice versa, after B_{11} neither A_{01} nor A_{21} can occur. After $A_{01} \| B_{01}$ we observe that timewise (by the synchronous evolution of the clocks x and y) B_{21} cannot proceed from $wait1_B$ to $cs1_B$ (due to the clock guard $y > 10$) before A_{11} has left $req1_A$ (with clock invariant $x \leq 10$) to reach its final location l_{A11}, and vice versa, A_{21} cannot proceed to $cs1_A$ before B_{11} reaches its final location l_{B11}. So the time precedences $A_{11} \prec B_{21}$ and $B_{11} \prec A_{21}$ hold. Thus expansion and Corollary 2 yield

$$\begin{aligned} A1 \| B1 &= (A_{01}; A_{11}; A_{21}) + (B_{01}; B_{11}; B_{21}) + (A_{01} \| B_{01}); ((A_{11}; A_{21}) \| (B_{11}; B_{21})) \\ &\equiv (A_{01}; A_{11}; A_{21}) + (B_{01}; B_{11}; B_{21}) + (A_{01} \| B_{01}); (A_{11} \| B_{11}); (A_{21} \| B_{21}) = SF, \end{aligned}$$

where + denotes a non-deterministic choice operator on ETA (cf. Definition 18 in Appendix A of [19]), and SF stands for sequential Fischer. Clearly, SF satisfies DMX because after $A_{11} \| B_{11}$ the data variable i stores either 1 or 2, and thus either A_{21} or B_{21} (but not both) can proceed to their critical section. Since each of the equivalences induced by our transformations (namely, $\equiv$, $\equiv_r$, $\equiv_L$, and equality w.r.t *Reachloc*) is clearly sufficient for DMX, we then conclude that DMX holds for DF as required.

6 Related Work

We now discuss related transformational approaches in the literature.

A constraint-based decompositional proof methodology was illustrated in [14] on the standard Fischer's protocol, formalized as a *timed modal specification*. More recently, an analysis of TA networks with "disjoint phases of activity" has been carried out in [18], where it has been shown that the parallel composition of two TA (without shared data variables) is bismilar to their sequential composition, if the TA exhibit certain *periodic but non-overlapping* behaviours.

In [8] it was shown that any TA (possibly containing nested cycles, but again without shared data variables) may be transformed into one that is flat (in the sense that each location is part of at most one cycle), while preserving the reachability relation between states. Their (non-local) transformation, while applicable to all TA, is however not preserved in the context of parallel composition, and suffers from an exponential blow-up in the number of locations in the resulting flattened TA, cf. Lemma 3 of [8]. Our (local) separation and flattening transformations, on the other hand, are applicable (in the context of parallel composition) to the data-enriched setting of ETA networks, and maintain the same number of locations, while reducing the nesting depth and deleting those transitions that (re-)enter memoryless locations, cf. Theorems 5 and 7.

A layered transformation for distributed algorithms with (predominantly synchronous) *message passing* was presented in [21]. *Round-based communication closedness* was considered in [6] for fault-tolerant distributed algorithms with *asynchronous message passing*, with messages being considered only in the rounds during which they were sent. Consensus algorithms in such a setting were then brought under the scope of automatic verification, by means of "reduction theorems", cf. [6]. Layered transformations for randomized distributed algorithms (modelled as compositions of probabilistic automata with shared data variables) have been recently investigated by the authors in [22].

Acknowledgements. A. Kupriyanov and the reviewers gave insightful feedback.

References

1. Alur, R., Dill, D.: A theory of timed automata. TCS (2), 183–235 (1994)
2. Alur, R., Brayton, R.K., Henzinger, T.A., Qadeer, S., Rajamani, S.K.: Partial-order reduction in symbolic state-space exploration. FMSD 18, 97–116 (2001)

3. Behrmann, G., David, A., Larsen, K.G.: A tutorial on Uppaal. In: Formal Methods for the Design of Real-Time Systems. LNCS, vol. 3185, pp. 200–236. Springer (2004)
4. Bengtsson, J., Jonsson, B., Lilius, J., Yi, W.: Partial order reductions for timed systems. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR. LNCS, vol. 1466, pp. 485–500. Springer (1998)
5. Bouyer, P., Petit, A.: Decomposition and composition of timed automata. In: Wiedermann, J., van Emde Boas, P., Nielsen, M. (eds.) ICALP. LNCS, vol. 1644, pp. 210–219. Springer (1999)
6. Chaouch-Saad, M., Charron-Bost, B., Merz, S.: A reduction theorem for the verification of round-based distributed algorithms. In: Bournez, O., Potapov, I. (eds.) Reachability Problems (RP). LNCS, vol. 5797, pp. 93–106. Springer (2009)
7. Cohen, E.: Separation and reduction. In: Backhouse, R.C., Oliveira, J.N. (eds.) Mathematics of Program Construction. LNCS, vol. 1837, pp. 45–59. Springer (2000)
8. Comon, H., Jurski, Y.: Timed automata and the theory of real numbers. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR. LNCS, vol. 1664, pp. 242–257. Springer (1999)
9. Dong, J.S., Hao, P., Qin, S., Sun, J., Yi, W.: Timed automata patterns. IEEE Trans. Software Eng. 34(6), 844–859 (2008)
10. Elrad, T., Francez, N.: Decomposition of distributed programs into communication-closed layers. Sci. Comput. Program. 2, 155–173 (1982)
11. Haakansson, J., Pettersson, P.: Partial order reduction for verification of real-time components. In: Raskin, J.F., Thiagarajan, P.S. (eds.) FORMATS. LNCS, vol. 4763, pp. 211–226. Springer (2007)
12. Janssen, W.: Layered Design of Parallel Systems. Ph.D. thesis, U. Twente (1994)
13. Janssen, W., Poel, M., Xu, Q., Zwiers, J.: Layering of real-time distributed processes. In: Langmaack, H., de Roever, W.P., Vytopil, J. (eds.) FTRTFT. LNCS, vol. 863, pp. 393–417. Springer (1994)
14. Larsen, K.G., Steffen, B., Weise, C.: Fischer's protocol revisited: A simple proof using modal constraints. In: Alur, R., Henzinger, T.A., Sontag, E.D. (eds.) Hybrid Systems. LNCS, vol. 1066, pp. 604–615. Springer (1996)
15. Lugiez, D., Niebert, P., Zennou, S.: A partial order semantics approach to the clock explosion problem of timed automata. Theor. Comput. Sci. 345, 27–59 (2005)
16. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
17. Minea, M.: Partial order reduction for model checking of timed automata. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR. LNCS, vol. 1664, pp. 431–436. Springer (1999)
18. Muniz, M., Westphal, B., Podelski, A.: Timed automata with disjoint activity. In: Jurdzinski, M., Nickovic, D. (eds.) FORMATS. LNCS, vol. 7595, pp. 188–203. Springer (2012)
19. Olderog, E.R., Swaminathan, M.: Structural transformations for data-enriched real-time systems. Tech. Rep. 90, Reports of SFB/TR 14 AVACS (2013), see `www.avacs.org`
20. Olderog, E.R., Swaminathan, M.: Layered composition for timed automata. In: Chatterjee, K., Henzinger, T.A. (eds.) FORMATS. LNCS, vol. 6246, pp. 228–242. Springer (2010)
21. Stomp, F.A., de Roever, W.P.: A principle for sequential reasoning about distributed algorithms. Formal Asp. Comput. 6(6), 716–737 (1994)
22. Swaminathan, M., Katoen, J.P., Olderog, E.R.: Layered reasoning for randomized distributed algorithms. Formal Asp. Comput. 24, 477–496 (2012)