Form Methods Syst Des
DOI 10.1007/s10703-012-0179-8

Event clock message passing automata: a logical characterization and an emptiness checking algorithm

S. Akshay · Benedikt Bollig · Paul Gastin

Abstract We are interested in modeling behaviors and verifying properties of systems in which time and concurrency play a crucial role. We introduce a model of distributed automata which are equipped with event clocks as in Alur et al. (Theor Comput Sci 211:253–273, 1999), which we call Event Clock Message Passing Automata (ECMPA). To describe the behaviors of such systems we use timed partial orders (modeled as message sequence charts with timing).

Our first goal is to extend the classical Büchi-Elgot-Trakhtenbrot equivalence to the timed and distributed setting, by showing an equivalence between ECMPA and a timed extension of monadic second-order (MSO) logic. We obtain such a constructive equivalence in two different ways: (1) by restricting the semantics by bounding the set of timed partial orders; (2) by restricting the timed MSO logic to its existential fragment. We next consider the emptiness problem for ECMPA, which asks if a given ECMPA has *some* valid timed execution. In general this problem is undecidable and we show that by considering only bounded timed executions, we can obtain decidability. We do this by constructing a timed automaton which accepts all bounded timed executions of the ECMPA and checking emptiness of this timed automaton.

Keywords Message passing automata · Timed automata · MSO logic · Message sequence charts

1 Introduction

In today's world, we encounter computational devices all around us. These devices do not act in isolation but interact in increasingly complex ways. For example, Automatic Teller Ma-

S. Akshay (✉) · B. Bollig · P. Gastin
Laboratoire de Specification et Verification, ENS Cachan and CNRS, 61, Avenue du President Wilson, 94235 Cachan Cedex, France
e-mail: akshay@irisa.fr

B. Bollig
e-mail: Benedikt.Bollig@lsv.ens-cachan.fr

P. Gastin
e-mail: Paul.Gastin@lsv.ens-cachan.fr

Published online: 08 December 2012

chines (ATMs), online banking systems, car braking systems, railway gate controllers are all composed of several components that communicate with each other over an extended period of time. A common factor in many such systems is the interplay between concurrency and timing. Concurrency plays an important role since systems usually consist of independent components that interact periodically to coordinate their behavior. Timing considerations play an important role in describing how these interactions proceed.

Our goal is to use formal methods to reason about systems where time and concurrency play a significant role. The first challenge is to choose a suitable formalism that admits automated analysis. For instance, if a system exhibits regular, finite-state behavior, we can use model checking to efficiently explore the state space and determine various behavioral properties. Finite-state automata provide an intuitively appealing machine model for generating regular behaviors in this setting. The regular behaviors can be represented as a set of words over an alphabet. Monadic second order logic is an elegant language to describe abstract properties of sets of words. The Büchi-Elgot-Trakhtenbrot Theorem [10, 16] links the two formalisms: a behavior can be described by a finite-state automaton if and only if it can be expressed in monadic second order logic. This correspondence is effective and forms the basis for model checking behavioral properties of finite-state systems. We would like to lift this approach to the timed and distributed setting.

In the timed but sequential setting, event clock automata are a well-known formalism, which have nice properties and describe timed behaviors as timed words. An event clock automaton uses implicit "event clocks" that record or predict time lapses with respect to the last or the next occurrence of an event. In [14], it was proved that a timed language, i.e., a set of timed words, can be recognized by an event clock automaton if and only if it can be defined in a timed version of MSO logic. On the other hand, in the concurrent setting, message passing automata (MPA) form a natural machine model for distributed systems. MPA comprise of several finite state automata communicating via FIFO channels and their runs can be described as Message sequence charts, which are labeled partial orders (over a labeling alphabet of sends and receives). In this case, it was proved in [17, 19] that with different restrictions on the MSC languages (which will be made precise later), such an MSC can be described by a MPA if and only if it can be expressed in a MSO logic over MSCs. In [7], this result was proved without any restriction on the language of MSCs by considering only the existential fragment of the MSO logic.

In this work, we unify the above approaches by defining a machine model for timed and concurrent systems, namely the Event clock message passing automata (event clock MPA). Event clock MPA are message passing automata which are equipped with event clocks (as in the event clock automaton in the sequential timed setting) to reason about timed and concurrent behavior. To describe the behavior of such automata, we generalize MSCs to their timed extension, which we do in two ways. We first consider timed MSCs which are just MSCs with time-stamps at events (as in timed words). These are ideal to describe real-time system executions, while keeping the causal relation between events explicit. Next, we consider MSCs with timing constraints or time-constrained MSCs, where we associate time-intervals to some pairs of events (instead of attaching time-stamps to individual events). The endpoints of the interval give us the upper and lower bounds on the time allowed to elapse between the events.

Results on event clock message passing automata Our first result is to lift the Büchi-Elgot equivalence [10, 16] to the timed and distributed setting. For the logical framework, we use a timed version of MSO logic. We interpret both event clock MPAs and timed MSO formulae over timed MSCs and prove a constructive equivalence between them, with and

without restriction on the MSC languages. Thus, we provide a logical characterization for event clock MPA. This is done by lifting the corresponding results from the untimed case [7, 17, 19]. An important intermediary step in this translation is the reinterpretation of the timed MSO and event clock MPA in terms of time-constrained MSCs rather than timed MSCs. The time-constrained MSCs provide a dual link: they can be seen as MSCs whose labelings are extended by timing information and they can also be seen as a representation of infinite sets of timed MSCs. Once this translation is done, we can essentially follow the technique of [14] where such an equivalence is shown for (sequential) timed words.

Next, we consider the emptiness problem, which is one of the most basic verification questions that one may wish to ask. In our setting, the emptiness problem for event clock MPAs asks if a given event clock MPA has any run, i.e., a timed MSC which is accepted by it. It can be easily seen that without any restriction on the timed MSCs, this problem is undecidable, since this is already the case in the untimed setting. However we prove that if we restrict to timed MSCs with at least one *bounded* linearization (i.e., in which any send and its matching receive are apart by at most K events), then the emptiness problem is decidable. Indeed, this condition is subsumed by one of restricted settings under which we proved the above equivalence result. Therefore, as a corollary, we also obtain that the satisfiability problem for our logic is decidable. Our approach to prove decidability consists of constructing a global finite timed automaton that can simulate the runs of an event clock MPA (which is a distributed machine) and so, reduce the problem to checking emptiness for a timed automaton. The hard part of the construction lies in "cleverly" maintaining the partial-order information (of the timed MSC) along the sequential runs of the global timed automaton, while using only finitely many clocks.

Related work Providing a timed partial order semantics as we have done above allows us to apply partial order reduction techniques [18] to address the model-checking problem. But indeed, there are several other models that also handle time and concurrency in a comparable way. Many timed extensions of Petri nets have been considered, for instance, time Petri nets [6], timed Petri nets [23]. Unfoldings of Petri nets provide a way to model the partial order behavior of these systems and by lifting these unfoldings to the timed extensions, they provide a timed partial order semantics [12]. However, these unfoldings are seldom graphically representable in a compact manner unlike MSCs (and their timed extensions). Further, unfoldings in Petri nets correspond to "branching time" whereas MSCs express "linear time" behavior. Other models dealing with time and concurrency include networks of timed automata [2] and products of timed automata [13]. In [8], unfolding techniques were applied to study such networks of timed automata. However, none of these models allow communication via explicit message passing, which is one of the main features of the event clock MPA that we have introduced.

The formal semantics and analysis of timing in MSCs has been addressed earlier in [4, 5, 11, 20]. In [4] and [5], only single timed MSCs or high-level timed MSCs were considered, while in [20] one of the first models of timed MPAs was introduced. However, the latter do not consider MSCs as semantics but rather look at restricted channel architectures (e.g., one-channel systems) to transfer decidability of reachability problems from the untimed to timed setting. The automaton model in [11] links the two approaches by considering a similar automaton model with semantics in terms of timed MSCs and proposes a practical solution to a very specific matching problem using the tool UPPAAL.

A preliminary version of the results in this paper was presented in [1].

2 Preliminaries

We denote by $\mathbb{N} = \{0, 1, 2, \ldots\}$ the set of natural numbers. For an alphabet Σ, a *Σ-labeled poset* is a structure $(E, \leq, \lambda)$ over Σ, where E is a set of events, $\leq$ is a partial order on the set of events E called its *ordering relation* and $\lambda : E \to \Sigma$ is the labeling function. A *linearization* of a Σ-labeled poset $(E, \leq, \lambda)$ is any Σ-labeled poset $(E, \leq', \lambda)$ such that $\leq'$ is a linear extension of $\leq$. Then, the set of events $E = \{e_1, \ldots, e_n\}$ can be rewritten as a sequence $e_{i_1} \leq' e_{i_2} \leq' \cdots \leq' e_{i_n}$ s.t., $\lambda(e_{i_1}) \ldots \lambda(e_{i_n}) \in \Sigma^*$. Thus, any linearization of $(E, \leq, \lambda)$ can be identified with a unique word over Σ.

Message sequence charts (MSCs) Let $Proc = \{p, q, r, \ldots\}$ be a non-empty finite set of *processes (agents)* that communicate through messages via reliable FIFO channels using an alphabet of *messages* $\mathcal{M}$. For $p \in Proc$, let $Act_p = \{p!q(m), p?q(m) \mid q \in Proc, q \neq p, m \in \mathcal{M}\}$ be the set of *communication actions of process p* . The action $p!q(m)$ is read as *p sends the message m to q* and the action $p?q(m)$ is read as *p receives the message m from q*. We set $Act = \bigcup_{p \in Proc} Act_p$. We also denote the set of *channels* by $Ch = \{(p, q) \in Proc \times Proc \mid p \neq q\}$. For an action $a \in Act$, we will sometimes write $a \in p!q$ (respectively, $a \in p?q$) to denote $a = p!q(m)$ (respectively, $p?q(m)$) for some $m \in \mathcal{M}$.

Let $M = (E, \leq, \lambda)$ be an Act-labeled partial order. For $e \in E$, let $\downarrow e = \{e' \in E \mid e' \leq e\}$. For $X \subseteq E$, let $\downarrow X = \bigcup_{e \in X} \downarrow e$. We call $X \subseteq E$ a *prefix* of M if $X = \downarrow X$. For $p \in Proc$ and $a \in Act$, we set $E_p = \{e \in E \mid \lambda(e) \in Act_p\}$ to be the set of all p-events and $E_a = \{e \in E \mid \lambda(e) = a\}$ to be the set of a-events. For each $(p, q) \in Ch$, we define a relation $<_{pq}$ as follows, to capture the fact that channels are FIFO with respect to each message—if $e <_{pq} e'$, the message m read by q at e' is the one sent by p at e.

$$e <_{pq} e' \quad \text{if } \exists m, \lambda(e) = p!q(m),\ \lambda(e') = q?p(m) \text{ and } |\downarrow e \cap E_{p!q(m)}| = |\downarrow e' \cap E_{q?p(m)}|$$

Finally, for each $p \in Proc$, we define the relation $\leq_{pp} = (E_p \times E_p) \cap \leq$, with $<_{pp}$ standing for the largest irreflexive subset of $\leq_{pp}$. Also, $\lessdot_{pp}$ denotes the *immediate successor relation* on process p: for $e, e' \in E_p$, $e \lessdot_{pp} e'$ if $e <_{pp} e'$ and for all $e'' \in E_p$, we have $e <_{pp} e'' \leq_{pp} e'$ implies $e'' = e'$.

Definition 2.1 A *message sequence chart (MSC)* over Act is a *finite* Act-labeled poset $M = (E, \leq, \lambda)$ such that:

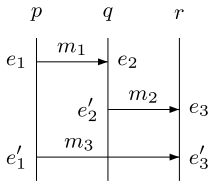
1. Each relation $\leq_{pp}$ is a total order on E_p.
2. If $p \neq q$ then for each $m \in \mathcal{M}$, $|E_{p!q(m)}| = |E_{q?p(m)}|$.
3. If $e <_{pq} e'$, then

$$\left|\downarrow e \cap \left(\bigcup_{m \in \mathcal{M}} E_{p!q(m)}\right)\right| = \left|\downarrow e' \cap \left(\bigcup_{m \in \mathcal{M}} E_{q?p(m)}\right)\right|$$

4. The partial order $\leq$ is the reflexive, transitive closure of $\bigcup_{p,q \in Proc} <_{pq}$.

As each *linearization* of an MSC $M = (E, \leq, \lambda)$ over Act corresponds to a *word over Act*, we will use these notions interchangeably. Under the FIFO assumption an MSC can be reconstructed, up to isomorphism, from any of its linearizations. In diagrams, the events of an MSC are presented in *visual order*. Events of a process are arranged in a vertical line and messages are displayed as horizontal or downward-sloping directed edges. Figure 1 shows an example of an MSC with three processes $\{p, q, r\}$ and six events $\{e_1, e_1', e_2, e_2', e_3, e_3'\}$

Fig. 1 An MSC



corresponding to three messages—m_1 from p to q, m_2 from q to r and m_3 from p to r. Then, for instance, $p!q(m_1)\ q?p(m_1)\ q!r(m_2)\ p!r(m_3)\ r?q(m_2)\ r?p(m_3)$ is one linearization or *execution* of this MSC seen as a word over Act.

An *MSC language over Act* is a set of MSCs over Act. An important subclass of MSCs is the set of *bounded* MSCs that correspond to systems whose channel capacity is restricted. These systems turn out to enjoy nice algorithmic properties and have liberal logical correspondences too. Let $B \in \mathbb{N}_{>0}$ be a positive integer. Then, a word $w \in Act^*$ is said to be B-bounded if for each prefix u of w and any $p, q \in Proc$, the number of occurrences of $p!q$ exceeds the number of occurrences of $q?p$ by at most B. This means that along the sequential execution of M described by w, no channel ever contains more than B-messages.

Definition 2.2 An MSC M is called *universally B-bounded* (or $\forall$-B-bounded) if every linearization of M is B-bounded. It is said to be *existentially B-bounded* (or $\exists$-B-bounded) if there exists a linearization which is B-bounded.

A set of MSCs is said to be *$\exists$-B-bounded (respectively, $\forall$-B-bounded)* if each MSC in the set is $\exists$-B-bounded (respectively, $\forall$-B-bounded). Further such a set is called *existentially bounded (respectively, universally bounded)* if there exists a B such that it is $\exists$-B-bounded (respectively, $\forall$-B-bounded). As shown in [19], any regular MSC language (i.e., the set of all possible linearizations is regular) is universally bounded.

3 Formalisms to describe timed and concurrent behaviors

The first natural attempt while trying to add timing information to MSCs would be to add time stamps to the events of the MSCs. This is motivated from timed words where we have words with time stamps added at each letter (action). This approach is quite realistic when we want to model the real-time execution of concurrent systems.

Definition 3.1 A *timed MSC* (TMSC) over Act is a pair (M, t) where $M = (E, \leq, \lambda)$ is an MSC over Act and $t : E \to \mathbb{R}_{\geq 0}$ is a function such that if $e \leq e'$ then $t(e) \leq t(e')$ for all $e, e' \in E$. The set of all TMSCs over Act is denoted $\mathbb{TMSC}(Act)$.

In the above definition note that over events e, e', $\leq$ is the partial order relating events, while over reals $t(e), t(e')$, $\leq$ refers to the usual ordering between real numbers.

Example 3.1 Consider the TMSC T presented in Fig. 2 which shows a scenario where two user processes p and r interact with a railway ticket-booking server process q to book the last available ticket for a certain train journey. Actions are of the form $p!q(req)$ meaning that *User1* sends the *Server* a request for a ticket. In the scenario shown, both users send booking requests to the *Server* at time instant 1. The *Server*

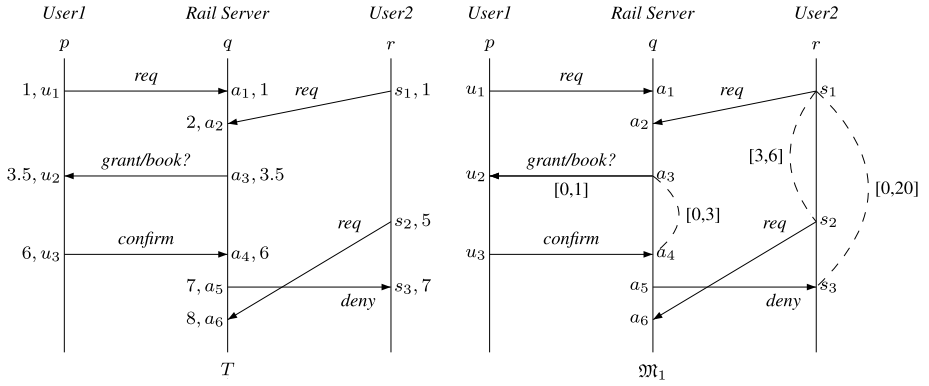


Fig. 2 (**a**) A timed MSC T and (**b**) a TCMSC $\mathfrak{M}_1$ describing the interaction of two users with a server

grants *User1*'s request since it is received first. Then, *User1* confirms his/her booking which leads to the server denying the booking to *User2*. Meanwhile, *User2* repeats her request at instant 5 which reaches the *Server* at time 8. An *execution* or *linearization* of this scenario is: $(p!q(req),1)(r!q(req),1)(q?p(req),1)(q?r(req),2)(q!p(grant),3.5)$ $(p?q(grant),3.5)(r!p(req),5)(p!q(conf),6)(q?p(conf),6)(q!r(deny),7)(r?q(deny),7)$ $(q?r(req),8)$.

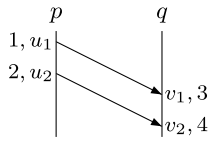
Note that in the above execution, the ordering on the time stamps is preserved, thus making it a timed word over the alphabet of actions. Such an execution is called a *timed linearization* of T. A single TMSC might have more than one timed linearization if concurrent events on different processes have the same time stamp. For instance, if we swap the first two pairs in the above execution we obtain another execution which respects the time stamps. Finally, observe that not all linearizations are timed linearizations as seen by swapping the last two pairs in the above execution.

Formally, let $T = (M, t)$ be a TMSC over Act with $M = (E, \leq, \lambda)$. Consider a linearization $(E, \leq', \lambda)$ of $(E, \leq, \lambda)$ according to which the events of E can be written as a sequence $e_1 \leq' e_2 \leq' \cdots \leq' e_n$. Then $(E, \leq' \lambda, t)$ is said to be a *linearization* of TMSC T. If in addition, for all $1 \leq j < k \leq n$, we have $t(e_j) \leq t(e_k)$, then it is said to be a *timed linearization*. Indeed, this can be seen as a timed word σ over Act by uniquely identifying it with $\sigma = (\lambda(e_1), t(e_1)) \ldots (\lambda(e_n), (t(e_n)))$. We let TW_{Act} denote the set of timed words over Act and t-lin$(T) \subseteq \mathrm{TW}_{Act}$ denote the set of timed linearizations of a TMSC T, seen as a timed word language over Act. As with untimed MSCs, a timed MSC can be faithfully reconstructed from any of its timed linearizations under FIFO assumption on channels. A *TMSC language $\mathcal{L}$ over Act* is a set of TMSCs over Act. Then, t-lin$(\mathcal{L})$ is the timed word language over Act consisting of all timed linearizations of TMSCs in $\mathcal{L}$, i.e., t-lin$(\mathcal{L}) = \bigcup\{\text{t-lin}(T) \mid T \in \mathcal{L}\}$.

Bounded channel setting As in the case of untimed MSCs, restricting the channel capacity in TMSCs gives rise to an interesting, more "tractable" subclass of TMSCs. We extend the definition of existential and universal bounds from MSCs to TMSCs.

Definition 3.2 A TMSC (M, t) is called *untimed-existentially-B-bounded* ($\exists^u$-B-bounded) if the MSC M is $\exists$-B-bounded. Similarly (M, t) is *untimed-universally-B-bounded* ($\forall^u$-B-bounded) if M is $\forall$-B-bounded.

Fig. 3 TMSC T'



Note that an existential *untimed* bound may not be achievable by a timed linearization. For instance, consider the TMSC T' in Fig. 3. T' is $\exists^{u}$-1-bounded as there exists a linearization of the untimed MSC which is 1-bounded, namely, $u_1v_1u_2v_2$. However, this does not correspond to a timed linearization. In fact, the only timed linearization of T, namely $(u_1, 1)(u_2, 2)(v_1, 3)(v_2, 4)$, is 2-bounded.

Message sequence charts with timing constraints TMSCs (and indeed timed linearizations) essentially capture only the operational/global behavior of distributed systems. In some sense, this is due to the fact that TMSCs and their timed linearizations are not very different. In other words, by attaching time-stamping to events of an MSC, we lose much of its partial order information. The only partial-order information retained is through events on different processes with the same time-stamps. The remaining are totally ordered due to the global time-stamping.

A richer partial order behavior can be retained by attaching *timing constraints* to pairs of events of the MSC. This approach has two other major advantages:

- Firstly, from a specification point of view, it allows the specifier to decide and enforce constraints between occurrences of events as he chooses.
- Secondly, a single MSC with timing constraints can describe a whole family of TMSCs (with the same underlying MSC) thus being a much more succinct description of the timed behaviors of a system.

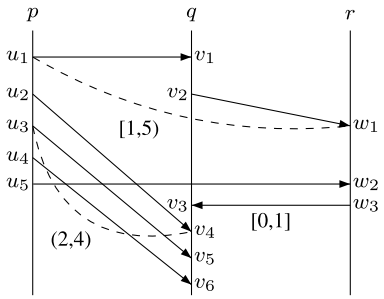
For instance, consider Example 3.1 again where two users interact with a railway booking server. It might be that after granting a *User* request, the *Server* waits only for a bounded amount of time for him/her to respond before canceling the request. The family of TMSCs satisfying such a constraint is easier to capture using a scenario with timing constraints as shown in $\mathfrak{M}_1$ of Fig. 2. The label $[0, 3]$ from a_3 to a_4 specifies that *User1* must respond to the grant within 3 time units. The label $[0, 1]$ on the message from a_3 to u_2 specifies the bounds on the delay of message delivery and so on.

In the above example we can, a priori, define timing constraints between *any* two distinct but arbitrary events. But is this really what we want? In fact, this might defeat our purpose for introducing timing constraints in MSCs in the first place. For instance, in the TCMSC $\mathfrak{M}_1$ in Fig. 2, should a specifier be allowed to have a *choice* of imposing constraints between the first event of User1 and the first event of User2? Or, from an implementation point of view, can a machine that implements such a constraint really be called a distributed machine?

In other words, the vital question is how flexible we want this timing to be, i.e., between which pairs of events we allow constraints. To define this formally, we fix an MSC $M = (E, \leq, \lambda)$ over Act and define the following relations that will relate the pairs of events on which we wish to impose timing constraints. First, we denote the set of all irreflexive pairs of E by $\mathrm{P}(E) = \{(e, e') \in E \times E \mid e \neq e'\}$. Then,

- The *message relation*, which is defined as
  $\mathrm{Msg}^M = \{(e, e') \in \mathrm{P}(E) \mid e <_{pq} e' \text{ for some } (p, q) \in Ch\}$
- The *previous occurrence of an action* $a \in Act$ is defined as:
  $\mathrm{Prev}_a^M = \{(e, e') \in \mathrm{P}(E) \mid \lambda(e') = a, e' \leq e \text{ and } \forall e'' \in E, (e'' \leq e \wedge \lambda(e'') = a) \Rightarrow e'' \leq e'\}$

Fig. 4 A TCMSC $\mathfrak{M}_2$



– The *next occurrence of an action* $a \in Act$ is defined as:
$\text{Next}_a^M = \{(e, e') \in \text{P}(E) \mid \lambda(e') = a, e \leq e' \text{ and } \forall e'' \in E, (e \leq e'' \wedge \lambda(e'') = a) \Rightarrow e' \leq e''\}$

Thus, the above relations form a flexible timing formalism, as they allow timing between the next and previous occurrence of any action from an event in the MSC along with the messages. Let $\mathcal{I}$ denote the set of all intervals over the real line with rational end-points.

Definition 3.3 Let $\mathcal{S} \subseteq \mathcal{I}$ be a set of intervals. An *MSC with timing constraints* or a *time-constrained MSC* (denoted *TCMSC*) over $(Act, \mathcal{S})$ is a pair $\mathfrak{M} = (M, \tau)$ where $M = (E, \leq, \lambda)$ is an MSC over Act and τ is a partial map from the set of irreflexive pairs of events $\text{P}(E)$ to the set of intervals $\mathcal{S}$ such that $\text{dom}(\tau) \subseteq \text{Msg}^M \cup \bigcup_{a \in Act}(\text{Next}_a^M \cup \text{Prev}_a^M)$.

With the above definition, TCMSCs can be considered as abstractions of TMSCs and timed words. Here and for the rest of the paper, we let $\mathcal{S} \subseteq \mathcal{I}$ be some fixed set of intervals. Also, when $\mathcal{S} = \mathcal{I}$, i.e., if $\mathfrak{M}$ is a TCMSC over $(Act, \mathcal{I})$, we ignore the latter component and say that $\mathfrak{M}$ is a TCMSC over Act. As usual, when the set of actions is clear from context, we may ignore Act as well. This notion of MSCs with interval constraints on arbitrary pairs of events is in fact similar to the approach adopted by Alur et al. [4]. Thus we can use their MSC analysis tool to check consistency of the timing constraints in a single TCMSC.

We remark here that the expressiveness of the relations Prev_a, *Msg* and Next_a are in fact incomparable and we cannot always subsume/replace one by the other. We illustrate this by the following example. Consider the TCMSC $\mathfrak{M}_2$ from Fig. 4. We have abstracted away the message contents $\mathcal{M}$ for simplicity. $\mathfrak{M}_2$ has timing constraints defined between the following pairs of events: The pair (w_3, v_3) is a message constraint. The pair (w_1, u_1) is related by $\text{Prev}_{p!q}$. However, note that this constraint can also be seen as between (u_1, w_1) which are related by the $\text{Next}_{r?q}$. In such cases, our definition requires us to have the same interval as a constraint. Finally, the third pair of events (u_3, v_4) are related by the $\text{Next}_{q?p}$ relation. We observe that this pair of events is not related by *Msg* or Prev_a relation for any $a \in Act$. Similarly, the pair of events (u_3, v_5) can be timed only by a message constraint. And again timing is allowed between the pair (v_5, u_4) only because of the $\text{Prev}_{p!q}$ relation. Thus Prev_a, *Msg*, and Next_a are expressively incomparable.

Definition 3.4 Let $\mathfrak{M} = (M, \tau)$ be a TCMSC over Act with $M = (E, \leq, \lambda)$. A TMSC $T = (M, t)$ is said to *realize* $\mathfrak{M}$ if for all $(e_1, e_2) \in \text{dom}(\tau)$ we have $|t(e_2) - t(e_1)| \in \tau(e_1, e_2)$. The set of all TMSCs that realize $\mathfrak{M}$ is denoted $\mathcal{L}_{time}(\mathfrak{M})$.

For instance, the TMSC T of Fig. 2 realizes the TCMSC $\mathfrak{M}_1$ from Fig. 2. Now, if all possible *allowed* pairs have explicit timing constraints defined, then we call such a TCMSC

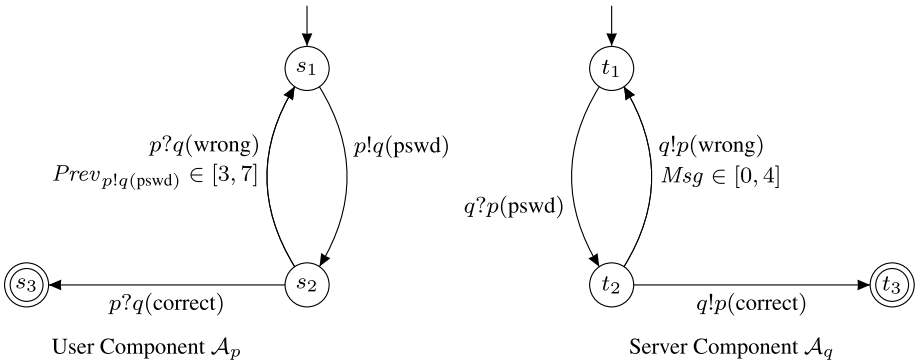


Fig. 5 An ECMPA $\mathcal{A}_1$

maximally defined. That is, a TCMSC $\mathfrak{M} = (M, \tau)$ over $(Act, \mathcal{S})$ is said to be *maximally defined* if $\text{dom}(\tau) = \text{Msg}^M \cup \bigcup_{a \in Act}(\text{Next}_a^M \cup \text{Prev}_a^M)$.

4 Event clock message passing automata

In this section we introduce our machine model which implements TMSCs as well as TCMSCs. We begin by fixing a formal set TC of symbols as follows:

$$\text{TC} = \{\text{Msg}\} \cup \{\text{Prev}_a \mid a \in Act\} \cup \{\text{Next}_a \mid a \in Act\} \tag{1}$$

When interpreted over a TCMSC $\mathfrak{M} = (M, \tau)$ over Act, each symbol $\alpha \in \text{TC}$ is interpreted as the relation α^M defined in the previous section. We set $\text{TC}^M = \bigcup_{\alpha \in \text{TC}} \alpha^M$ and we let $[\text{TC} \dashrightarrow \mathcal{I}]$ denote the set of partial maps from the set of symbols TC to the set $\mathcal{I}$.

Definition 4.1 An *event clock message passing automaton* (ECMPA) is a tuple $\mathcal{A} = (\{\mathcal{A}_p\}_{p \in Proc}, Act, \Delta, F)$, where

- Δ is a finite set of auxiliary messages;
- Act is the alphabet;
- for each $p \in Proc$, the component $\mathcal{A}_p$ is a structure $(S_p, \rightarrow_p, \iota_p)$ where
  - S_p is a finite set of p-local states,
  - $\iota_p \in S_p$ is the p-local initial state,
  - $\rightarrow_p$ is a *finite* subset of $(S_p \times Act_p \times [\text{TC} \dashrightarrow \mathcal{I}] \times \Delta \times S_p)$;
- $F \subseteq \prod_{p \in Proc} S_p$ is a set of global final states.

With the above definition, ECMPA are extensions of both event clock automata (over timed words) [3] and message passing automata (over MSCs) [9].

Example 4.1 A simple example of an ECMPA is shown in Fig. 5. It describes the interaction between a user and a server, with the server trying to authenticate the user. The user component is denoted $\mathcal{A}_p$ and the server $\mathcal{A}_q$. The set of actions Act consists of: $p!q(pswd)$ (user sends password to server), $q?p(pswd)$ (server receives password), $q!p(correct)$ and $q!p(wrong)$ (server sends appropriate message to user) and $p?q(correct)$, $p?q(wrong)$ (user

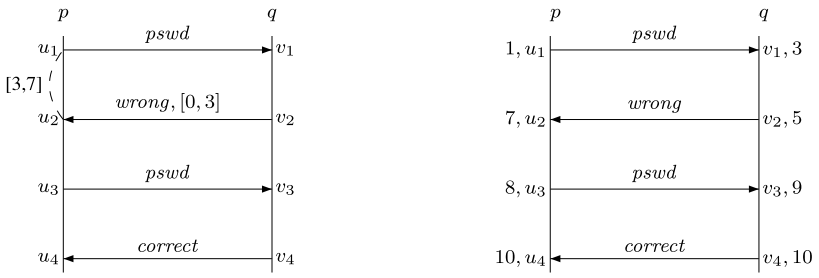


Fig. 6 TCMSC $\mathfrak{M}_3$ and TMSC T_3

receives message from server). Thus, in the above automaton, the user starts by sending its password. If the password received by server is correct, it acknowledges this and goes to final state. Else, it sends message *wrong* which must reach in 4 time units and waits in state t_1. If the user receives *correct* it goes to its final state. If not, it must receive *wrong* in a bounded amount of time since it last sent its password. And in this case, the user tries to resend password. Otherwise, the current interaction is considered void and the run is rejected.

In Fig. 5 the auxiliary data set Δ is a singleton and not depicted for the sake of simplicity. In general, ECMPAs allow every message to be tagged with auxiliary data from a finite set Δ. The ability to convey this finite amount of extra information is quite useful, even in the (untimed) case of message passing automata [9], and it increases the expressive power significantly. For further discussion on message passing automata with and without auxiliary data we refer to the survey at [21].

Semantics over TCMSCs We define the run of an ECMPA $\mathcal{A}$ over a TCMSC $\mathfrak{M} = (M, \tau)$ over *Act*, where $M = (E, \leq, \lambda)$. Consider $r : E \to \bigcup_{p \in Proc} S_p$ labeling each event of process p with a local state from S_p. Define $r^- : E \to \bigcup_{p \in Proc} S_p$ as follows: For event $e \in E_p$, if there is another event $e' \in E_p$ such that $e' \lessdot_{pp} e$, then $r^-(e) = r(e')$ and $r^-(e) = \iota_p$ otherwise. Then r is a *run* of $\mathcal{A}$ on $\mathfrak{M}$ if, for all $e, e' \in E$, with $e <_{pq} e'$ for some channel $(p, q) \in Ch$, there are $g, g' \in [\mathrm{TC} \dashrightarrow \mathcal{I}]$ and an auxiliary message $d \in \Delta$ such that,

- $(r^-(e), \lambda(e), g, d, r(e)) \in \to_p$ and $(r^-(e'), \lambda(e'), g', d, r(e')) \in \to_q$ (2)
- $\forall \alpha \in \mathrm{dom}(g), \exists \widetilde{e} \in E$ s.t., $(e, \widetilde{e}) \in \alpha^M$ and $\tau(e, \widetilde{e}) \subseteq g(\alpha)$ (3)
- $\forall \alpha \in \mathrm{dom}(g'), \exists \widetilde{e}' \in E$ s.t., $(e', \widetilde{e}') \in \alpha^M$ and $\tau(e', \widetilde{e}') \subseteq g'(\alpha)$ (4)

Note that given e, e' as above and α^M, $\widetilde{e}$ and $\widetilde{e}'$ are uniquely defined since α^M is a partial function. We define $s_p = r(e_p)$, where e_p is the maximal event on process p. If there are no events on process p, we set $s_p = \iota_p$. Then run r is *successful* if $(s_p)_{p \in Proc} \in F$. A TCMSC over *Act* is *accepted* by an ECMPA $\mathcal{A}$ if it admits a successful run. We denote by $\mathcal{L}_{TC}(\mathcal{A})$, the set of all TCMSCs over *Act* that are accepted by $\mathcal{A}$. As an example, we may observe that the TCMSC $\mathfrak{M}_3$ from Fig. 6 is accepted by $\mathcal{A}_1$, the ECMPA shown in Fig. 5.

Semantics over TMSCs The semantics of ECMPAs over TMSCs is obtained similarly. The definition of a run of $\mathcal{A}$ over a TMSC $T = (M, t)$ is the same as over a TCMSC $\mathfrak{M} = (M, \tau)$, except that conditions (3) and (4) are respectively replaced by (5) and (6) below:

- for all $\alpha \in \mathrm{dom}(g)$, $\exists \widetilde{e} \in E$ s.t., $(e, \widetilde{e}) \in \alpha^M$ and $\left|t(\widetilde{e}) - t(e)\right| \in g(\alpha)$, (5)
- for all $\alpha \in \mathrm{dom}(g')$, $\exists \widetilde{e}' \in E$ s.t., $(e', \widetilde{e}') \in \alpha^M$ and $\left|t(\widetilde{e}') - t(e')\right| \in g'(\alpha)$. (6)

Then, with the notion of acceptance as above, we can denote the set of all TMSCs accepted by a given ECMPA $\mathcal{A}$ as $\mathcal{L}_{time}(\mathcal{A})$. Again, as an example, the TMSC T_3 from Fig. 6 is accepted by the ECMPA $\mathcal{A}_1$ shown in Fig. 5. We may notice here that T_3 realizes $\mathfrak{M}_3$. This is not a coincidence. In fact, we can make the following general observation.

Lemma 4.1 *Suppose a TMSC T over Act realizes a TCMSC $\mathfrak{M}$ over Act. Then for an ECMPA $\mathcal{A}$, $\mathfrak{M} \in \mathcal{L}_{TC}(\mathcal{A})$ implies $T \in \mathcal{L}_{time}(\mathcal{A})$.*

Proof The lemma follows directly from the definitions. Let $T = (M, t)$ be the TMSC over Act with $M = (E, \leq, \lambda)$ and $\mathfrak{M} = (M, \tau)$. Also let $\mathcal{A} = (\{\mathcal{A}_p\}_{p \in Proc}, Act, \Delta, F)$ from Definition 4.1. Then, any run r of $\mathcal{A}$ on $\mathfrak{M}$ satisfies conditions (2)–(4). Now, since T realizes $\mathfrak{M}$, for all $(e_1, e_2) \in \mathrm{dom}(\tau)$, we have $|t(e_1) - t(e_2)| \in \tau(e_1, e_2)$. This along with (3), implies that for all $\alpha \in \mathrm{dom}(g)$ there exists $\widetilde{e} \in E$ such that $(e, \widetilde{e}) \in \alpha^M$ and $|t(e) - t(\widetilde{e})| \in \tau(e, \widetilde{e}) \subseteq g(\alpha)$. Thus, (5) holds. Similarly, from (4) we obtain (6) and conclude that r is a run of $\mathcal{A}$ on T. As every run of $\mathcal{A}$ on $\mathfrak{M}$ is also a run of $\mathcal{A}$ on T and the acceptance criterion is the same in both cases we conclude that if $\mathfrak{M} \in \mathcal{L}_{TC}(\mathcal{A})$ then $T \in \mathcal{L}_{time}(\mathcal{A})$. □

5 Timed monadic second-order logic

We introduce the logical framework for timed partial orders, which will serve as our specification formalism. As usual, we start with a supply of individual variables $x, y, \ldots$, and set variables $X, Y, \ldots$ which range over events (and sets of events) of the timed MSC. We generalize the usual MSO logic by using (other than unary predicates $P_a(x)$ for $a \in Act$) timing predicates of the form $\delta_\alpha(x) \in I$ for a variable x, $\alpha \in \mathrm{TC}$, and $I \in \mathcal{I}$. Here TC is the same set of symbols defined by (1) and are interpreted as relations over the events. Again, as for MSO over MSCs, the logic depends on a set $\mathfrak{R}$ of (binary) relation symbols, which settles the access to the partial order relation. Thus,

Definition 5.1 The set $\mathrm{TMSO}(Act, \mathfrak{R})$ of all timed monadic second-order logic formulae over Act with relational symbols from $\mathfrak{R}$, is generated inductively using the grammar:

$$\varphi ::= P_a(x) \mid x \in X \mid x = y \mid R(x, y) \mid \delta_\alpha(x) \in I \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \varphi \mid \exists X \varphi$$

where, x, y are individual variables, X is a set variable, $a \in Act$, $R \in \mathfrak{R}$, $\alpha \in \mathrm{TC}$ and $I \in \mathcal{I}$.

The *existential* fragment of $\mathrm{TMSO}(Act, \mathfrak{R})$, denoted $\mathrm{ETMSO}(Act, \mathfrak{R})$, consists of all formulas $\exists X_1 \ldots \exists X_n \varphi$ such that φ does not contain any set quantifier. Though we have defined the logic above for arbitrary sets of relational symbols $\mathfrak{R}$, we are in fact interested only in two restricted sets, namely $\mathfrak{R}_{\leq} = \{\leq\}$ and $\mathfrak{R}_{\lessdot} = \{\lessdot_{pp} \mid p \in Proc\} \cup \{\lessdot_{pq} \mid p \neq q\}$. Then a formula φ from any of these logics, i.e., $\mathrm{TMSO}(Act, \mathfrak{R}_{\leq})$, $\mathrm{TMSO}(Act, \mathfrak{R}_{\lessdot})$, $\mathrm{ETMSO}(Act, \mathfrak{R}_{\leq})$ or $\mathrm{ETMSO}(Act, \mathfrak{R}_{\lessdot})$ can be interpreted over TMSCs as well as over TCMSCs as follows. We write TMSO to denote $\mathrm{TMSO}(Act, \mathfrak{R}_{\leq} \cup \mathfrak{R}_{\lessdot})$, i.e., the union of all formulae from these logics, when there is no scope for confusion.

Now, we will give the semantics for this logic over both TMSCs and TCMSCs. Given an MSC M, let μ be an *interpretation* mapping first order variables to elements in E and second

order variables to subsets of E. Then, for $\varphi \in \text{TMSO}$ and a TMSC $T = (M, t)$, we define the satisfaction relation $T, \mu \models \varphi$, by induction on the structure of φ. For all operators, except the timing predicate, this is given as usual. For instance, the unary predicate $P_a(x)$ expresses that $\mu(x)$ is labeled with $a \in Act$, i.e., $\lambda(\mu(x)) = a$. The only novelty is the timing predicate, for which we define the satisfaction relation as follows. Intuitively, by $\delta_\alpha(x) \in I$ we mean that there is an event $e \in E$ such that $\mu(x)$ and e are related by α^M and the time difference between the events $\mu(x)$ and e is in I. Formally for each $\alpha \in \text{TC}$ we define,

$$T, \mu \models \delta_\alpha(x) \in I \quad \text{if } \exists e \in E, \text{ s.t., } \big(\mu(x), e\big) \in \alpha^M \text{ and } \big|t(e) - t\big(\mu(x)\big)\big| \in I \tag{7}$$

Then, as usual, for sentences φ (i.e., formulae without free variables) we write $T \models \varphi$ instead of $T, \mu \models \varphi$ and denote by $\mathcal{L}_{time}(\varphi)$ the set of all TMSCs T over Act such $T \models \varphi$. Turning to TCMSCs, we can give a formula $\varphi \in \text{TMSO}(Act)$ a natural semantics over $\mathfrak{M} = (M, \tau)$ exactly as done for TMSCs above. The only noteworthy difference is in the timing predicate $\delta_\alpha(x) \in I$, where for any $\alpha \in \text{TC}$, we define:

$$\mathfrak{M}, \mu \models \delta_\alpha(x) \in I \quad \text{if } \exists e \in E \text{ s.t., } \big(\mu(x), e\big) \in \alpha^M \cap \text{dom}(\tau) \text{ and } \tau\big(\mu(x), e\big) \subseteq I \tag{8}$$

The set of TCMSCs over Act that satisfy a TMSO sentence φ is denoted by $\mathcal{L}_{TC}(\varphi)$.

Example 5.1 Consider again the interaction scenario in Example 4.1, where a server is authenticating a user. Suppose we wish to specify that every Message *wrong* sent by the server is conveyed within 4 time units. This can be written as the following sentence in our logic:

$$\forall x \big(P_{q!p(wrong)}(x) \rightarrow \delta_{\text{Msg}}(x) \in [0, 4]\big) \tag{9}$$

Similarly, we may require that whenever Message *wrong* is received by the user, the time elapsed since it last sent its password is within 3 and 7 time units. This can be expressed as:

$$\forall x \big(P_{p?q(wrong)}(x) \rightarrow \delta_{\text{Prev}_{p!q(pswd)}}(x) \in [3, 7]\big) \tag{10}$$

We observe immediately that both these sentences (9) and (10) are satisfied by TCMSC $\mathfrak{M}_3$ and TMSC T_3 from Fig. 6.

The following proposition relates the expressiveness of TMSO and ETMSO under the different signatures that we have introduced above, namely $\mathfrak{R}_{\leq}$ and $\mathfrak{R}_{<}$.

Proposition 5.1

1. *For all* $\varphi \in \text{TMSO}(Act, \mathfrak{R}_{\leq})$ *there exists* $\psi \in \text{TMSO}(Act, \mathfrak{R}_{<})$ *such that* $\mathcal{L}_{time}(\varphi) = \mathcal{L}_{time}(\psi)$.
2. *Let* φ *be a* (E)TMSO$(Act, \mathfrak{R}_{<})$ *formula and* $B \in \mathbb{N}$ *such that* $\mathcal{L}_{time}(\varphi)$ *is a set of* $\exists^u$*-*B*-bounded TMSCs over Act. Then, there exists* $\psi \in$ (E)TMSO$(Act, \mathfrak{R}_{\leq})$ *such that* $\mathcal{L}_{time}(\varphi) = \mathcal{L}_{time}(\psi)$.
3. *There exists a formula* $\gamma_B \in \text{TMSO}(Act, \mathfrak{R}_{\leq})$ (*respectively, in* MSO$(Act, \mathfrak{R}_{\leq})$) *such that* $\mathcal{L}_{time}(\gamma_B)$ *is the set of all* $\exists^u$*-*B*-bounded TMSCs* (*respectively, all* $\exists$*-*B*-bounded MSCs*) *over Act.*

The first part of the proposition is a generic result which works as in the untimed case. It follows since the partial order relation can be recovered from the immediate successor and

message relations in TMSO. Note however that this is not true if we restrict to ETMSO. Indeed, even in the untimed case, it is an open question whether the transitive closure relation, i.e., $x \leq y$ can be expressed as an EMSO$(Act, \mathfrak{R}_{\lessdot})$ formula over the set of all MSCs over Act. The second part of the proposition says that the converse is true when restricted to B-bounded setting. The third part says that in both the timed and untimed settings, the set of all existentially(-untimed)-B-bounded MSCs over Act can be expressed as a formula in the (timed) MSO over $(Act, \mathfrak{R}_{\leq})$. In the untimed setting, the second part is proved in [17, Proposition 6.2] where the proof uses the (very difficult) construction of a message passing automaton accepting all ∃-B-bounded MSCs. The third part then follows as an easy consequence of this construction. However, it turns out that both these results can be proved on the same lines as [17], but without referring to the expensive automaton construction. This consequently reduces the complexity of these constructions. For this reason, and for the sake of completeness (in the timed setting) we provide proofs of both these statements below.

Proof of Proposition 5.1(2) The first two steps are exactly the same as in [17, Proposition 6.2]. First, notice that $\lessdot_{pp}$ can be easily expressed with a first-order formula over $(Act, \mathfrak{R}_{\leq})$, hence the difficulty is to express $<_{pq}$ for channels $(p, q) \in Ch$. We consider set variables $X_0, \ldots X_{B-1}$ that are not used in φ. These will represent variables that count the number of $p!q$ (resp. $q?p$) actions modulo B. This is ensured by a formula φ_0 which expresses that each X_n contains precisely the set of events e such that for some channel $(p, q) \in Ch$, either e is the n-th send from p to q modulo B, or e is the n-th receive on q from p modulo B.

Second, for each channel $(p, q) \in Ch$, we define the abbreviation $x <'_{pq} y$ by

$$\bigvee_{n<B} \begin{pmatrix} x \in X_n \wedge \lambda(x) \in p!q \\ \wedge\, y \in X_n \wedge \lambda(y) \in q?p \wedge x \leq y \\ \wedge\, \forall z (z \in X_n \wedge \lambda(z) \in q?p \wedge x \leq z) \Rightarrow y \leq z \end{pmatrix}$$

This expresses that for this channel, x is a send event and y is the least receive event above x which has the same number modulo B. Note that, for any send event e from p to q, there exists a (unique) event f such that $e <'_{pq} f$. Moreover, $f \leq g$ where g is the matching receive, i.e., is such that $e <_{pq} g$. This follows from the FIFO assumption which implies that e and g have the same number modulo B. Thus $<'_{pq}$ can be seen as a total function from the set of send events to the set of receive events. Then, we define formula φ_1 which is $\bigwedge_{(p,q)\in Ch} \forall x, x', y\ (x <'_{pq} y \wedge x' <'_{pq} y) \Rightarrow x = x'$ saying that each $<'_{pq}$ is injective function (and thus a bijection from sends to receives).

Now, denoting $\alpha = \exists X_0 \ldots X_{B-1}\ \varphi_0 \wedge \varphi_1$, we make the following crucial claims:

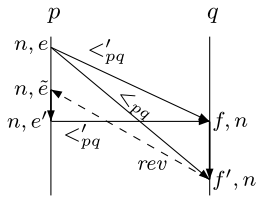
Claim *For a TMSC $T = (M, t)$ over Act with $M = (E, \leq, \lambda)$,*

(1) *if $T \models \alpha$, then the relations $<_{pq}$ and $<'_{pq}$ coincide on T, and*
(2) *if T is $\exists^{\mu}$-B-bounded, then $T \models \alpha$.*

Proof (1) Let $e \in E$. If e is not a send event from p to q then $e \not<_{pq} f$ and $e \not<'_{pq} f$ for all $f \in E$. So we assume in the following that e is a send event from p to q. Let $f, g \in E$ be the unique events with $e <'_{pq} f$ and $e <_{pq} g$. We have already seen that $f \leq g$. Assume towards a contradiction that $f < g$. By induction, we may assume that for all send event $e' < e$ from p to q, if $e' <'_{pq} f'$ and $e' <_{pq} g'$ then $f' = g'$. Event f is a receive on q from p hence there is a unique event $e' \in E$ such that $e' <_{pq} f$. Using the FIFO assumption and $f < g$ we get $e' < e$. By induction we obtain $e' <'_{pq} f$, a contradiction with φ_1.

(2) We will use an alternate characterization of $\exists^u$-bounded TMSCs obtained by lifting from the untimed setting in [22]. First, we define a relation *rev* which associates receive events to some send events. Formally, f *rev* e' if there is a channel $(p,q) \in Ch$ and an event $e \in E$ such that $e <_{pq} f$ and $\lambda(e') \in p!q$ and $|\{e'' \in E \mid \lambda(e'') \in p!q \wedge e < e'' \leq e'\}| = B$.

Now, [22] states that an MSC M is $\exists$-B-bounded iff the relation $\prec_B = < \cup rev$ is acyclic. Recalling that a TMSC is $\exists^u$-B-bounded iff its underlying MSC is $\exists$-B-bounded, we conclude that the above characterization holds for TMSCs as well.



Next, observe that in any TMSC T over Act, by counting for each channel the send (resp. receive) events modulo B, we get unique sets $X_i \subseteq E$ for $i < B$ such that $T, (X_i)_{i<B} \models \varphi_0$. Assume that $T, (X_i)_{i<B} \not\models \varphi_1$. Then, there exists a channel $(p,q) \in Ch$ and send events $e < e'$ from p to q such that $e <'_{pq} f$ and $e' <'_{pq} f$. Let $\tilde{f}$ be such that $e <_{pq} \tilde{f}$. Since $e <'_{pq} f$ we get $f \leq \tilde{f}$. By definition of $<'_{pq}$, the events e, e', f have the same number modulo B. Since $e < e'$, it follows that there exists $\tilde{e} \leq e'$ such that $\tilde{f}$ *rev* $\tilde{e}$. Therefore, we have a $\prec_B$ cycle $\tilde{e} \leq e' <'_{pq} f \leq \tilde{f}$ *rev* $\tilde{e}$ (observing that $e <'_{pq} f$ implies $e < f$), as depicted in the adjoining figure. □

Now, we are in a position to complete the proof of Part (2) of this proposition. Starting from a formula $\varphi \in$ (E)TMSO$(Act, \mathfrak{R}_<)$, let φ' be the formula obtained by replacing every occurrence of $<_{pq}$ by $x <'_{pq} y$. Now, consider the formula $\psi = \exists (X_i)_{i<B}\ \varphi_0 \wedge \varphi_1 \wedge \varphi'$. Clearly it is a (E)TMSO$(Act, \mathfrak{R}_\leq)$ formula. We claim that $\mathcal{L}_{time}(\varphi) = \mathcal{L}_{time}(\psi)$. If $T \models \varphi$, then T is $\exists^u$-B-bounded over Act therefore $T \models \alpha$ by Claim (2) above. But since $T \models \alpha$, by Claim (1), $<'_{pq}$ and $<_{pq}$ coincide on T, which implies that $T, (X_i)_{i<B} \models \varphi'$. Thus combining the two, we obtain $T \models \psi$. Conversely, assume that $T \models \psi$. Since $T \models \alpha$, we know that $<'_{pq}$ and $<_{pq}$ coincide on M. Hence $T, (X_i)_{i<B} \models \varphi'$ implies $T \models \varphi$. □

Proof of Proposition 5.1(3) Now, we prove the final part of the proposition in the timed case (the untimed setting follows in exactly the same way). We use freely the formulas from the proof above. First, we let y *rev*′ z stand for

$$\bigvee_{(p,q)\in Ch} \exists x\ x < z \wedge x <'_{pq} y \wedge \lambda(z) \in p!q \wedge \bigvee_{n<B} x, z \in X_n$$

If $T \models \alpha$ then Claim (1) implies that $<'_{pq}$ and $<_{pq}$ coincide and it follows easily that y *rev* z implies y *rev*′ z and that y *rev*′ z' implies y *rev* z for some $z \leq z'$. Therefore, there is a $\prec_B$ cycle iff there is a $\prec'_B$ cycle where $\prec'_B = < \cup rev'$. We define below a new formula φ_2 to check, in the context of α, the existence of a $\prec'_B$ cycle.

In fact, it suffices to test for short cycles. Indeed, let $x_0 \prec'_B x_1 \prec'_B \cdots \prec'_B x_k \prec'_B x_0$ be a cycle with $k > 0$ and $x_i \neq x_j$ for $0 \leq i < j \leq k$. If two events x_i and x_j are on the same process for some $0 < i + 1 < j < k$ then either $x_i < x_j$ and we have a shorter cycle by

removing $x_{i+1}, \ldots, x_{j-1}$; or $x_j < x_i$ and $x_i, \ldots, x_j$ is a shorter cycle. Hence, it suffices to test for the existence of a $\prec'_B$ cycle of length bounded by $2|Proc| + 1$, which can easily be expressed with a first-order formula φ_2 over $(Act, \mathfrak{R}_{\leq})$.

Finally, let $\gamma_B = \exists X_0 \ldots X_{B-1}\, \varphi_0 \wedge \varphi_1 \wedge \neg\varphi_2$ which is in TMSO$(Act, \mathfrak{R}_{\leq})$. We can easily check that γ_B defines the set of all $\exists^u$-B-bounded TMSCs. □

In the above sections, we have introduced our models to describe as well as implement timed and distributed scenarios. In the next sections, we state and prove our main results.

6 Equivalence between ECMPA and TMSO logic over TMSCs

We now state our main results showing an effective equivalence between ECMPAs and TMSOs over TMSCs, which we will prove in this section.

Theorem 6.1 *Let $\mathcal{L}$ be a set of TMSCs over Act. Then, the following are equivalent*:

1. *There is an ECMPA $\mathcal{A}$ such that $\mathcal{L}_{time}(\mathcal{A}) = \mathcal{L}$.*
2. *There is $\varphi \in$ ETMSO$(Act, \mathfrak{R}_{<})$ such that $\mathcal{L}_{time}(\varphi) = \mathcal{L}$.*

Theorem 6.2 *Let $B \in \mathbb{N}_{>0}$ and let $\mathcal{L}$ be a set of $\exists^u$-B-bounded TMSCs over Act. Then, the following are equivalent*:

1. *There is an ECMPA $\mathcal{A}$ such that $\mathcal{L}_{time}(\mathcal{A}) = \mathcal{L}$.*
2. *There is $\varphi \in$ TMSO$(Act, \mathfrak{R}_{\leq})$ such that $\mathcal{L}_{time}(\varphi) = \mathcal{L}$.*

This equivalences are effective in the sense that we can explicitly construct the (E)TMSO formula from the ECMPA and vice versa.

The construction of an (E)TMSO formula from an ECMPA follows the similar constructions applied, for example, to finite and asynchronous automata. In addition, we have to cope with the timing predicate. Assume that $g : \mathrm{TC} \dashrightarrow \mathcal{I}$ is such a guard occurring on a local transition of the given ECMPA. To ensure that the timing constraints that come along with g are satisfied we use the formula $\bigwedge_{\alpha \in \mathrm{dom}(g)} \delta_\alpha(x) \in g(\alpha)$.

The difficult part is the construction of an ECMPA from an (E)TMSO formula. The basic idea is to reduce this to an analogous untimed case, which has also been applied in the settings of words and traces [14, 15]. Usually, the untimed formalisms need to be parameterized by a finite alphabet, so that we can speak of structures whose labelings are extended with this alphabet. Hence, in our framework, we need to find a finite abstraction of the infinite set of possible time stamps. For this, we move from TMSCs to TCMSCs over a finite alphabet, using Lemma 6.2 which strengthens Lemma 4.1 and Lemma 6.3 which is the corresponding result for TMSO. This allow us to establish a translation of (E)TMSO formulas into ECMPAs.

6.1 From TMSCs to TCMSCs

TCMSCs are abstractions of TMSCs. Thus, if a TCMSC exhibits a property, the corresponding TMSC should also do so. This is illustrated in Lemma 4.1 where the *property* is having a run on an automaton. In this section, we are interested in the converse question. In other words, if a TMSC exhibits a property, when can we say that a TCMSC that it realizes also exhibits the same property? For this, given a TMSC, we derive a canonical representative

TCMSC using intervals from a specific set which depends on the property. Then it turns out that, this representative exhibits the property iff a TMSC realizing it exhibits the property.

We formalize these ideas now. We begin by introducing the notion of a *proper interval set* from [14], which will play an important role in what follows.

Definition 6.1 A set of intervals $\mathcal{S} \subseteq \mathcal{I}$ is said to be *proper* if it forms a finite partition of $\mathbb{R}_{\geq 0}$. An interval set $\mathcal{S}$ is said to *refine* another interval set $\mathcal{S}'$ if every interval $I' \in \mathcal{S}'$ is the union of some collection of intervals of $\mathcal{S}$.

Example 6.1 Consider the set of intervals $\mathcal{S}_1 = \{[0, 4], [3, 7]\}$. Then, we may observe that the interval set $\mathcal{S}_2 = \{[0, 3), [3, 4], (4, 7]\}$ refines $\mathcal{S}_1$ and if we add the interval $(7, \infty)$ to $\mathcal{S}_2$ we obtain a proper interval set $\mathcal{S}_3$ that refines $\mathcal{S}_1$.

Now, observe that if $\mathcal{S}$ is a proper interval set which refines another interval set $\mathcal{S}'$, then for all $I \in \mathcal{S}$ and $I' \in \mathcal{S}'$, we have, either $I \subseteq I'$ or $I \cap I' = \emptyset$. Also, we have,

Proposition 6.1 *For any finite interval set, there exists a proper interval set that refines it.*

Proof Let $\mathcal{S} \subseteq \mathcal{I}$ be a finite interval set. Then, we define a canonical proper interval set of $\mathcal{S}$ denoted $\text{prop}(\mathcal{S})$ as follows. If R is empty, we define $\text{prop}(\mathcal{S}) = \{[0, \infty)\}$. Otherwise, we let $R = (t_1, \ldots, t_n)$ be the sequence of bounds that appear in $\mathcal{S}$, arranged in increasing order $t_1 < \cdots < t_n$ and which are different from $0, \infty$. Then, we define $\text{prop}(\mathcal{S}) = \{[0, 0], (0, t_1), [t_1, t_1], (t_1, t_2), \ldots, [t_n, t_n], (t_n, \infty)\}$. With this definition it follows that $\text{prop}(\mathcal{S})$ is a proper interval set and that $\text{prop}(\mathcal{S})$ refines $\mathcal{S}$. □

Now we can show that for a proper interval set $\mathcal{S}$ and a TMSC T, there is a unique maximally defined TCMSC using intervals only from $\mathcal{S}$ such that T realizes it. Formally,

Lemma 6.1 *Let $\mathcal{S}$ be a proper interval set and $T = (M, t)$ be a TMSC over Act. Then, there exists a unique TCMSC $\mathfrak{M} = (M, \tau)$ over $(Act, \mathcal{S})$ such that $\tau : \text{TC}^M \to \mathcal{S}$, T realizes $\mathfrak{M}$ and $\mathfrak{M}$ is maximally defined. This unique TCMSC is denoted $\mathfrak{M}_T^{\mathcal{S}}$.*

Proof We first observe that, for each $(e, e') \in \text{TC}^M$, the real number $|t(e') - t(e)|$ is in a unique interval of $\mathcal{S}$. Thus, consider the maximally defined TCMSC defined as: $\mathfrak{M}_T^{\mathcal{S}} = (M, \tau)$ where, for any $(e, e') \in \text{TC}^M$, $\tau(e, e')$ is defined to be the unique interval of $\mathcal{S}$ containing $|t(e') - t(e)|$. Then, T realizes $\mathfrak{M}_T^{\mathcal{S}}$ by definition and the uniqueness follows since $\mathcal{S}$ is a proper interval set. □

It turns out that this unique TCMSC is the "canonical representative" for a TMSC that we were searching for. As an example, consider the TMSC T_4 from Fig. 7, which represents a part of the scenario of TMSC T_3 from Fig. 6 (and abstracting away the message contents). Now, let $\mathcal{S}_3$ be the proper interval set defined in Example 6.1. Then, the unique maximally defined TCMSC over $(Act, \mathcal{S}_3)$ which is realized by T_4 is the TCMSC $\mathfrak{M}_{T_4}^{\mathcal{S}_3}$ shown in Fig. 7.

Now, given an ECMPA $\mathcal{A}$, let $\text{Int}(\mathcal{A})$ denote the finite set of intervals that occur in $\mathcal{A}$ as guards. Now look at any proper interval set $\mathcal{S}$ that refines $\text{Int}(\mathcal{A})$. By Proposition 6.1, there exists at least one, namely $\text{prop}(\text{Int}(\mathcal{A}))$. Then,

Lemma 6.2 *Given a TMSC T, an ECMPA $\mathcal{A}$ and a proper interval set $\mathcal{S}$ that refines* $\text{Int}(\mathcal{A})$, *we have $T \in \mathcal{L}_{time}(\mathcal{A})$ iff $\mathfrak{M}_T^{\mathcal{S}} \in \mathcal{L}_{TC}(\mathcal{A})$.*

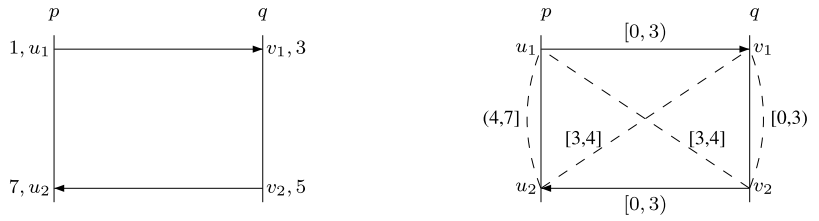


Fig. 7 TMSC T_4 and its representative TCMSC $\mathfrak{M}_{T_4}^{\mathcal{S}_3}$

Proof Let $T = (M, t)$ be the TMSC over Act with $M = (E, \leq, \lambda)$ and let the ECMPA be $\mathcal{A} = (\{\mathcal{A}_p\}_{p \in Proc}, Act, \Delta, F)$ with $\mathcal{A}_p = (S_p, \iota_p, \rightarrow_p)$. Now, since T realizes $\mathfrak{M}_T^{\mathcal{S}}$, by Lemma 4.1, we obtain one direction of the result, $\mathfrak{M}_T^{\mathcal{S}} \in \mathcal{L}_{TC}(\mathcal{A})$ implies $T \in \mathcal{L}_{time}(\mathcal{A})$.

For the reverse direction, assume $T \in \mathcal{L}_{time}(\mathcal{A})$. Then by definition there is a successful run r of $\mathcal{A}$ on T. Since r is a run, for all $e, e' \in E$ s.t., $e <_{pq} e'$ for $(p, q) \in Ch$, we find $g, g' \in [TC \dashrightarrow \mathcal{I}]$ and $d \in \Delta$ such that conditions (2), (5) and (6) hold. We show then that r is also a run of $\mathcal{A}$ on $\mathfrak{M}_T^{\mathcal{S}}$, for which it is enough to show that conditions (3) and (4) hold. We start from (5), which says that for each $\alpha \in \mathrm{dom}(g)$, there is $\widetilde{e} \in E$ such that $(e, \widetilde{e}) \in \alpha^M$. Since $\mathfrak{M}_T^{\mathcal{S}}$ is maximally defined, $\tau(e, \widetilde{e})$ exists and as T realizes $\mathfrak{M}_T^{\mathcal{S}}$ we obtain $|t(\widetilde{e}) - t(e)| \in \tau(e, \widetilde{e})$. But again from (5), $|t(\widetilde{e}) - t(e)| \in g(\alpha)$. Thus, we find $\tau(e, \widetilde{e}) \cap g(\alpha) \neq \emptyset$. Now, $\tau(e, \widetilde{e}) \in \mathcal{S}$, $g(\alpha) \in \mathrm{Int}(\mathcal{A})$ and we know that $\mathcal{S}$ is a proper interval set that refines $\mathrm{Int}(\mathcal{A})$, which implies $\tau(e, \widetilde{e}) \subseteq g(\alpha)$ concluding that (3) holds. Similarly (4) can be shown starting from (6). Thus any run of $\mathcal{A}$ on T is also a run on $\mathfrak{M}_T^{\mathcal{S}}$. Since acceptance criterion for a run is the same, $T \in \mathcal{L}_{time}(\mathcal{A})$ implies $\mathfrak{M}_T^{\mathcal{S}} \in \mathcal{L}_{TC}(\mathcal{A})$. □

We can do the same for TMSO as well. That is, given a TMSO formula φ, we let $\mathrm{Int}(\varphi)$ denote the finite set of intervals I for which φ has a sub-formula of the form $\delta_\alpha(x) \in I$. Again we can consider a proper interval set $\mathcal{S}$ which refines $\mathrm{Int}(\varphi)$.

Lemma 6.3 *Given a TMSC T, a TMSO formula φ, and a proper interval set $\mathcal{S}$ that refines* $\mathrm{Int}(\varphi)$, *we have $T \models \varphi$ iff $\mathfrak{M}_T^{\mathcal{S}} \models \varphi$.*

Proof Let $T = (M, t)$ be a TMSC with $M = (E, \leq, \lambda)$. Then, by Lemma 6.1, we have the TCMSC $\mathfrak{M}_T^{\mathcal{S}} = (M, \tau)$ such that, for all $\alpha \in \mathrm{TC}$, and all $(e, e') \in \alpha^M$, $|t(e') - t(e)| \in \tau(\alpha)$.

We prove the lemma by structural induction on φ. Let μ be any interpretation. The only interesting case is the timing predicate. The others are routine deductions. We have,

$$
\begin{aligned}
& T, \mu \models \delta_\alpha(x) \in I \\
& \quad \Longleftrightarrow \quad \exists e \in E, \big(\mu(x), e\big) \in \alpha^M \wedge \big|t(e) - t\big(\mu(x)\big)\big| \in I \\
& \quad \Longleftrightarrow \quad \exists e \in E, \big(\mu(x), e\big) \in \alpha^M \cap \mathrm{dom}(\tau) \wedge \big|t(e) - t\big(\mu(x)\big)\big| \in I \cap \tau\big(\mu(x), e\big) \\
& \quad \Longleftrightarrow \quad \exists e \in E, \big(\mu(x), e\big) \in \alpha^M \cap \mathrm{dom}(\tau) \wedge \tau\big(\mu(x), e\big) \subseteq I \\
& \quad \Longleftrightarrow \quad \mathfrak{M}_T^{\mathcal{S}}, \mu \models \delta_\alpha(x) \in I
\end{aligned}
$$

using successively the definition of the semantics (7) for TMSCs, $\mathfrak{M}_T^{\mathcal{S}}$ is maximally defined and T realizes $\mathfrak{M}_T^{\mathcal{S}}$, $\mathcal{S}$ is proper and refines $\mathrm{Int}(\varphi)$, and the semantics (8) for TCMSCs. □

6.2 Extending the alphabet

In this subsection, we provide the final pieces of our jigsaw. In particular, we fix a finite set Π and lift MSCs over Act to MSCs over $\Gamma = Act \times \Pi$. A *Π-extended MSC* over Act or an *MSC over Γ* is a finite Γ-labeled poset $M = (E, \leq, \lambda)$ such that conditions 1–4 of Definition 2.1 are satisfied, ignoring the extra labeling. Note that the definition of boundedness can be immediately adapted to this setting.

We lift the MPA and MSO definitions to include the additional alphabet. Since such a lift is purely syntactical, we preserve the validity of the equivalence theorems 6.3 and 6.4 in these settings. We define an MPA $\mathcal{A} = (\{\mathcal{A}_p\}_{p \in Proc}, \Delta, F)$ over Γ as in Definition 4.1 with the only change being the transition relation $\rightarrow_p \subseteq (S_p \times Act_p \times \Pi \times \Delta \times S_p)$ of component $\mathcal{A}_p$. Runs of $\mathcal{A}$ over an MSC over Γ is defined as in Sect. 4, ignoring guards g, g' and conditions 7 and 8. $\mathcal{L}_{MSC}(\mathcal{A})$ denotes the set of all MSCs over Γ that are accepted by $\mathcal{A}$.

Similarly, dropping the timing predicate, we define the logics $\mathrm{MSO}(Act \times \Pi, \mathfrak{R}_{\leq})$ and $\mathrm{MSO}(Act \times \Pi, \mathfrak{R}_{<})$ over MSCs over Γ as in Sect. 5. For a sentence φ, $\mathcal{L}_{MSC}(\varphi)$ denotes the set of MSCs over Γ that satisfy it. With the above definitions, it is easy to see that we can lift the results from [7, 17]. We restate the relevant theorems in our terminology.

Theorem 6.3 (From [7]) *Let $\mathcal{L}$ be a language of MSCs over Γ. Then $\mathcal{L} = \mathcal{L}_{MSC}(\varphi)$ for some $\varphi \in \mathrm{EMSO}(\Gamma, \mathfrak{R}_{<})$ iff $\mathcal{L} = \mathcal{L}_{MSC}(\mathcal{A})$ for some MPA $\mathcal{A}$ over Γ.*

Theorem 6.4 (From [17]) *Let $B \in \mathbb{N}_{>0}$. Let $\mathcal{L}$ be a language of $\exists$-B-bounded MSCs over Γ. Then, $\mathcal{L} = \mathcal{L}_{MSC}(\varphi)$ for some sentence $\varphi \in \mathrm{MSO}(\Gamma, \mathfrak{R}_{\leq})$ iff $\mathcal{L} = \mathcal{L}_{MSC}(\mathcal{A})$ for some MPA $\mathcal{A}$ over Γ.*

TCMSCs as extended MSCs Let $\mathcal{S} \subseteq \mathcal{I}$ be a finite set of intervals. Then, we can consider the alphabet $\Gamma = Act \times [\mathrm{TC} \dashrightarrow \mathcal{S}]$ where $[\mathrm{TC} \dashrightarrow \mathcal{S}]$ is a finite set of partial maps from TC to $\mathcal{S}$. Recall that TC is the finite set of symbols defined in Sect. 4. A TCMSC over $(Act, \mathcal{S})$ can be directly seen as an MSC over Γ. We make this precise by defining an *untiming* function $\mathcal{U}$ which maps TCMSCs over $(Act, \mathcal{S})$ to MSCs over Γ.

Let $\mathfrak{M} = (M, \tau)$ be any TCMSC over $(Act, \mathcal{S})$, with $M = (E, \leq, \lambda)$, $\tau : \mathrm{TC}^M \dashrightarrow \mathcal{S}$. Then $\mathcal{U}(\mathfrak{M}) = (E, \leq, \lambda')$ is an MSC over Γ with the same set of events E and partial order $\leq$. Also, $\lambda' : E \rightarrow \Gamma$ is defined for all $e \in E$, by $\lambda'(e) = (\lambda(e), g_e)$ where $g_e : \mathrm{TC} \dashrightarrow \mathcal{S}$ is such that, for all $\alpha \in \mathrm{TC}$, $g_e(\alpha) = \tau(e, e')$ if there exists $e' \in E$, s.t., $(e, e') \in \alpha^M \cap \mathrm{dom}(\tau)$ (and $g_e(\alpha)$ is undefined otherwise). Recall that given $e \in E$, $\alpha \in \mathrm{TC}$, there exists at most one event $e' \in E$ s.t., $(e, e') \in \alpha^M$.

Lemma 6.4 *Let $\mathcal{S}$ be a finite set of intervals and $\Gamma = Act \times [\mathrm{TC} \dashrightarrow \mathcal{S}]$. We can build an MPA $\mathcal{B}$ over Γ such that for all TCMSC $\mathfrak{M}$ over $(Act, \mathcal{S})$ we have $\mathcal{U}(\mathfrak{M}) \in \mathcal{L}_{MSC}(\mathcal{B})$ iff $\mathfrak{M}$ is maximally defined.*

Proof (sketch) First, we check that for each event labeled (b, g) we have $\mathrm{Msg} \in \mathrm{dom}(g)$ iff b is a send action. Then, we have to check that $\mathrm{Prev}_a \in \mathrm{dom}(g)$ iff some action a occurs in the *past* of the current event. The set of actions occurring in the past can be computed (deterministically) by an MPA using the set Δ of auxiliary messages. Finally, we have to check that $\mathrm{Next}_a \in \mathrm{dom}(g)$ iff some action a occurs in the *future* of the current event. The set of actions occurring in the future can be guessed non-deterministically by an MPA. Such guesses can be checked using the set Δ of auxiliary messages and the set *Fin* of global accepting states. □

TMSO as MSO over the extended alphabet We will now see that each TMSO formula can be rewritten as an MSO formula over the extended alphabet of guards Γ introduced above. Let $\mathcal{S}$ be a finite set of intervals and $\Gamma = Act \times [\text{TC} \dashrightarrow \mathcal{S}]$. Then from a TMSO formula φ, we obtain $\varphi^{\mathcal{S}} \in \text{MSO}(\Gamma)$ by replacing each sub-formula of the form $P_a(x)$ by the formula $\bigvee_{\{(b,g)\in\Gamma \mid b=a\}} P_{(b,g)}(x)$ and each sub-formula of the form $\delta_\alpha(x) \in I$ (i.e., timing predicate) by the formula $\bigvee_{\{(b,g)\in\Gamma \mid g(\alpha)\subseteq I\}} P_{(b,g)}(x)$. This translation preserves the existential fragment, for instance, if $\varphi \in \text{ETMSO}(Act, \mathfrak{R}_{\leq})$, then $\varphi^{\mathcal{S}} \in \text{EMSO}(\Gamma, \mathfrak{R}_{\leq})$. As before, we can easily obtain the semantics of a formula from this logic in terms of MSCs over Γ. Then, we can relate the TCMSC-language of a TMSO formula and the extended MSC language of its MSO translation as follows:

Lemma 6.5 *Let φ be a TMSO sentence and $\mathcal{S}$ be a finite set of intervals. Then for a TCMSC $\mathfrak{M}$ over $(Act, \mathcal{S})$, $\mathfrak{M} \models \varphi$ if and only if $\mathcal{U}(\mathfrak{M}) \models \varphi^{\mathcal{S}}$.*

Proof Let $\mathfrak{M} = (M, \tau)$ be a TCMSC over $(Act, \mathcal{S})$ with $M = (E, \leq, \lambda)$ and let $\mathcal{U}(\mathfrak{M}) = (E, \leq, \lambda')$ as defined above. We show the lemma for any interpretation μ by induction on structure of φ. As before, the only interesting cases are atomic and timing predicates. The others are routine deductions.

– Suppose φ is of the form $P_a(x)$ for some $a \in Act$. Then, we obtain easily,

$$\mathfrak{M}, \mu \models P_a(x) \quad \text{iff} \quad \mathcal{U}(\mathfrak{M}), \mu \models \bigvee_{\{(b,g)\in\Gamma \mid b=a\}} P_{(b,g)}(x)$$

– Suppose φ be of the form $\delta_\alpha(x) \in I$ for some $\alpha \in \text{TC}$, $I \in \mathcal{I}$. Then, we show that,

$$\mathfrak{M}, \mu \models \delta_\alpha(x) \in I \quad \text{iff} \quad \mathcal{U}(\mathfrak{M}), \mu \models \bigvee_{\{(b,g)\in\Gamma \mid g(\alpha)\subseteq I\}} P_{(b,g)}(x)$$

($\Rightarrow$) Assume $\mathfrak{M}, \mu \models \delta_\alpha(x) \in I$ where $\mu(x) = e \in E$. By definition, this means that there exists $e' \in E$ such that $(e, e') \in \alpha^M$ and $\tau(e, e') \subseteq I$. By the definition of $\mathcal{U}(\mathfrak{M})$, we then have $\lambda'(e) = (\lambda(e), g_e)$ with $g_e : \text{TC} \dashrightarrow \mathcal{S}$ such that $g_e(\alpha) = \tau(e, e') \subseteq I$. This implies that $\mathcal{U}(\mathfrak{M}), \mu \models P_{(\lambda(e),g_e)}(x)$ with $g_e(\alpha) \subseteq I$. Hence we have, $\mathcal{U}(\mathfrak{M}), \mu \models \bigvee_{\{(b,g)\in\Gamma \mid g(\alpha)\subseteq I\}} P_{(b,g)}(x)$.
($\Leftarrow$) Conversely assume $\mathcal{U}(\mathfrak{M}), \mu \models P_{(b,g)}(x)$ for some $(b, g) \in \Gamma$ such that $g(\alpha) \subseteq I$ and let $\mu(x) = e \in E$. Then by definition $\lambda'(e) = (b, g)$, $\lambda(e) = b$ and $g : \text{TC} \dashrightarrow \mathcal{S}$ with $g(\alpha) \subseteq I$. Since $g(\alpha)$ is defined, there exists $e' \in E$ s.t., $(e, e') \in \text{dom}(\tau) \cap \alpha^M$ and $g(\alpha) = \tau(e, e') \subseteq I$. Thus, we conclude that $\mathfrak{M}, \mu \models \delta_\alpha(x) \in I$. □

ECMPA as MPA over the extended alphabet Let us fix a finite set of intervals $\mathcal{S}$ and an alphabet of guards $\Gamma = Act \times [\text{TC} \dashrightarrow \mathcal{S}]$. Then, observe that any MPA $\mathcal{A}$ over Γ is itself an ECMPA which uses intervals from $\mathcal{S}$ as guards, i.e., $\text{Int}(\mathcal{A}) \subseteq \mathcal{S}$.

Lemma 6.6 *Let $\mathcal{S}$ be a finite set of intervals and $\mathcal{A}$ be an MPA over $\Gamma = Act \times [\text{TC} \dashrightarrow \mathcal{S}]$. Then for any TCMSC $\mathfrak{M}$ over $(Act, \mathcal{S})$, $\mathcal{U}(\mathfrak{M}) \in \mathcal{L}_{MSC}(\mathcal{A})$ implies $\mathfrak{M} \in \mathcal{L}_{TC}(\mathcal{A})$.*

Proof Let $\mathcal{A}$ be the MPA over Γ and let $\mathfrak{M} = (M, \tau)$ over $(Act, \mathcal{S})$ with $M = (E, \leq, \lambda)$. Let $\mathcal{U}(\mathfrak{M}) = (E, \leq, \lambda') \in \mathcal{L}_{MSC}(\mathcal{A})$, i.e., there exists a run r of the MPA $\mathcal{A}$ on the MSC $\mathcal{U}(\mathfrak{M})$

which is accepting. Now $r : E \to \bigcup_{p \in Proc} S_p$ satisfies for all $e, e' \in E$ such that $e <_{pq} e'$ with $(p, q) \in Ch$, there exists $d \in \Delta$ such that,

$$\big(r^-(e), \lambda'(e), d, r(e)\big) \in \to_p \quad \text{and} \quad \big(r^-\big(e'\big), \lambda'\big(e'\big), d, r\big(e'\big)\big) \in \to_q$$

We claim that r is a run of ECMPA $\mathcal{A}$ on the TCMSC $\mathfrak{M}$. Since $\lambda'(e) = (\lambda(e), g_e)$ and $\lambda'(e') = (\lambda(e'), g_{e'})$, condition (2) follows at once with $g = g_e$ and $g' = g_{e'}$. Again, by definition of λ', $g_e \in [\mathrm{TC} \dashrightarrow \mathcal{S}]$ has the property that for all $\alpha \in \mathrm{dom}(g_e)$, there exists $\widetilde{e} \in E$ such that $(e, \widetilde{e}) \in \alpha^M \cap \mathrm{dom}(\tau)$ and $g_e(\alpha) = \tau(e, \widetilde{e})$. Thus condition (3) is satisfied. Similarly, $g_{e'}$ satisfies condition (4). Thus, r is a run of $\mathcal{A}$ on $\mathfrak{M}$. It is accepting since the acceptance condition is the same and depends on reaching the final state. Thus we conclude that $\mathfrak{M} \in \mathcal{L}_{TC}(\mathcal{A})$. □

Lemma 6.7 *Let $\mathcal{S}$ be a proper finite set of intervals and $\mathcal{A}$ be an MPA over $\Gamma = Act \times [\mathrm{TC} \dashrightarrow \mathcal{S}]$. Then, for any TMSC $T \in \mathcal{L}_{time}(\mathcal{A})$, there exists a TCMSC $\mathfrak{M}$ over $(Act, \mathcal{S})$ such that $\mathcal{U}(\mathfrak{M}) \in \mathcal{L}_{MSC}(\mathcal{A})$ and T realizes $\mathfrak{M}$.*

Proof We begin with an accepting run r of $\mathcal{A}$ on TMSC $T = (M, t)$. Thus, for all $e, e' \in E$ with $e <_{pq} e'$ for some $(p, q) \in Ch$, there are $g_e, g_{e'} \in [\mathrm{TC} \dashrightarrow \mathcal{I}]$ and $d \in \Delta$ such that conditions (2), (5), (6) hold. But since $\mathrm{Int}(\mathcal{A}) \subseteq \mathcal{S}$, we obtain $g_e, g_{e'} \in [\mathrm{TC} \dashrightarrow \mathcal{S}]$.

We recall that by condition (5), for all $\alpha \in \mathrm{dom}(g_e)$ there exists $\tilde{e} \in E$ such that $(e, \tilde{e}) \in \alpha^M$ and $|t(e) - t(\tilde{e})| \in g_e(\alpha)$. Similarly from condition (6), for all $\alpha \in \mathrm{dom}(g_{e'})$ there exists $\tilde{e}' \in E$ such that $(e', \tilde{e}') \in \alpha^M$ and $|t(e') - t(\tilde{e}')| \in g_{e'}(\alpha)$. Using these partial maps, we define another partial function $\tau \in [\mathrm{TC}^M \dashrightarrow \mathcal{S}]$ as

$$\tau(e, \tilde{e}) = \begin{cases} g_e(\alpha) & \text{if } \exists\ \alpha \in \mathrm{dom}(g_e) \text{ s.t. } (e, \tilde{e}) \in \alpha^M \\ \text{undefined} & \text{otherwise} \end{cases}$$

Observe that τ is well-defined. If we find $\alpha, \alpha' \in \mathrm{dom}(g_e)$ such that $(e, \tilde{e}) \in \sigma^M \cap \alpha'^M$ then $|t(e) - t(\tilde{e})| \in g_e(\alpha') \cap g_e(\alpha)$ by (5), (6). Now $\mathcal{S}$ is a proper interval set implies that $g_e(\alpha) = g_e(\alpha')$. Now using the above map we define a TCMSC over $(Act, \mathcal{S})$ as $\mathfrak{M} = (M, \tau)$. T realizes $\mathfrak{M}$ by definition, since for all $(e, \tilde{e}) \in \mathrm{dom}(\tau)$, $|t(e) - t(\tilde{e})| \in g_e(\alpha) = \tau(e, \tilde{e})$ for some $\alpha \in \mathrm{dom}(g_e)$. We are done if we show that $\mathcal{U}(\mathfrak{M}) \in \mathcal{L}_{MSC}(\mathcal{A})$. But this follows since r is itself a run of $\mathcal{A}$ over $\mathcal{U}(\mathfrak{M})$. It is enough to observe that $\lambda'(e) = (\lambda(e), g_e)$ where g_e is the partial map given above from the run on the TMSC. Then, for all $e, e' \in E$ such that $e <_{pq} e'$ for some $(p, q) \in Ch$, there exists $d \in \Delta$ such that $(r^-(e), \lambda'(e), d, r(e)) \in \to_p$, $(r^-(e'), \lambda'(e'), d, r(e')) \in \to_q$. Thus r is an accepting run of $\mathcal{A}$ on $\mathcal{U}(\mathfrak{M})$. □

6.3 Proof of Theorems 6.1 and 6.2

Proof of Theorem 6.1 (1 ⇒ 2) Given an ECMPA $\mathcal{A}$, we construct an ETMSO formula $\varphi \in \mathrm{ETMSO}(Act, \mathfrak{R}_{\lessdot})$ such that $\mathcal{L}_{time}(\mathcal{A}) = \mathcal{L}_{time}(\varphi)$. This direction of the proof is standard and does not need the lemmas we proved in the two previous subsections.

Let $\mathcal{A} = (\{\mathcal{A}_p\}_{p \in Proc}, Act, \Delta, F)$ be the ECMPA at hand with $\mathcal{A}_p = (S_p, \iota_p, \to_p)$. For any local state $s \in S = \bigcup_{p \in Proc} S_p$, we introduce a second order variable X_s. The formula we are targeting guesses a run of $\mathcal{A}$ in terms of an assignment of events to the variables $(X_s)_{s \in S}$. Accordingly, $(X_s)_{s \in S}$ needs to be a partition of all the events of an MSC. This can easily be done by a first order formula $Partition((X_s)_{s \in S})$ with free variables $(X_s)_{s \in S}$. We now define

some further macros. For a synchronization message $d \in \Delta$, we define $Trans_d(x, (X_s)_{s\in S})$ by

$$\bigvee_{\substack{p\in Proc\\(s,g,a,d,s')\in\rightarrow_p}} \left(P_a(x) \wedge x \in X_{s'} \wedge \left[\exists y(y \lessdot_{pp} x \wedge y \in X_s)\right] \wedge \bigwedge_{\alpha\in\mathrm{dom}(g)} \delta_\alpha(x) \in g(\alpha) \right)$$

$$\vee \bigvee_{\substack{p\in Proc\\(\iota_p,g,a,d,s')\in\rightarrow_p}} \left(P_a(x) \wedge x \in X_{s'} \wedge \left[\neg\exists y(y \lessdot_{pp} x)\right] \wedge \bigwedge_{\alpha\in\mathrm{dom}(g)} \delta_\alpha(x) \in g(\alpha) \right)$$

This formula describes that, under the assignment $(X_s)_{s\in S}$, the execution of x actually corresponds to a local transition of $\mathcal{A}$ that communicates $d \in \Delta$. Moreover, for a global state $\overline{s} = (s_p)_{p\in Proc} \in \prod_{s\in Proc} S_p$, we define $Final_{\overline{s}}((X_s)_{s\in S})$ by

$$\bigvee_{\substack{Proc'\subseteq Proc\\ \forall p\in Proc': s_p=\iota_p}} \left(\bigwedge_{p\in Proc\setminus Proc'} \exists x \left(\max_p(x) \wedge x \in X_{s_p}\right) \wedge \bigwedge_{\substack{p\in Proc'\\ a\in Act_p}} \neg\exists x\, P_a(x) \right)$$

Hereby, given a process $p \in Proc$, $\max_p(x) = \bigvee_{a\in Act_p} P_a(x) \wedge (\neg\exists y\ (x \lessdot_{pp} y))$. Moreover, observe that $Proc'$ comprises those processes that are assumed not to move. Hence, $Final_{\overline{s}}((X_s)_{s\in S})$ formulates that the run described by $(X_s)_{s\in S}$ ends up in global state $\overline{s}$.

We are now prepared to give the formula φ with $\mathcal{L}_{time}(\varphi) = \mathcal{L}_{time}(\mathcal{A})$. Namely,

$$\varphi = \exists (X_s)_{s\in S}\ Partition\big((X_s)_{s\in S}\big) \wedge \bigvee_{\overline{s}\in F} Final_{\overline{s}}\big((X_s)_{s\in S}\big)$$

$$\wedge \forall x \forall y \bigwedge_{(p,q)\in Ch} \left(x <_{pq} y \rightarrow \bigvee_{d\in\Delta} \big(Trans_d\big(x, (X_s)_{s\in S}\big) \wedge Trans_d\big(y, (X_s)_{s\in S}\big)\big) \right)$$

This concludes the proof in one direction.

($2 \Rightarrow 1$) For the other direction we will require all the machinery that we set up in the two previous subsections.

Let φ be the given ETMSO$(Act, \mathfrak{R}_{\lessdot})$ formula and let $\mathcal{S}$ be a proper interval set which refines Int(φ). Fix an alphabet $\Gamma = Act \times [\mathrm{TC} \dashrightarrow \mathcal{S}]$. Thus, we have $\varphi^{\mathcal{S}} \in \mathrm{EMSO}(\Gamma, \mathfrak{R}_{\lessdot})$ using the translation from Sect. 6.2. Then, by Theorem 6.3 we obtain an MPA $\mathcal{A}$ over Γ such that $\mathcal{L}_{MSC}(\mathcal{A}) = \mathcal{L}_{MSC}(\varphi^{\mathcal{S}})$. Let $\mathcal{B}$ be the MPA over Γ from Lemma 6.4 accepting (images of) maximally defined TCMSCs. We view $\mathcal{A}' = \mathcal{A} \cap \mathcal{B}$ as an ECMPA with $\mathrm{Int}(\mathcal{A} \cap \mathcal{B}) \subseteq \mathcal{S}$, hence $\mathcal{S}$ refines Int$(\mathcal{A} \cap \mathcal{B})$. Now we claim that this $\mathcal{A}'$ is, in fact, the ECMPA that we require. That is, we will show that $\mathcal{L}_{time}(\varphi) = \mathcal{L}_{time}(\mathcal{A}')$, thus completing the proof the theorem.

The proof follows the scheme depicted in Fig. 8. We start by showing that $\mathcal{L}_{time}(\varphi) \subseteq \mathcal{L}_{time}(\mathcal{A} \cap \mathcal{B})$. Let $T \models \varphi$. Since $\mathcal{S}$ is a proper interval set that refines Int(φ), we can apply Lemma 6.3 to get $\mathfrak{M}_T^{\mathcal{S}} \models \varphi$. Since, $\mathcal{S}$ is proper, it is finite and hence we use Lemma 6.5 to obtain $\mathcal{U}(\mathfrak{M}_T^{\mathcal{S}}) \models \varphi^{\mathcal{S}}$. But $\mathcal{L}_{MSC}(\varphi^{\mathcal{S}}) = \mathcal{L}_{MSC}(\mathcal{A})$ and $\mathfrak{M}_T^{\mathcal{S}}$ is maximally defined, so we have $\mathcal{U}(\mathfrak{M}_T^{\mathcal{S}}) \in \mathcal{L}_{MSC}(\mathcal{A} \cap \mathcal{B})$. As $\mathcal{S}$ is finite and $\mathfrak{M}_T^{\mathcal{S}}$ is a TCMSC over $(Act, \mathcal{S})$, we can then use Lemma 6.6 to conclude that $\mathfrak{M}_T^{\mathcal{S}} \in \mathcal{L}_{TC}(\mathcal{A} \cap \mathcal{B})$. Then, we can use Lemma 6.2 to obtain $T \in \mathcal{L}_{time}(\mathcal{A} \cap \mathcal{B})$.

Conversely, we show that $\mathcal{L}_{time}(\varphi) \supseteq \mathcal{L}_{time}(\mathcal{A} \cap \mathcal{B})$. Let $T \in \mathcal{L}_{time}(\mathcal{A} \cap \mathcal{B})$. Then as $\mathcal{S}$ is a proper interval set, and $\mathcal{A} \cap \mathcal{B}$ is an MPA over $Act \times [\mathrm{TC} \dashrightarrow \mathcal{S}]$, by Lemma 6.7, there exists

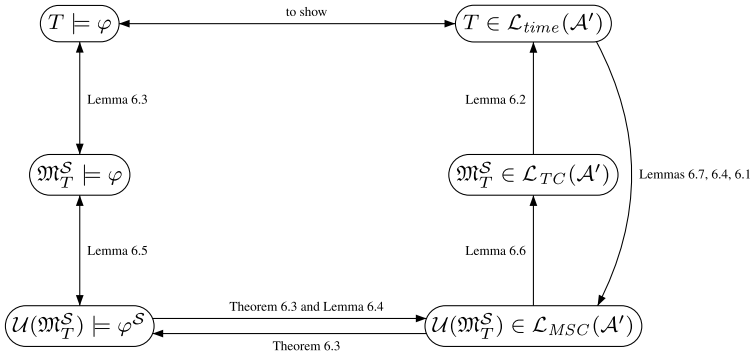


Fig. 8 Proof schematic for $2 \Rightarrow 1$ direction of Theorem 6.1 (*arrows* depict implication)

a TCMSC $\mathfrak{M}$ over $(Act, \mathcal{S})$ such that $\mathcal{U}(\mathfrak{M}) \in \mathcal{L}_{MSC}(\mathcal{A} \cap \mathcal{B})$ and T realizes $\mathfrak{M}$. Using Lemmas 6.1 and 6.4 we deduce that $\mathfrak{M} = \mathfrak{M}_T^{\mathcal{S}}$. But $\mathcal{L}_{MSC}(\varphi^{\mathcal{S}}) = \mathcal{L}_{MSC}(\mathcal{A})$ implies $\mathcal{U}(\mathfrak{M}_T^{\mathcal{S}}) \models \varphi^{\mathcal{S}}$. Now, by Lemma 6.5 we have $\mathfrak{M}_T^{\mathcal{S}} \models \varphi$. Finally, by Lemma 6.3 we conclude that $T \models \varphi$. □

Along the lines of the proof above, we can also get a characterization of the full TMSO. However, we have to restrict to untimed-existentially-bounded TMSCs.

Proof of Theorem 6.2 Let $B \in \mathbb{N}_{>0}$ and $\mathcal{L}$ be a set of $\exists^u$-B-bounded TMSCs over Act.

$(1 \Rightarrow 2)$ Let $\mathcal{A}$ be an ECMPA with $\mathcal{L}_{time}(\mathcal{A}) = \mathcal{L}$. By Theorem 6.1, we obtain an ETMSO$(Act, \mathfrak{R}_{<})$ formula φ such that $\mathcal{L}_{time}(\mathcal{A}) = \mathcal{L}_{time}(\varphi)$ and by Proposition 5.1(2), we obtain an ETMSO$(Act, \mathfrak{R}_{\leq})$ formula ψ such that $\mathcal{L}_{time}(\psi) = \mathcal{L}_{time}(\varphi)$ which completes the proof in this direction.

$(2 \Rightarrow 1)$ Let φ be a TMSO$(Act, \mathfrak{R}_{\leq})$ formula with $\mathcal{L}_{time}(\varphi) = \mathcal{L}$ and let $\mathcal{S}$ be a proper interval set which refines $\mathrm{Int}(\varphi)$. Fix an alphabet $\Gamma = Act \times [\mathrm{TC} \dashrightarrow \mathcal{S}]$. Thus, we have $\varphi^{\mathcal{S}} \in \mathrm{MSO}(\Gamma, \mathfrak{R}_{\leq})$ using the translation from Sect. 6.2. Now from Proposition 5.1(3), we obtain a formula $\gamma_B \in \mathrm{MSO}(\Gamma, \mathfrak{R}_{\leq})$ such that $\mathcal{L}_{MSC}(\gamma_B)$ is the set of all $\exists$-B-bounded MSCs over Γ. Then, indeed $\mathcal{L}_{MSC}(\varphi^{\mathcal{S}} \wedge \gamma_B)$ is an $\exists$-B-bounded set of MSCs over Γ and so, we apply Theorem 6.4 to obtain an MPA $\mathcal{A}$ over Γ such that $\mathcal{L}_{MSC}(\varphi^{\mathcal{S}} \wedge \gamma_B) = \mathcal{L}_{MSC}(\mathcal{A})$. Consider again the MPA $\mathcal{B}$ over Γ from Lemma 6.4. We view $\mathcal{A} \cap \mathcal{B}$ as an ECMPA and $\mathcal{S}$ refines $\mathrm{Int}(\mathcal{A} \cap \mathcal{B})$. We show that $\mathcal{L}_{time}(\varphi) = \mathcal{L}_{time}(\mathcal{A} \cap \mathcal{B})$.

The proof follows the same basic structure as depicted in Fig. 8. However, to capture full MSO (and not just its existential fragment), we use Theorem 6.4 in the place of Theorem 6.3, and to do this, we replace $\varphi^{\mathcal{S}}$ by $\varphi^{\mathcal{S}} \wedge \gamma_B$.

First observe that if T realizes $\mathfrak{M}$, then T is $\exists^u$-B-bounded iff $\mathcal{U}(\mathfrak{M})$ is $\exists$-B-bounded. Indeed a (untimed) linearization of a TMSC only depends on its set of events and the partial order, which are the same for T, $\mathfrak{M}$ and $\mathcal{U}(\mathfrak{M})$.

Thus, let us consider $T \in \mathcal{L}_{time}(\varphi)$. Then, T is $\exists^u$-B-bounded and $T \models \varphi$. By Lemma 6.3, we obtain that $\mathfrak{M}_T^{\mathcal{S}} \models \varphi$. Again since $\mathcal{S}$ is proper, it is finite and hence by Lemma 6.5 we get $\mathcal{U}(\mathfrak{M}_T^{\mathcal{S}}) \models \varphi^{\mathcal{S}}$. Since T realizes $\mathfrak{M}_T^{\mathcal{S}}, \mathcal{U}(\mathfrak{M}_T^{\mathcal{S}})$ is $\exists$-B-bounded, and so we get $\mathcal{U}(\mathfrak{M}_T^{\mathcal{S}}) \models \gamma_B$. Thus we conclude that $\mathcal{U}(\mathfrak{M}_T^{\mathcal{S}}) \in \mathcal{L}_{MSC}(\varphi^{\mathcal{S}} \wedge \gamma_B) = \mathcal{L}_{MSC}(\mathcal{A})$. Since $\mathfrak{M}_T^{\mathcal{S}}$ is maximally defined, we obtain $\mathcal{U}(\mathfrak{M}_T^{\mathcal{S}}) \in \mathcal{L}_{MSC}(\mathcal{A} \cap \mathcal{B})$. Using Lemma 6.6 we get $\mathfrak{M}_T^{\mathcal{S}} \in \mathcal{L}_{TC}(\mathcal{A} \cap \mathcal{B})$. Finally, Lemma 6.2 implies $T \in \mathcal{L}_{time}(\mathcal{A} \cap \mathcal{B})$.

Conversely, let $T \in \mathcal{L}_{time}(\mathcal{A} \cap \mathcal{B})$. Then as $\mathcal{S}$ is a proper interval set, and $\mathcal{A} \cap \mathcal{B}$ is an MPA over $Act \times [\mathrm{TC} \dashrightarrow \mathcal{S}]$, by Lemma 6.7, there exists a TCMSC $\mathfrak{M}$ over $(Act, \mathcal{S})$ such

that $\mathcal{U}(\mathfrak{M}) \in \mathcal{L}_{MSC}(\mathcal{A} \cap \mathcal{B})$ and T realizes $\mathfrak{M}$. Using Lemmas 6.1 and 6.4 we deduce that $\mathfrak{M} = \mathfrak{M}_T^S$. But $\mathcal{L}_{MSC}(\varphi^S \wedge \gamma_B) = \mathcal{L}_{MSC}(\mathcal{A})$ implies $\mathcal{U}(\mathfrak{M}_T^S) \models \varphi^S$ and by Lemma 6.5 we get $\mathfrak{M}_T^S \models \varphi$. Finally, Lemma 6.3 implies $T \models \varphi$. □

7 Checking emptiness of ECMPAs

In this section, we investigate emptiness checking for ECMPAs, leading to a partial solution to the satisfiability problem for TMSO formulas, which is undecidable in its full generality. Since the MSCs are used in early protocol design, this problem is of vital interest as it allows detection of possible design failures at this stage.

Theorem 7.1 *The following problem*:

INPUT: *An ECMPA* $\mathcal{A} = (\{\mathcal{A}_p\}_{p \in Proc}, Act, \Delta, F)$, *and an integer* $B > 0$.
QUESTION: *Does there exist* $T \in \mathcal{L}_{time}(\mathcal{A})$ *such that* T *has a* B*-bounded timed linearization?*

is decidable in space $P(|\mathcal{A}|, (B+1)^{|Act|})$ *for some polynomial* P.

Note that the size of $\mathcal{A}$ also depends on Act via the transition relations $\rightarrow_p$ of the components $\mathcal{A}_p$. But the space complexity above is only exponential in $|Act|$ and not in the much bigger size $|\mathcal{A}|$.

Then, using Theorem 6.2 we can conclude that,

Corollary 7.1 *The following problem is decidable*:

INPUT: *A TMSO formula* φ *and an integer* $B > 0$.
QUESTION: *Does there exist* $T \in \mathcal{L}_{time}(\varphi)$ *such that* T *has a* B*-bounded timed linearization?*

The rest of the section formulates the proof of the above theorem. Let $\mathcal{A}$ be an ECMPA and let $B > 0$. We construct a (finite) *timed automaton* $\mathcal{B}$ (denoted TA) that accepts a timed word w over Act iff w is a B-bounded timed linearization of some TMSC accepted by $\mathcal{A}$. Since emptiness is decidable for finite timed automata [2], we are done.

The remainder of this section is dedicated to the construction of such a $\mathcal{B}$, which is done in three steps as sketched below:

- First, we address the main hurdle in simulating an ECMPA by a TA, namely, a run of a TA is totally ordered, while ECMPAs have partially ordered runs. Hence, to keep track of clock constraints used in the ECMPA, the TA needs to recover the partial order information from its runs, i.e., words. This is done using gadgets that we will define as our first step.
- Next, using these gadgets, we describe an infinite TA which simulates the ECMPA.
- Though we allow infinitely many clocks and states in this intermediate construction, on any run we will see that only finitely many states and clocks are used. We modify this automaton to obtain finitely many states and clocks thus completing our construction.

7.1 Recovering the partial order

We begin by introducing some notations that will be used to classify the set of actions. For any channel $(p, q) \in Ch$, and each $\theta \in \{!, ?\}$, $p\theta q$ denotes the set of actions $\{p\theta q(m) \in Act \mid m \in \mathcal{M}\}$. Recall that, we write $a \in p!q$ (respectively, $a \in p?q$) if $a = p!q(m)$ (respectively, $p?q(m)$) for some $m \in \mathcal{M}$.

Now, recall that the partial order of an MSC can be recovered from any of its linearizations under the FIFO assumption. Indeed, if $w = a_1 \dots a_n \in Act^*$ is a linearization of MSC $M = (E, \leq^M, \lambda)$ over Act, then M is isomorphic to the unique MSC $M^w = (E^w, \leq^{M^w}, \lambda^w)$ over Act, where $E^w = \{1, \dots, n\}$ (i.e., the set of positions of the word w), $\lambda^w(i) = a_i$, and $\leq^{M^w}$ is defined as the reflexive transitive closure of $\bigcup_{p,q \in Proc} <_{pq}^{M^w}$ where, for all $p \in Proc$, $<_{pp}^{M^w}$ is the set of pairs $(i, j) \in E^w \times E^w$ such that $i < j$ and $\lambda^w(i), \lambda^w(j) \in Act_p$, and for all $(p, q) \in Ch$, $<_{pq}^{M^w}$ is the set of pairs $(i, j) \in E^w \times E^w$ such that $\lambda^w(i) = p!q(m)$ and $\lambda^w(j) = q?p(m)$ for some $m \in \mathcal{M}$ and $|\{k \leq i \mid \lambda^w(k) \in p!q\}| = |\{k \leq j \mid \lambda^w(k) \in q?p\}|$.

Therefore, we can consider the partial order relation of M to be a relation over the positions of a given linearization of M. Thus, given a linearization w of an MSC M, we can identify M with M^w and write $i \leq^M j$ for $1 \leq i, j \leq |w|$, to mean $i \leq^{M^w} j$, i.e., the isomorphic images of positions i, j are related by $\leq^M$. Similarly, we may write $i <_{pp}^M j$, $i <_{pq}^M j$, $(i, j) \in \mathrm{Prev}_a^M$ and $(i, j) \in \mathrm{Next}_a^M$ for $a \in Act$ to mean that the corresponding events are related in M, respectively by, the local-process relation $<_{pp}^M$, the message relation $<_{pq}^M$, the previous and the next occurrence of a relations. In the same way, we can also write $\lambda(i)$ instead of $\lambda^w(i)$. Note that $i \leq^M j$ implies $i \leq j$ but the converse need not be true (where $\leq$ is the usual ordering between i and j as elements of $\mathbb{N}$). However, if $\lambda(i) = \lambda(j)$, then $i \leq j$ implies $i \leq^M j$.

Now, we describe gadgets, which are deterministic finite-state automata that run on words which are linearizations of an MSC and accept if the first and last position of the word are related under the partial order. For this, we restrict to B-bounded linearizations.

Definition 7.1 Let $M = (E, \leq, \lambda)$ be an MSC over Act and $B \in \mathbb{N}_{>0}$. Then a *B-well-stamping for M* is a map $\rho : E \to \{0, \dots, B-1\}$ s.t., for any $e \in E$ with $\lambda(e) = p\theta q(m)$ for some $p, q \in Proc$, $m \in \mathcal{M}$, $\theta \in \{!, ?\}$, we have $\rho(e) = |{\downarrow}e \cap \bigcup_{m' \in \mathcal{M}} E_{p\theta q(m')}| \bmod B$.

Then, by the above considerations, given a linearization w of M, ρ is also a map from the positions of w to $\{0, \dots, B-1\}$. Now, the following property is immediate from the definitions of the message relation in an MSC (Sect. 2) and B-boundedness.

Proposition 7.1 *Let $w = a_1 \dots a_n \in Act^*$ be a B-bounded linearization of an MSC M over Act and ρ be a B-well-stamping for M. Then, for $1 \leq i < j \leq n$ and $(p, q) \in Ch$, we have $i <_{pq}^M j$ iff the following conditions hold*:

(a) $a_i \in p!q$, $a_j \in q?p$,
(b) $\rho(i) = \rho(j)$,
(c) *for all k, $i < k < j$, $a_k \in q?p$ implies $\rho(k) \neq \rho(i)$.*

Proof First, for any $i \in \{1, \dots, n\}$, if $a_i = p\theta q(m)$ for some $\theta \in \{!, ?\}$, $m \in \mathcal{M}$, we fix the notation ${\Downarrow}i = ({\downarrow}i \cap \bigcup_{m' \in \mathcal{M}} E_{p\theta q(m')}) = \{i' \leq i \mid \lambda(i') \in p\theta q\}$. Now, the proof follows once we recall the relevant definitions in this setting.

(1) For $1 \leq i < j \leq n$, $i <_{pq}^M j$ iff $\lambda(i) = p!q(m)$, $\lambda(j) = q?p(m)$ for some $m \in \mathcal{M}$ and $|{\Downarrow}i| = |{\Downarrow}j|$.

(2) For all $1 \leq i \leq n$, $\rho(i) = |\Downarrow i| \bmod B$.
(3) w is B-bounded iff for all $1 \leq \ell \leq n$, $(p, q) \in Ch$, $|\{i' \mid i' \leq \ell, \lambda(i') \in p!q\}| - |\{j' \mid j' \leq \ell, \lambda(j') \in q?p\}| \leq B$.

Now, for the first direction, let $i <^{M}_{pq} j$. Then, by (1), (a) holds and $|\Downarrow i| = |\Downarrow j| = \alpha \cdot B + b$. Then, $b = \rho(i) = \rho(j)$ and so (b) holds. Now, suppose (c) did not hold. Then there exists k, $i \leq k < j$ such that $a_k \in q?p$ and $\rho(k) = \rho(i) = \rho(j)$. Then, by (2), $|\Downarrow k| = \alpha' \cdot B + b$. But then $\Downarrow k \subseteq \Downarrow j$ and $j \in \Downarrow j$, $j \notin \Downarrow k$ implies that $|\Downarrow j| > |\Downarrow k| \Rightarrow \alpha \cdot B + b > \alpha' \cdot B + b \Rightarrow (\alpha - \alpha') \cdot B > 0$. Thus $|\Downarrow j| - |\Downarrow k| \geq B$ and so $|\Downarrow i| - |\Downarrow k| \geq B$. But now, $i < k$ and $\lambda(k) \in q?p$ implies that $|\Downarrow k| \geq |\{i' \leq i \mid \lambda(i') \in q?p\}| + 1$ and so we can conclude that $|\Downarrow i| - |\{i' \leq i \mid \lambda(i') \in q?p\}| > B$. This means $|\{i' \leq i \mid \lambda(i') \in p!q\}| - |\{i' \leq i \mid \lambda(i') \in q?p\}| > B$ and so by (3) (in particular, by letting $\ell = i$) it contradicts the B-boundedness of w. Thus (c) also holds.

Conversely, let conditions (a), (b) and (c) be true. Then, (a) and (b) together imply that $|\Downarrow i| \bmod B = |\Downarrow j| \bmod B$. That is $|\Downarrow i| = \alpha \cdot B + b$ and $|\Downarrow j| = \alpha' \cdot B + b$ for some $\alpha, \alpha' \in \mathbb{N}$. We now show that $\alpha = \alpha'$. First, observe that between i and j, there can be at most $B - 1$ actions labeled $q?p(m')$ for some $m' \in \mathcal{M}$. Otherwise, we can find an event k, $i < k < j$ such that $a_k \in q?p$ and $|\Downarrow k| = (\alpha' - 1)B + b$. And for this k, $\rho(k) = \rho(j)$ which contradicts condition (c). Also, we have $i < j$ and $a_j \in q?p$. Thus, $(\alpha' - 1) \cdot B + b < |\{j' \leq i \mid a_i \in q?p\}| < \alpha' \cdot B + b$. Now, if $\alpha > \alpha'$, then $|\{i' \leq i \mid \lambda(i') \in p!q\}| - |\{j' \leq i, \lambda(j') \in q?p\}| > \alpha \cdot B - \alpha' \cdot B > B$ which contradicts B-boundedness of w by (3). Else if $\alpha' > \alpha$, then $|\{j' \leq i, \lambda(j') \in q?p\}| - |\{i' \leq i \mid \lambda(i') \in p!q\}| > (\alpha' - 1 - \alpha) \cdot B \geq 0$ which is a contradiction since in a linearization we cannot have the number of receives exceeding the number of sends. Thus $\alpha = \alpha'$. Now since number of $p!q$ events below i is equal to the number of $q?p$ events below j, we conclude by FIFO property that j is the matching receive for i, i.e., $a_i = p!q(m)$ and $a_j = q?p(m)$ for some $m \in \mathcal{M}$. Thus, by (1), $i <^{M}_{pq} j$. □

Constructing the gadgets The first gadget we build is a deterministic finite-state automaton $\mathcal{C}^{\leq}$ over $Act \times \{0, \ldots, B - 1\}$, that detects if the first and last positions of a word, provided the word corresponds to some factor of a B-bounded MSC linearization, are related with respect to the associated partial ordering. In what follows, we let *Send* denote the set of symbols $\{p!q \mid (p, q) \in Ch\}$.

We define $\mathcal{C}^{\leq} = (Q^{\leq}, \delta^{\leq}, s_0^{\leq}, F^{\leq})$ where a state of $Q^{\leq}$ is a triple of the form (P, O, f) where $f \in \{0, 1\}$, $P \in 2^{Proc}$ and $O : Send \dashrightarrow \{0, \ldots, B - 1\}$ s.t., if $p!q \in \mathrm{dom}(O)$ for some $(p, q) \in Ch$, then $p \in P$. Intuitively, the triple (P, O, f) represents the information we need to recover the partial order. The set P contains the processes in the future partial view, O indicates the stamp/numbering of each "open send", and f will be 1 iff the first and the last position are in fact related as desired. Note that, letting $n = |Proc|$, we have

$$\left|Q^{\leq}\right| = 2 \cdot \sum_{k=0}^{n} \left(\binom{n}{k} \cdot (B + 1)^{k(n-1)} \right) = \mathcal{O}\left((B + 1)^{|Proc|^2}\right) \tag{11}$$

The initial state of $\mathcal{C}^{\leq}$ is $s_0^{\leq} = (\emptyset, \emptyset, 0) \in Q^{\leq}$ and the set of final states $F^{\leq} = \{(P, O, f) \in Q^{\leq} \mid f = 1\}$. The transition function $\delta^{\leq} : Q^{\leq} \times (Act \times \{0, \ldots, B - 1\}) \to Q^{\leq}$ is defined as $\delta^{\leq}((P, O, f), (a, \beta)) = (P', O', f')$ where

$$P' = \begin{cases} P \cup \{p\} & \text{if } a \in Act_p \text{ and either } (P = \emptyset) \text{ or } (\exists q \in P \text{ such that,} \\ & a \in p?q \text{ and } O(q!p) = \beta) \\ P & \text{otherwise} \end{cases}$$

$$O' = \begin{cases} O[p!q \mapsto \beta] & \text{if } a \in p!q \notin \mathrm{dom}(O) \text{ for some } (p,q) \in Ch \text{ and} \\ & \text{either } p \in P \text{ or } P = \emptyset \\ O & \text{otherwise} \end{cases}$$

$$f' = \begin{cases} 1 & \text{if } a \in Act_p \text{ for } p \in P' \\ 0 & \text{otherwise} \end{cases}$$

As usual, a run of $\mathcal{C}^{\leq}$ on $(a_1, \beta_1) \ldots (a_n, \beta_n) \in (Act \times \{0, \ldots, B-1\})^*$ is a sequence $s_0^{\leq} \xrightarrow{(a_1,\beta_1)} \ldots \xrightarrow{(a_n,\beta_n)} s_n^{\leq}$, where $s_i^{\leq} \in Q^{\leq}$ for each $i \in \{0, \ldots, n\}$ and $(s_i^{\leq}, a_{i+1}, \beta_{i+1}, s_{i+1}^{\leq}) \in \delta^{\leq}$ for all $i \in \{0, \ldots, n-1\}$ and is accepting if $s_n^{\leq} \in F^{\leq}$. Then, $\mathcal{L}(\mathcal{C}^{\leq})$ denotes the set of words over $Act \times \{0, \ldots, B-1\}$ having accepting runs. Note that with this definition, $\mathcal{C}^{\leq}$ has a run (which may be accepting or non-accepting) on every word of $(Act \times \{0, \ldots, B-1\})^*$. For linearizations, $\mathcal{C}^{\leq}$ has the desired property:

Lemma 7.1 *Let $w = a_1 \ldots a_n \in Act^*$ be a B-bounded linearization of an MSC M over Act and ρ be a B-well-stamping for M. Then, for $1 \leq i \leq j \leq n$, $(a_i, \rho(i)) \ldots (a_j, \rho(j)) \in \mathcal{L}(\mathcal{C}^{\leq})$ iff $i \leq^M j$.*

Proof Let us fix $1 \leq i \leq n$ and let $a_i \in Act_p$ for some $p \in Proc$. For each j such that $i \leq j \leq n$, we denote $w_{ij}^{\rho} = (a_i, \rho(i)) \ldots (a_j, \rho(j))$. Then for each j, $i \leq j \leq n$, there is a run r_{ij} of $\mathcal{C}^{\leq}$ on w_{ij}^{ρ}, namely, $s_0^{\leq} \xrightarrow{(a_i,\rho(i))} (P_i, O_i, f_i) \ldots \xrightarrow{(a_j,\rho(j))} (P_j, O_j, f_j)$ where $s_0^{\leq} = (\emptyset, \emptyset, 0)$ and for each $k \in \{i, \ldots, j\}$, $(P_k, O_k, f_k) \in Q^{\leq}$ is given by the definition of $\mathcal{C}^{\leq}$. Indeed each r_{ij} is a prefix of $r_{ij'}$ for $j \leq j' \leq n$. For simplicity of notation, let $P_{i-1} = \emptyset$, $O_{i-1} = \emptyset$, $f_{i-1} = 0$, i.e., $s_0^{\leq} = (P_{i-1}, O_{i-1}, f_{i-1})$. Now by definition of $\mathcal{C}^{\leq}$, $P_{i-1} = \emptyset$, $a_i \in Act_p$ implies that $P_i = \{p\}$. Once a process gets into P it is never removed, and so, $p \in P_j$ for all $i \leq j \leq n$. Also for each j, $i \leq j \leq n$, r_{ij} is an accepting run iff $f_j = 1$ i.e., iff $a_j \in Act_q$ for some $q \in Proc$ and $q \in P_j$.

We now distinguish two cases. First, suppose $a_j \in Act_p$. Then we already have $p \in P_j$ and so r_{ij} is accepting. Also in this case, a_i, a_j belong to the same process p and so $i \leq_{pp}^M j$ which implies $i \leq^M j$. Thus,

$$\forall j \in \{i, \ldots, n\}, a_j \in Act_p \quad \Longrightarrow \quad (r_{ij} \text{ is accepting}) \wedge \big(i \leq^M j\big). \tag{12}$$

On the other hand, for all $j \in \{i+1, \ldots, n\}$ such that $a_j \in Act_q$ for some $q \neq p$, we will show that r_{ij} is an accepting run iff $i \leq^M j$ which completes the proof of the lemma. We proceed by induction on $j - i$.

The base case is when $j - i = 1$, i.e. $w_{ij}^{\rho} = (a_i, \rho(i))(a_j, \rho(j))$. Then, $P_{j-1} = P_i = \{p\}$. Now, since $a_j \in Act_q$, r_{ij} is accepting iff $q \in P_j$. By definition of $\mathcal{C}^{\leq}$, this happens iff $a_j \in q?p$ and $O_{j-1}(p!q) = \rho(j)$. But since $j - 1 = i$ we have $p!q \in \mathrm{dom}(O_i)$ and along with $O_{i-1} = \emptyset$, this implies that $a_i \in p!q$ and $O_i(p!q) = \rho(i)$. Thus we can conclude that $q \in P_j$ iff $a_i \in p!q, a_j \in q?p$ and $\rho(i) = \rho(j)$ which by Proposition 7.1 happens iff $i <_{pq}^M j$ (by noting that there is no event between i and j). But again $i + 1 = j$ and i, j are not on the same process, thus, $i <_{pq}^M j$ iff $i \leq^M j$, which completes the proof of the base case.

Now, suppose $j - i > 1$. Then, r_{ij} is accepting implies $q \in P_j$. But since $q \notin P_i = \{p\}$, there exists j' for $i < j' \leq j$ such that $q \notin P_{j'-1}, q \in P_{j'}$. By the definition of $\mathcal{C}^{\leq}$, since $P_{j'-1} \neq \emptyset$, there is $p' \in Proc$ such that $a_{j'} \in q?p'$ and $O_{j'-1}(p'!q) = \rho(j')$. But since $O_{i-1} = \emptyset$, there is k, $i \leq k < j'$ such that $p'!q \in \mathrm{dom}(O_k)$ and $p'!q \notin \mathrm{dom}(O_{k-1})$. Then at k, $a_k \in p'!q$ and $O_k(p'!q) = \rho(k)$ and either $p' \in P_{k-1}$ or $k = i$ and $P_{k-1} = \emptyset$. In both cases,

$p' \in P_k$ since $a_k \in Act_{p'}$. Thus, in fact $f_k = 1$ and so r_{ik} is an accepting run of $\mathcal{C}^{\leq}$ on w_{ik}^{ρ}. Now, if $p' = p$, i.e., $a_k \in Act_p$, by (12), we conclude that $i \leq^M k$. Otherwise, $a_k \in Act_{p'}$, $p' \neq p$ and since, $k < j' \leq j$, we can apply the induction hypothesis to conclude that $i \leq^M k$. Now, the values in O, once defined, are never modified and so $O_k(p'!q) = O_{j'-1}(p'!q)$ which implies $\rho(k) = \rho(j')$. Also for any j'', $k \leq j'' < j'$, if $a_{j''} \in q?p'$ and $\rho(j'') = \rho(j')$, then at j'', we have $O_{j''}(p'!q) = \rho(k) = \rho(j'')$ and so $q \in P_{j''}$. This is a contradiction since we assumed that $q \in P_{j'-1}$. Thus, by Proposition 7.1, $k <^M_{p'q} j'$. Also $a_{j'}, a_j \in Act_q$ implies that $j' \leq^M_{qq} j$. Thus we have $i \leq^M k <^M_{p'q} j' \leq^M_{qq} j$ and so by definition of $\leq^M$, we conclude that $i \leq^M j$.

Conversely, let $i \leq^M j$, be such that i is a p-event and j is a q-event for $p \neq q$. Then let j' be the earliest event on process q which is related to i. This implies that $a_{j'}$ is a receive action from some process, say $p' \in Proc$. In other words, there exists k, j, such that $i \leq k < j' \leq j$ and $i \leq^M k <^M_{p'q} j' \leq^M_{qq} j$ and for all j'' with $j'' <^M_{qq} j'$, $i \not\leq^M j''$, where $a_k = p'!q(m)$ and $a_{j'} = q?p'(m)$ for some $p' \in Proc$, $m \in \mathcal{M}$. Then, if $p' = p$, then by (12), and otherwise by induction hypothesis, r_{ik} is an accepting run of $\mathcal{C}^{\leq}$ on w_{ik}. Thus $f_k = 1$ and so $p' \in P_k$. But since, $a_k \in p'!q$, we find that either $P_k = P_{k-1}$ or $k = i$. Thus either $p' \in P_{k-1}$ or $P_{k-1} = \emptyset$. Also, $p'!q \notin \mathrm{dom}(O_{k-1})$, for otherwise, we can find k', $i \leq k' < k$ with $a_{k'} = p'!q$, whose corresponding receive contradicts the minimality of j' under FIFO condition. So by definition of O, we have $O_k(p'!q) = \rho(k)$ and by Proposition 7.1, $\rho(k) = \rho(j')$. Thus, by definition of $\mathcal{C}^{\leq}$, we have $P_{j'} = P_{j'-1} \cup \{q\}$. Thus, $q \in P_{j'} \subseteq P_j$ and along with $a_j \in Act_q$, we get that $f_j = 1$ and so r_{ij} is an accepting run. □

An ECMPA uses Prev_a and Next_a in the guards, which along a run constrain the previous and next occurrence of an action a respectively. Hence, we need to recover not only the general partial order relation but also these previous and next occurrence relations from the linearization. For this, we build two more gadgets using the gadget described above.

We begin by defining a deterministic finite-state automaton $\mathcal{C}^{\triangleleft} = (Q^{\triangleleft}, \delta^{\triangleleft}, s_0^{\triangleleft}, F^{\triangleleft})$, which we refer to as the $\mathcal{C}^{\triangleleft}$ *gadget* and use to recover the previous occurrence relation. Its state space is given by $Q^{\triangleleft} = Q^{\leq} \times Q^{\leq} \times Act$. Thus, using (11) we obtain

$$\left|Q^{\triangleleft}\right| = \mathcal{O}\big((B+1)^{2|Proc|^2} \times |Act|\big). \tag{13}$$

Now, the idea is that the first component, mimicking the automaton $\mathcal{C}^{\leq}$, is started when reading the first occurrence of an action a, which is henceforth stored in the third component. The second component is run if and when a is executed for the second time. Any further event that is related in the partial order to the first occurrence of a but not to the second occurrence matches the Prev_a relation so that $F^{\triangleleft} = F^{\leq} \times (Q^{\leq} \setminus F^{\leq}) \times Act$. We let $s_0^{\triangleleft} = (s_0^{\leq}, s_0^{\leq}, b)$ for some arbitrary action $b \in Act$. Finally, for any $(s_1, s_2, a) \in Q^{\triangleleft}$, $b \in Act$, and $n \in \{0, \ldots, B-1\}$, we set

$$\delta^{\triangleleft}\big((s_1, s_2, a), (b, n)\big) = \begin{cases} (\delta^{\leq}(s_1, (b, n)), s_2, b) & \text{if } s_1 = s_0^{\leq} \\ (\delta^{\leq}(s_1, (b, n)), s_2, a) & \text{if } s_1 \neq s_0^{\leq},\ s_2 = s_0^{\leq} \\ & \text{and } b \neq a \\ (\delta^{\leq}(s_1, (b, n)), \delta^{\leq}(s_2, (b, n)), a) & \text{otherwise} \end{cases}$$

Similarly, the $\mathcal{C}^{\triangleright}$ *gadget* is defined as $\mathcal{C}^{\triangleright} = (Q^{\triangleright}, \delta^{\triangleright}, s_0^{\triangleright}, F^{\triangleright})$ with $Q^{\triangleright} = Q^{\leq} \times 2^{Act} \times \{0, 1\}$. Again by (11) we obtain,

$$\left|Q^{\triangleright}\right| = \mathcal{O}\big((B+1)^{|Proc|^2} \times 2^{|Act|}\big) \tag{14}$$

The idea here is that the second component of a state keeps track of the actions we have seen so far in the future of the first action, and the third component indicates a final state. Accordingly, $s_0^{\triangleright} = (s_0^{\leq}, \emptyset, 0)$, $F^{\triangleright} = F^{\leq} \times 2^{Act} \times \{1\}$ and for any $(s, A, f) \in Q^{\triangleright}$, $a \in Act$, $n \in \{0, \ldots, B-1\}$, we set

$$\delta^{\triangleright}\big((s, A, f), (a, n)\big) = \begin{cases} (\delta^{\leq}(s, (a, n)), A \cup \{a\}, 1) & \text{if } \delta^{\leq}(s, (a, n)) \in F^{\leq} \\ & \text{and } a \notin A \\ (\delta^{\leq}(s, (a, n)), A, 0) & \text{otherwise} \end{cases}$$

Then, the following lemma describes the *nice* property of the above gadgets.

Lemma 7.2 *Let $w = a_1 \ldots a_n \in Act^*$ be a B-bounded linearization of an MSC M over Act and ρ be a B-well-stamping for M. Then, for all $1 \leq i \leq j \leq n$, letting $w_{ij}^{\rho} = (a_i, \rho(i)) \ldots (a_j, \rho(j)) \in (Act \times \{0, \ldots, B-1\})^*$, we have* (1) *$w_{ij}^{\rho} \in \mathcal{L}(\mathcal{C}^{\triangleleft})$ iff $(j, i) \in \mathrm{Prev}_{a_i}^{M}$ and* (2) *$w_{ij}^{\rho} \in \mathcal{L}(\mathcal{C}^{\triangleright})$ iff $(i, j) \in \mathrm{Next}_{a_j}^{M}$.*

Again, recalling that there is an isomorphism that maps positions of w to events of M, in the above statement $(j, i) \in \mathrm{Prev}_{a_i}^{M}$ means that the events in M corresponding to the positions i, j are related by $\mathrm{Prev}_{a_i}^{M}$ and so on.

Proof (1) By definition of $\mathcal{C}^{\triangleleft}$, there is a run r of $\mathcal{C}^{\triangleleft}$ on w_{ij}^{ρ}, namely, $s_0^{\triangleleft} = s_{i-1}^{\triangleleft} \xrightarrow{(a_i, \rho(i))} s_i^{\triangleleft} \cdots \xrightarrow{(a_j, \rho(j))} s_j^{\triangleleft}$ where for all k, $i-1 \leq k \leq j$, $s_k^{\triangleleft} = (s_k^{\leq}, s_k'^{\leq}, b_k)$. Further note that, $s_{i-1}^{\leq} = s_0^{\leq}$ and $s_i'^{\leq} = s_{i-1}'^{\leq} = s_0^{\leq}$ and for all k, $i \leq k \leq j$, $b_k = a_i$. Now $s_j'^{\leq} \in F^{\leq}$ implies there is k, $i < k \leq j$ such that $s_{k+1}'^{\leq} \neq s_0^{\leq}$ which means that $a_k = b_k$. But $b_k = a_i$ and so $a_k = a_i$. Also, for this k, $s_k'^{\leq} \xrightarrow{(a_k, \rho(k))} \cdots \xrightarrow{(a_j, \rho(j))} s_j'^{\leq}$ is an accepting run of $\mathcal{C}^{\leq}$ on w_{kj}^{ρ}. Conversely, if $\exists k$, $i < k \leq j$ such that $a_k = a_i$ and there is an accepting run of $\mathcal{C}^{\leq}$ on w_{kj}^{ρ} then in r we find that $s_j'^{\leq} \in F^{\leq}$. Thus, we can conclude that $s_j'^{\leq} \in F^{\leq}$ iff there is k, $i < k \leq j$ such that $a_k = a_i$ and there is an accepting run of $\mathcal{C}^{\leq}$ on w_{kj}^{ρ}. But by Lemma 7.1 there is an accepting run of $\mathcal{C}^{\leq}$ on w_{kj}^{ρ} iff $k \leq^M j$. Thus $s_j'^{\leq} \in F^{\leq}$ iff there is k, $i < k \leq j$ such that $a_k = a_i$ and $k \leq^M j$. Also by Lemma 7.1 we have $s_j^{\leq} \in F^{\leq}$ iff $i \leq^M j$. Finally r is an accepting run of $\mathcal{C}^{\triangleright}$ on w_{ij}^{ρ} iff $s_j^{\triangleleft} \in F^{\triangleleft}$, i.e., $s_j^{\leq} \in F^{\leq}$ and $s_j'^{\leq} \notin F^{\leq}$. And by the above arguments, this happens iff $i \leq^M j$ and for all k, $i < k \leq j$, either $a_k \neq a_i$ or $k \not\leq^M j$. But this is exactly the definition of $(j, i) \in \mathrm{Prev}_{a_i}^{M}$. Thus $w_{ij}^{\rho} \in \mathcal{L}(\mathcal{C}^{\triangleleft})$ iff $(i, j) \in \mathrm{Prev}_{a_i}^{M}$.

(2) Similarly, there is a run r' of $\mathcal{C}^{\triangleright}$ on w_{ij}^{ρ}, namely, $s_0^{\triangleright} = s_{i-1}^{\triangleright} \xrightarrow{(a_i, \rho(i))} s_i^{\triangleright} \cdots \xrightarrow{(a_j, \rho(j))} s_j^{\triangleright}$ where for all k, $i-1 \leq k \leq j$, $s_k^{\triangleright} = (s_k^{\leq}, A_k, b_k)$ for $b_k \in \{0, 1\}$, such that $r_1' = s_{i-1}^{\leq} \xrightarrow{(a_i, \rho(i))} \cdots \xrightarrow{(a_j, \rho(j))} s_j^{\leq}$ is a run of $\mathcal{C}^{\leq}$ on w_{ij}^{ρ}. Then r' is accepting iff $s_j^{\leq} \in F^{\leq}$ and $b_j = 1$ i.e., iff r_1' is accepting and $a_j \notin A_{j-1}$. By Lemma 7.1, r_1' is accepting iff $i \leq^M j$. Also $a_j \notin A_{j-1}$ implies $a_j \notin A_k$ for all $i \leq k < j$ which means that for all k, $i \leq k < j$, $a_k \neq a_j$. Thus, we conclude that r' is accepting iff $i \leq^M j$ and for all k, $i \leq k < j$, $a_k \neq a_j$ i.e., iff $(i, j) \in \mathrm{Next}_{a_j}^{M}$. □

7.2 From ECMPA to timed automata

If we ignore the clock constraints and timing issues, then this simulation is the same as giving an alternate semantics of an MPA directly over linearizations of MSCs. However,

when we include the timing constraints of the ECMPA, then along the run of the TA we need to know when to verify these constraints and against which clocks. We maintain this information in the state space using the gadgets.

Intuitively, at each position of a run of the TA, we start a new copy of the Prev gadget and reset a corresponding clock z. Thus, at a later position, if we encounter a Prev_a constraint for some $a \in Act$, and if *this* copy of the Prev gadget is in a final state, then we know that these positions are related by the Prev_a relation, hence the constraint must be checked against clock z.

Similarly, at each position of the run where we encounter a Next_a constraint, we start a new copy of the Next gadget and reset a clock z'. Then, when it reaches an accepting state and the last transition is an a-action, we know that these positions are related by the Next_a relation. Hence, at this point we verify that clock z' satisfies the constraint mentioned when the gadget was started. However, for this we also need to maintain in the state space the constraint itself so that we can recover it and verify it at the later position.

Finally, for message constraints, we reset a clock when we encounter the send action and verify it when we reach the correct receive. This information is already contained in the state space as we will see. In addition however, we need to maintain the message constraint itself in the state space, so that when we reach the receive we know which constraint to check the clock against.

We formalize these ideas below. Let us first recall the well known notion of timed automata [2]. Unlike event-clock automata, whose clocks are implicit, timed automata have a set $\mathcal{Z}$ of explicit clocks that can be guarded by means of clock formulas. The set $\text{Form}(\mathcal{Z})$ of *clock formulas* over $\mathcal{Z}$ is given by the grammar

$$\varphi ::= \text{true} \mid x \bowtie c \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$$

where x ranges over $\mathcal{Z}$, $\bowtie \in \{<, \leq, >, \geq, =\}$, and c ranges over $\mathbb{N}$. A *clock valuation* over $\mathcal{Z}$ is a mapping $\nu : \mathcal{Z} \to \mathbb{R}_{\geq 0}$. We say that ν satisfies $\varphi \in \text{Form}(\mathcal{Z})$, written $\nu \models \varphi$, if φ evaluates to true using the values given by ν. For $R \subseteq \mathcal{Z}$, we let $\nu[R \mapsto 0]$ denote the clock valuation defined by $\nu[R \mapsto 0](x) = 0$ if $x \in R$, and $\nu[R \mapsto 0](x) = \nu(x)$, otherwise. A *timed automaton* (over Act) is a tuple $\mathcal{B} = (Q_\mathcal{B}, \mathcal{Z}_\mathcal{B}, \delta_\mathcal{B}, \iota_\mathcal{B}, F_\mathcal{B})$ where $Q_\mathcal{B}$ is its set of *states*, $\mathcal{Z}_\mathcal{B}$ is its set of *clocks*, $\delta_\mathcal{B} \subseteq Q_\mathcal{B} \times \text{Form}(\mathcal{Z}_\mathcal{B}) \times Act \times 2^{\mathcal{Z}_\mathcal{B}} \times Q_\mathcal{B}$ is the set of *transitions*, $\iota \in Q_\mathcal{B}$ is the initial state, and $F_\mathcal{B} \subseteq Q_\mathcal{B}$ is the set of *final states*. We say that $\mathcal{B}$ is *finite* if both $Q_\mathcal{B}$ and $\mathcal{Z}_\mathcal{B}$ are finite. The *language* of $\mathcal{B}$ is a set of timed words $\mathcal{L}_{tw}(\mathcal{B}) \subseteq (Act \times \mathbb{R}_{\geq 0})^*$. For $\sigma = (a_1, t_1) \ldots (a_n, t_n) \in (Act \times \mathbb{R}_{\geq 0})^*$, we let $\sigma \in \mathcal{L}_{tw}(\mathcal{B})$ iff there is an *accepting run* r of $\mathcal{B}$ on σ, i.e., there is a sequence

$$r = (\text{st}_0, \nu_0) \xrightarrow{a_1, t_1} (\text{st}_1, \nu_1) \xrightarrow{a_2, t_2} \cdots \xrightarrow{a_n, t_n} (\text{st}_n, \nu_n)$$

(where, for all $i \in \{0, \ldots, n\}$, $\text{st}_i \in Q_\mathcal{B}$ and $\nu_i : \mathcal{Z}_\mathcal{B} \to \mathbb{R}_{\geq 0}$ with $\nu_0(x) = 0$ for all $x \in \mathcal{Z}_\mathcal{B}$) such that $\text{st}_0 = \iota_\mathcal{B}$, $\text{st}_n \in F_\mathcal{B}$, and, for each $i \in \{1, \ldots, n\}$, there exist $\varphi_i \in \mathit{Form}(\mathcal{Z}_\mathcal{B})$ and $R_i \in 2^{\mathcal{Z}_\mathcal{B}}$ satisfying $(\text{st}_{i-1}, \varphi_i, a_i, R_i, \text{st}_i) \in \delta_\mathcal{B}$, $(\nu_{i-1} + t_i - t_{i-1}) \models \varphi_i$, and $\nu_i = (\nu_{i-1} + t_i - t_{i-1})[R_i \mapsto 0]$ (where we assume $t_0 = 0$).

Theorem 7.2 (From [2]) *Given a finite timed automaton $\mathcal{B}$, one can decide if $\mathcal{L}_{tw}(\mathcal{B}) \neq \emptyset$.*

For the rest of this section, we fix an ECMPA $\mathcal{A} = (\{\mathcal{A}_p\}_{p \in \mathit{Proc}}, Act, \Delta, F)$, with $\mathcal{A}_p = (S_p, \iota_p, \rightarrow_p)$, and an integer $B > 0$. We also fix an (infinite) set of indices $\text{Ind} = Act \times \mathbb{N}$ which will be used to index the "copies" of the gadgets that we will use. We will define an

infinite timed automaton $\mathcal{B} = (Q_\mathcal{B}, \mathcal{Z}_\mathcal{B}, \delta_\mathcal{B}, \iota_\mathcal{B}, F_\mathcal{B})$ such that $\mathcal{L}_{tw}(\mathcal{B}) = \{\sigma \in (Act \times \mathbb{R}_{\geq 0})^* \mid$ there is $T \in \mathcal{L}_{time}(\mathcal{A})$ such that σ is a B-bounded timed linearization of $T\}$. A state $\mathrm{st} \in Q_\mathcal{B}$ is a 6-tuple $(\overline{s}, \chi, \eta, \xi^\triangleleft, \xi^\triangleright, \gamma)$ where:

- $\overline{s} = (s_p)_{p \in Proc} \in \prod_{p \in Proc} S_p$ is a tuple of local states from the ECMPA.
- $\chi : Ch \to (\mathcal{M} \times \Delta)^{\leq B}$ describes the contents of the channels.
- $\eta : Act \to \{0, \ldots, B-1\}$ gives the B-stamping number that should be assigned to the next occurrence of an action.
- $\xi^\triangleleft : \mathrm{Ind} \dashrightarrow Q^\triangleleft$ associates with certain indices, states of $\mathcal{C}^\triangleleft$. Thus, the indices in $\mathrm{dom}(\xi^\triangleleft)$ of a state specify which copies of the gadgets are "active" in that state.
- $\xi^\triangleright : \mathrm{Ind} \dashrightarrow Q^\triangleright \times \mathrm{Int}(\mathcal{A})$ associates with some indices, states of $\mathcal{C}^\triangleright$ along with constraints that need to be maintained
- $\gamma : Ch \times \{0, \ldots, B-1\} \dashrightarrow \mathrm{Int}(\mathcal{A})$ describes the guards attached to messages.

The initial state is $\iota_\mathcal{B} = ((\iota_p)_{p \in Proc}, \chi_0, \eta_0, \xi_0^\triangleleft, \xi_0^\triangleright, \gamma_0)$ where χ_0 and η_0 map any argument to the empty word and 0 respectively, and the partial maps $\xi_0^\triangleleft$, $\xi_0^\triangleright$ and γ_0 are nowhere defined. We will use clocks from the set $\mathcal{Z}_\mathcal{B} = \{z^\triangleleft_{a,i}, z^\triangleright_{a,i} \mid (a,i) \in \mathrm{Ind}\} \cup \{z^\gamma_{p,q,i} \mid (p,q) \in Ch, i \in \{0, \ldots, B-1\}\}$. Also, we fix some notations regarding how we write constraints using these clocks. For a clock $z \in \mathcal{Z}_\mathcal{B}$ and an interval $I \in \mathcal{I}$ with endpoints $l, r \in \mathbb{N}$, we write "$x \in I$" to denote a constraint from $\mathrm{Form}(\mathcal{Z}_\mathcal{B})$. More specifically, $x \in (l, r)$ denotes $x > l \wedge x < r$, whereas $x \in [l, r)$ refers to $x \geq l \wedge x < r$, and so on.

The transition relation $\delta_\mathcal{B} \subseteq Q_\mathcal{B} \times \mathrm{Form}(\mathcal{Z}_\mathcal{B}) \times Act \times 2^{\mathcal{Z}_\mathcal{B}} \times Q_\mathcal{B}$ is defined by,

$$\left((\overline{s}, \chi, \eta, \xi^\triangleleft, \xi^\triangleright, \gamma), \varphi, a, R, (\overline{s}', \chi', \eta', \xi'^\triangleleft, \xi'^\triangleright, \gamma')\right) \in \delta_\mathcal{B}$$

if there are $p, q \in Proc, m \in \mathcal{M}, \theta \in \{!, ?\}$ such that $a = p\theta q(m)$ and there exists a p-local transition of the ECMPA, $(s_p, a, g, d, s'_p) \in \to_p$ for some $g \in [\mathrm{TC} \dashrightarrow \mathcal{I}]$ and $d \in \Delta$, such that the following conditions (i)–(ix) hold:

(i) $s'_r = s_r$ for all $r \in Proc \setminus \{p\}$.

(ii) If $\theta = !$ then $\chi'(p,q) = (m,d) \cdot \chi(p,q)$, $\chi'(r,s) = \chi(r,s)$ for all $(r,s) \in Ch \setminus \{(p,q)\}$.

(iii) If $\theta = ?$ then $\chi(q,p) = \chi'(q,p) \cdot (m,d)$, $\chi'(r,s) = \chi(r,s)$ for all $(r,s) \in Ch \setminus \{(q,p)\}$.

(iv) For all $b \in Act$,

$$\eta'(b) = \begin{cases} (\eta(b)+1) \bmod B & \text{if } b \in p\theta q \\ \eta(b) & \text{otherwise} \end{cases}$$

(v) The states of the *previous* automata are updated and we have initialized a new copy of $\mathcal{C}^\triangleleft$ starting at the current position so that we can determine which later positions are related with the current one by the *previous* relation.

$$\xi'^\triangleleft(b,i) = \begin{cases} \delta^\triangleleft(s_0^\triangleleft, (a, \eta(a))) & \text{if } b = a \text{ and } i = \min(\mathbb{N} \setminus \mathrm{dom}(\xi^\triangleleft(a))) \\ \delta^\triangleleft(\xi^\triangleleft(b,i), (a, \eta(a))) & \text{if } (b,i) \in \mathrm{dom}(\xi^\triangleleft) \\ \text{undefined} & \text{otherwise} \end{cases}$$

(vi) The states of the *next* automata are updated along with the corresponding guards. A new copy of $\mathcal{C}^\triangleright$ is initialized for each $b \in Act$, if there is a Next_b constraint on the local transition. The guard itself is stored in the second component, so that it can be

verified when we reach the *next* occurrence of the action. Once verified, we release the guard and the corresponding copy of $\mathcal{C}^{\triangleright}$.

$$\xi'^{\triangleright}(b,i) = \begin{cases} (\delta^{\triangleright}(s_0^{\triangleright},(a,\eta(a))), g(\mathrm{Next}_b)) & \text{if } \mathrm{Next}_b \in \mathrm{dom}(g) \text{ and} \\ & i = \min(\mathbb{N} \setminus \mathrm{dom}(\xi^{\triangleright}(b))) \\ (\delta^{\triangleright}(s^{\triangleright},(a,\eta(a))), I) & \text{if } \xi^{\triangleright}(b,i) = (s^{\triangleright}, I) \text{ and} \\ & \neg(a = b \wedge \delta^{\triangleright}(s^{\triangleright},(a,\eta(a))) \in F^{\triangleright}) \\ \text{undefined} & \text{otherwise} \end{cases}$$

(vii) The guards attached to message constraints are maintained as expected. A send event introduces a constraint, which is retained until its matching receive releases it.

$$\gamma'\big((r,s),i\big) = \begin{cases} g(\mathrm{Msg}) & \text{if } a \in r!s, i = \eta(a),\ \mathrm{Msg} \in \mathrm{dom}(g) \\ \text{undefined} & \text{if } a \in s?r \text{ and } i = \eta(a) \\ \gamma((r,s),i) & \text{otherwise} \end{cases}$$

(viii) A clock is reset for every new copy of $\mathcal{C}^{\triangleleft}$, $\mathcal{C}^{\triangleright}$ and message constraint introduced at this transition.

$$R = \left\{ z^{\triangleleft}_{a,i} \;\middle|\; i = \min\big(\mathbb{N} \setminus \mathrm{dom}\big(\xi^{\triangleleft}(a)\big)\big) \right\} \cup \left\{ z^{\gamma}_{p,q,i} \;\middle|\; a \in p!q, i = \eta(a), \mathrm{Msg} \in \mathrm{dom}(g) \right\}$$
$$\cup \left\{ z^{\triangleright}_{b,i} \mathrm{Next}_b \in \mathrm{dom}(g) \text{ and } i = \min\big(\mathbb{N} \setminus \mathrm{dom}\big(\xi^{\triangleright}(b)\big)\big) \right\}$$

(ix) The guard must ensure that all constraints that get *matched* at the current event are satisfied. Thus $\varphi = \varphi^{\triangleleft} \wedge \varphi^{\triangleright} \wedge \varphi^{m}$ where

$$\varphi^{\triangleleft} = \bigwedge_{\substack{\{(b,i) \mid \mathrm{Prev}_b \in \mathrm{dom}(g) \\ \text{and } \xi'^{\triangleleft}(b,i) \in F^{\triangleleft}\}}} z^{\triangleleft}_{b,i} \in g(\mathrm{Prev}_b) \wedge \bigwedge_{\substack{\{b \mid \mathrm{Prev}_b \in \mathrm{dom}(g) \\ \text{and } \{i \mid \xi'^{\triangleleft}(b,i) \in F^{\triangleleft}\} = \emptyset\}}} \text{false}$$

ensures that all previous constraints that are matched are satisfied. Thus if the local transition contains a Prev_b constraint, then we have to check $z^{\triangleleft}_{b,i} \in g(\mathrm{Prev}_b)$ for the (unique) i such that $\xi'^{\triangleleft}(b,i) \in F^{\triangleleft}$. If there is no such i then there is no b-action in the past of the current event and the Prev_b constraint of the local transition cannot be satisfied. In this case, we set $\varphi^{\triangleleft}$ to false. For next constraints, we have

$$\varphi^{\triangleright} = \bigwedge_{\substack{\{i \in \mathrm{dom}(\xi^{\triangleright}(a)) \mid \xi^{\triangleright}(a,i) = (s^{\triangleright}, I), \\ \delta^{\triangleright}(s^{\triangleright},(a,\eta(a))) \in F^{\triangleright}\}}} z^{\triangleright}_{a,i} \in I$$

If the current action is the next occurrence of a from some positions where a *next* guard was registered, for each there is a copy (a,i) of $\mathcal{C}^{\triangleright}$ which reaches a final state. Thus, we verify the corresponding clock with the constraint recovered from $\xi^{\triangleright}$. For message constraints, we have

$$\varphi^{m} = \bigwedge_{\substack{\{((q,p),i) \in \mathrm{dom}(\gamma) \mid \\ a \in p?q,\ \eta(a) = i\}}} z^{\gamma}_{q,p,i} \in \gamma\big((q,p),i\big)$$

The set of final states is $F_{\mathcal{B}} = \{(\overline{s}, \chi, \eta, \xi^{\triangleleft}, \xi^{\triangleright}, \gamma) \in Q_{\mathcal{B}} \mid \overline{s} \in F, \chi = \chi_0, \mathrm{dom}(\xi^{\triangleright}) = \mathrm{dom}(\gamma) = \emptyset\}$. This ensures that each registered guard has been checked. Indeed, a *next* or *message* constraint is released only when it is checked with the guard φ.

One critical observation here is that, once we have specified the local transition of $\mathcal{A}$, this global transition of $\mathcal{B}$ gets determined uniquely. Thus, this step is always deterministic. Note that the above automaton $\mathcal{B}$ has no ϵ-transitions either. Now, we prove that $\mathcal{L}_{tw}(\mathcal{B})$ contains precisely the B-bounded timed linearizations of $\mathcal{L}_{time}(\mathcal{A})$.

Theorem 7.3 $\mathcal{L}_{tw}(\mathcal{B}) = \{\sigma \in (Act \times \mathbb{R}_{\geq 0})^* \mid$ *there is* $T \in \mathcal{L}_{time}(\mathcal{A})$ *such that* σ *is a* B*-bounded timed linearization of* $T\}$.

Proof Let $T = (M, t)$ with $M = (E, \leq, \lambda)$ over Act. Then, a B-bounded timed linearization $\sigma = (a_1, t_1) \ldots (a_n, t_n)$ of T generates the corresponding B-bounded linearization of M, namely $a_1 \ldots a_n$ over the same set of positions $\{1, \ldots, n\}$. Thus, as stated in the previous section, we can interpret the events from E to be positions from $\{1, \ldots, n\}$.

Then, recall that $T \in \mathcal{L}_{time}(\mathcal{A})$ iff there is an accepting run r of $\mathcal{A}$ on T, where $r : E \to \bigcup_{p \in Proc} S_p$ is given by Definition 4.1. Also, let us recall that $\sigma \in \mathcal{L}_{tw}(\mathcal{B})$ iff there is an accepting run r' of $\mathcal{B}$ on σ, i.e., there is a sequence,

$$r' = (\mathrm{st}_0, \nu_0) \xrightarrow{a_1, t_1} (\mathrm{st}_1, \nu_1) \xrightarrow{a_2, t_2} \cdots \xrightarrow{a_n, t_n} (\mathrm{st}_n, \nu_n) \tag{15}$$

where for all $i \in \{0, \ldots, n\}$, $\mathrm{st}_i = (\overline{s}_i, \chi_i, \eta_i, \xi_i^{\triangleleft}, \xi_i^{\triangleright}, \gamma_i) \in Q_{\mathcal{B}}$ (with $\overline{s}_i = (s_p^i)_{p \in Proc} \in \prod_{p \in Proc} S_p$), $\mathrm{st}_0 = \iota_{\mathcal{B}}$, $\mathrm{st}_n \in F_{\mathcal{B}}$ and $\nu_i : \mathcal{Z}_{\mathcal{B}} \to \mathbb{R}_{\geq 0}$ (with $\nu_0(x) = 0$ for all $x \in \mathcal{Z}_{\mathcal{B}}$), and for each $i \in \{1, \ldots, n\}$, there exist $\varphi_i \in Form(\mathcal{Z}_{\mathcal{B}})$ and $R_i \in 2^{\mathcal{Z}_{\mathcal{B}}}$ such that,

$$(\mathrm{st}_{i-1}, \varphi_i, a_i, R_i, \mathrm{st}_i) \in \delta_{\mathcal{B}} \tag{16}$$

$$(\nu_{i-1} + t_i - t_{i-1}) \models \varphi_i \tag{17}$$

$$\nu_i = (\nu_{i-1} + t_i - t_{i-1})[R_i \mapsto 0] \tag{18}$$

Now we construct an accepting run r' of $\mathcal{B}$ on σ from an accepting run r of $\mathcal{A}$ on T and vice versa.

($\Rightarrow$) Let r be an accepting run of $\mathcal{A}$ on T. Then, we construct run r' inductively from $i = 0$ to $i = n$. Further, we maintain two further state invariants at each step $i \in \{0, \ldots, n\}$:

$$\text{for all } p \in Proc\text{: } s_p^i = \begin{cases} r(j) & \text{if } \exists j \leq i,\ j \in E_p,\ \nexists k \in E_p,\ j < k \leq i \\ \iota_p & \text{otherwise} \end{cases} \tag{19}$$

$$\text{for all } b \in Act \text{ with } b = p\theta q(m)\text{: } \eta_i(b) = \left|\{i' \leq i \mid \lambda(i') \in p\theta q\}\right| \bmod B \tag{20}$$

Then, indeed at $i = 0$, we have $\mathrm{st}_0 = \iota_{\mathcal{B}} = (\overline{s}_0, \chi_0, \eta_0, \xi_0^{\triangleleft}, \xi_0^{\triangleright}, \gamma_0)$. This satisfies our state invariants (19)–(20) since $s_p^0 = \iota_p$ for all $p \in Proc$ and $\eta_0(a) = 0$ for all $a \in Act$. Now for some $i \in \{1, \ldots, n\}$ assuming we have constructed the run r' until $(\mathrm{st}_{i-1}, \nu_{i-1})$, let $a_i = p\theta q(m)$ for some $p, q \in Proc$, $m \in \mathcal{M}$, and $\theta \in \{!, ?\}$. We then extend the run r' to stage i by exhibiting st_i, ν_i, φ_i, and R_i such that conditions (16)–(20) hold.

From the definition of r, we have a local transition on the ECMPA on event i. More precisely, $(r^-(i), a_i, g_i, d_i, r(i)) \in \to_p$ for some guard g_i and $d_i \in \Delta$. Recall that $r^-(i) = r(i')$ for $i' \lessdot_{pp} i$, and $r^-(i) = \iota_p$ if such an event i' does not exist. Thus, by condition (19) at stage $i - 1$, we have $s_p^{i-1} = r^-(i)$. And by choosing $s_p^i = r(i)$ we obtain $(s_p^{i-1}, a_i, g_i, d_i, s_p^i) \in \to_p$. Again, by choosing for all $p' \neq p$, $s_{p'}^i = s_{p'}^{i-1}$, condition (19) holds at stage i. (This follows since for p, the largest $j \leq i$ such that $j \in E_p$ is i itself. And for $p' \neq p$, the largest $j' \leq i$ such that $j' \in E_{p'}$ is the largest such event $j' \leq i - 1$.)

Now, as we commented in our construction, the local transition fully specifies the global transition and thus we get a transition of $\mathcal{B}$,

$$\big((\overline{s}_{i-1}, \chi_{i-1}, \eta_{i-1}, \xi^{\triangleleft}_{i-1}, \xi^{\triangleright}_{i-1}, \gamma_{i-1}), \varphi_i, a_i, R_i, (\overline{s}_i, \chi_i, \eta_i, \xi^{\triangleleft}_i, \xi^{\triangleright}_i, \gamma_i)\big) \in \delta_{\mathcal{B}},$$

where $\varphi_i, R_i, \chi_i, \eta_i, \xi^{\triangleleft}_i, \xi^{\triangleright}_i, \gamma_i$ are defined from their values at stage $i-1$ and the local transition. Thus condition (16) holds at i, with $\mathrm{st}_i = (\overline{s}_i, \chi_i, \eta_i, \xi^{\triangleleft}_i, \xi^{\triangleright}_i, \gamma_i)$. Again condition (20) continues to hold at i if it holds at $i-1$. (If $b \in p\theta q$, then $\eta_i(b) = (\eta_{i-1}(b)+1) \bmod B = |\{i' \le i-1 \mid \lambda(i') \in p\theta q\}| + 1 \bmod B = |\{i' \le i \mid \lambda(i') \in p\theta q\}| \bmod B$. And for $b \in Act \setminus p\theta q$, it follows since $\eta_i(b) = \eta_{i-1}(b)$.) Now, we just define $\nu_i = (\nu_{i-1} + t_i - t_{i-1})[R_i \mapsto 0]$, so that condition (18) holds at i. Thus, we have extended run r' of $\mathcal{B}$ on σ to i, if we prove condition (17).

Claim $(\nu_{i-1} + t_i - t_{i-1}) \models \varphi_i$.

Proof The proof of this claim is by induction on the structure of φ_i. But first we observe that the mapping $\rho : E \to \{0, \ldots, B-1\}$ which maps $\rho(i) = \eta_i(a_i)$ is a B-well-stamping for M. This follows from the fact that the state invariant, condition (20), holds until stage i. We have the following cases to consider.

(1) *Previous* constraint of the form $z^{\triangleleft}_{a,k} \in g_i(\mathrm{Prev}_a)$ or false: If for some $a \in Act$, $\mathrm{Prev}_a \in \mathrm{dom}(g_i)$, then by the definition of a run r of $\mathcal{A}$ on T (i.e., condition (5) or (6)), there exists an event j such that $(i, j) \in \mathrm{Prev}^T_a$ (thus, $a_j = a$). By Lemma 7.2, $(a_j, \eta_j(a_j)) \ldots (a_i, \eta_i(a_i)) \in \mathcal{L}(\mathcal{C}^{\triangleleft})$. And so, in the run of $\mathcal{B}$ at stage i, $\xi^{\triangleleft}_i(a,k) \in F^{\triangleleft}$ where $k = \min(\mathbb{N} \setminus \mathrm{dom}(\xi^{\triangleleft}_{j-1}(a)))$. Hence, for any a such that $\mathrm{Prev}_a \in \mathrm{dom}(g_i)$, the set $\{\ell \mid \xi'^{\triangleleft}(a,\ell) \in F^{\triangleleft}\} \neq \emptyset$ and so the *false* constraint cannot occur as a guard in this simulation, i.e., as part of φ_i. Now, at stage j, we have $z^{\triangleleft}_{a,k} \in R_j$ (there cannot be another Prev_a guard for some other k' since there is a unique preceding occurrence of each letter). Also $z^{\triangleleft}_{a,k} \notin R'_j$ for all $j' \in \{j+1, \ldots, i\}$, since $(a, j') \in \mathrm{dom}(\xi^{\triangleleft}_{j'-1})$. Therefore in the valuation $(\nu_{i-1} + t_i - t_{i-1})(z^{\triangleleft}_{a,k}) = \nu_{i-1}(z^{\triangleleft}_{a,k}) + t_i - t_{i-1} = \nu_{i-2}(z^{\triangleleft}_{a,k}) + t_{i-1} - t_{i-2} + t_i - t_{i-1} = \cdots = \nu_j(z^{\triangleleft}_{a,k}) + t_i - t_j$ and $\nu_j(z^{\triangleleft}_{a,k}) = 0$. Thus $\nu_j(z^{\triangleleft}_{a,k}) + t_i - t_j = t_i - t_j$ and so we are done if we show that $t_i - t_j \in g_i(\mathrm{Prev}_a)$. But this follows from condition (5) or (6), where we have $|t(j) - t(i)| \in g_i(\mathrm{Prev}_a)$ where $t(j) = t_j$ and $t(i) = t_i$. Thus we conclude that the valuation satisfies any *previous* clock constraint of the form $z^{\triangleleft}_{a,k} \in g_i(\mathrm{Prev}_a)$.

(2) *Next* constraint of the form $z^{\triangleright}_{a_i,k} \in I$: This implies that there is $k \in \mathrm{dom}(\xi^{\triangleright}_{i-1}(a_i))$ such that, $\xi^{\triangleright}_{i-1}(a_i, k) = (s^{\triangleright}, I)$ and $\delta^{\triangleright}(s^{\triangleright}, (a_i, \eta_i(a_i))) \in F^{\triangleright}$. Then, by the definition of the *next* update function $\xi^{\triangleright}$, we can conclude that there exists $j < i$, $k = \min(\mathbb{N} \setminus \mathrm{dom}(\xi^{\triangleright}_{j-1}(a_i)))$ such that $\xi^{\triangleright}_j(a_i, k) = (\delta^{\triangleright}(s^{\triangleright}_0, (a_j, \eta(a_j)), I))$ such that $I = g_j(\mathrm{Next}_{a_i})$. Thus, $(a_j, \eta_j(a_j)) \ldots (a_i, \eta_i(a_i)) \in L(\mathcal{C}^{\triangleright})$ and by Lemma 7.2, we can conclude that $(j, i) \in \mathrm{Next}^T_{a_i}$. Hence from the definition of run r on TMSC T, we get $|t(j) - t(i)| \in g_j(\mathrm{Next}_{a_i}) = I$. Also, $z^{\triangleright}_{a_i,k} \in R_j$ and it is not reset until i. If not, let j' with $j < j' < i$ be the first instance where $z^{\triangleright}_{a_i,k} \in R_{j'}$. This $(a_i, k) \notin \mathrm{dom}(\xi^{\triangleright}_{j'})$ and $(a_i, k) \in \mathrm{dom}(\xi^{\triangleright}_{j'-1})$ implies that $a_{j'} = a_i$ and $(a_j, \eta(j)) \ldots (a_{j'}, \eta(j')) \in \mathcal{L}(\mathcal{C}^{\triangleright})$ which contradicts $(j, i) \in \mathrm{Next}_{a_i}$. Thus, for all $j < j' < i$ we get $z^{\triangleright}_{a_i,k} \notin R_{j'}$, $(\nu_{i-1} + t_i - t_{i-1})(z^{\triangleright}_{a_i,k}) = t_i - t_j \in I$ and so we are done.

(3) *Message* constraint of the form $z^{\gamma}_{q,p,k} \in \gamma_{i-1}(q, p, k)$: Here, $(q, p, k) \in \mathrm{dom}(\gamma_{i-1})$ such that $a_i \in p?q$, $\eta_{i-1}(a_i) = k$. Then we look at the largest $j \le i$ such that $a_j = q!p$ and $\eta_{j-1}(a_j) = k$. Such a j exists since if not it would contradict the fact that σ is a linearization of a valid TMSC T. Further, for all j' with $j < j' < i$, it is not the case that $a_{j'} \in p?q$ and $\eta_{j'-1}(a_{j'}) = k$. Then it follows that for all j', $j \le j' < i$, $\gamma_{j'}(q, p, k) = g_j(\mathrm{Msg})$. Also $(j, i) \in \mathrm{Msg}^T$ and so by definition of r on TMSC, $|t_j - t_i| \in g_j(\mathrm{Msg}) = \gamma_{i-1}(q, p, k)$. And

again, $z^{\gamma}_{q,p,k} \in R_j$ and for all $j < j' \leq i$, $z^{\gamma}_{q,p,k} \notin R_{j'}$, so $(\nu_{i-1} + t_i - t_{i-1})(z^{\gamma}_{q,p,k}) = t_i - t_j \in \gamma_{i-1}(q, p, k)$ and thus we are done. □

Now it is easy to see that the state $\widetilde{s}_n = (\overline{s}_n, \chi_n, \eta_n, \xi^{\triangleleft}_n, \xi^{\triangleright}_n, \gamma_n)$ reached at the end of the above run, is a final state. This follows from the fact that r is a successful run of $\mathcal{A}$ on T, since then we have $\overline{s}_n \in F$ and $\chi_n = \chi_0$ (since at the end of r the channel contents must be empty), and the partial maps $\xi^{\triangleright}$ and γ are nowhere defined (since if that were not the case then this means that a constraint was not checked with its guard).

($\Leftarrow$) For the converse, from r' as defined in (15) above, we want to construct a run $r : E \to \bigcup_{p \in Proc} S_p$ of $\mathcal{A}$ on T. We define, for each event $i \in \{1, \ldots, n\}$, $r(i) = s^i_{p_i}$. Now, at each $i \in \{0, \ldots, n\}$, by (16), we have $(\mathrm{st}_{i-1}, a_i, \varphi_i, R_i, \mathrm{st}_i) \in \delta_{\mathcal{B}}$. By definition of $\mathcal{B}$, for each $i \in \{0, \ldots, n\}$, we have $(s^{i-1}_{p_i}, a_i, g_i, d_i, s^i_{p_i}) \in \to_{p_i}$ for some g_i and d_i. We now show that this map is a run of $\mathcal{A}$ on T by verifying conditions (2), (5)–(6) in Definition 4.1. First, $r^-(i) = s^{i-1}_{p_i}$, as can be proved by induction on i: For $i = 1$, $\mathrm{st}_0 = \iota_{\mathcal{B}}$ implies that $s^0_{p_1} = \iota_{p_1}$ and $r^-(a_1) = \iota_{p_1}$ since a_1 is the minimal event in that process. For $i > 1$, $r^-(i) = r(j)$ if there is $j \lessdot_{p_i p_i} i$ and $r^-(i) = \iota$ otherwise. Thus we have $s^{i-1}_{p_i} = s^j_{p_j}$. Now, for events i, j, if $i <^M_{pq} j$ then $(r^-(i), a_i, g_i, d_i, r(i)) \in \to_{p_i}$ and $(r^-(j), a_j, g_j, d_j, r(j)) \in \to_{p_j}$ implies that $d_i = d_j$ which means that condition (2) holds. This follows from the definition of χ in $\delta_{\mathcal{B}}$ ensuring that sent and received messages are synchronized. To prove the other conditions, for any event i, and $\alpha \in \mathrm{TC}$, we need to show that, if $\alpha \in \mathrm{dom}(g_i)$, then there exists j with $(i, j) \in \mathrm{dom}(\alpha^T)$ such that $|t(i) - t(j)| \in g_i(\alpha)$. We have three cases depending on α.

- $\alpha = \mathrm{Prev}_a$ for some $a \in Act$: If $\mathrm{Prev}_a \in \mathrm{dom}(g_i)$, then firstly there exists some (unique) k such that $\xi^{\triangleleft}_i(a, k) \in F^{\triangleleft}$. If not, then the *false* constraint will occur in φ_i which contradicts acceptance of σ by $\mathcal{B}$. Thus we have that for this (a, k), the constraint $z^{\triangleleft}_{a,k} \in g_i(\mathrm{Prev}_a)$ occurs in φ_i and $(\nu_{i-1} + t_i - t_{i-1})(z^{\triangleleft}_{a,k}) \models \varphi_i$ implies that $(\nu_{i-1} + t_i - t_{i-1})(z^{\triangleleft}_{a,k}) \in g_i(\mathrm{Prev}_a)$. Now by definition of *previous* state updates in the run of $\mathcal{B}$, we can use Lemma 7.2 to conclude that there exists $j \leq i$ for which $(i, j) \in \mathrm{Prev}^M_a$. Further, at j, we can see that $z^{\triangleleft}_{a,k} \in R_j$, and for all j' with $j < j' < i$, we have $z^{\triangleleft}_{a,k} \notin R_{j'}$. Thus, $(\nu_{i-1} + t_i - t_{i-1})(z^{\triangleleft}_{a,k}) = t_i - t_j$. Hence $(i, j) \in \mathrm{Prev}^T_a$ and $|t(i) - t(j)| \in g_i(\mathrm{Prev}_a)$.
- $\alpha = \mathrm{Next}_a$ for some $a \in Act$: If $\mathrm{Next}_a \in \mathrm{dom}(g_i)$, then by definition of $\xi^{\triangleright}$, for $k = \min(\mathbb{N} \setminus \mathrm{dom}(\xi^{\triangleright}_{i-1}(a)))$, we have $\xi^{\triangleright}_i(a, k) = (\delta^{\triangleright}(s^{\triangleright}_0, (a_i, \eta(a_i)), g_i(\mathrm{Next}_a)))$. Also we reset clock $z^{\triangleright}_{a,k}$. But as r' is an accepting run, $\xi^{\triangleright}_n(a, k)$ is undefined and so there exists j such that $a = a_j$ and $\xi^{\triangleright}_j(a, k) = (s, I)$ with $s \in F^{\triangleright}$. Let j be the smallest such j such that, for all j' with $i < j' < j$, we have $(a, k) \in \mathrm{dom}(\xi^{\triangleright}_{j'})$. Thus $I = g_i(\mathrm{Next}_a)$. Also $(a_i, \eta_i(a_i)) \ldots (a_j, \eta_j(a_j)) \in \mathcal{L}(\mathcal{C}^{\triangleright})$ and by Lemma 7.2, $(i, j) \in \mathrm{Next}^T_a$. Now, we have $(\nu_{j-1} + t_j - t_{j-1})(z^{\triangleright}_{a,k}) = \nu_i(z^{\triangleright}_{a,k}) + t_j - t_i = t_j - t_i$. Again as the run was successful, $z^{\triangleright}_{a,k} \in I$ occurs in φ_j and $(\nu_{j-1} + t_j - t_{j-1})(z^{\triangleright}_{a,k}) \in I = g_i(\mathrm{Next}_a)$. Thus $t_j - t_i \in g_i(\mathrm{Next}_\sigma)$ and we are done.
- $\alpha = \mathrm{Msg}$: This is similar to the above case (and easier, as the *next* automaton is not needed). □

This completes both directions of the proof.

7.3 A finite version of $\mathcal{B}$

To get a finite version of $\mathcal{B}$, we will bound the set of indices Ind to a finite set Ind_{fin}, thus, constructing a finite timed automaton $\mathcal{B}'$ that is equivalent to $\mathcal{B}$. The state space of $\mathcal{B}'$ is the same as $\mathcal{B}$ except that it uses indices from Ind_{fin} to define the $\xi^{\triangleleft}$ and $\xi^{\triangleright}$ components.

We construct $\mathcal{B}'$ in two steps. First, we describe how transitions of $\mathcal{B}$ are modified to handle *previous* gadgets. Next, we describe how to modify the transitions of $\mathcal{B}$ to handle *next* gadgets. Thus, $\mathcal{B}'$, obtained after both these modifications will turn out to be a finite timed automaton. To bound the set of indices, our basic idea is to reuse copies of the previous and next gadgets when it is safe to do so. First, we handle the *previous* case by examining when it is "safe" to release a copy of $\mathcal{C}^{\triangleleft}$. The following proposition gives us the criterion required.

Proposition 7.2 *Let* $(\overline{s}, \chi, \eta, \xi^{\triangleleft}, \xi^{\triangleright}, \gamma)$ *be a reachable state of* $\mathcal{B}$. *If there exist two indices* $(a,i), (a,j) \in \mathrm{dom}(\xi^{\triangleleft})$, $i \neq j$, *such that* $\xi^{\triangleleft}(a,i) = \xi^{\triangleleft}(a,j) = s^{\triangleleft} \in Q^{\triangleleft}$, *then no final state of* $s_f^{\triangleleft} \in F^{\triangleleft}$ *is reachable from* $s^{\triangleleft}$.

Proof The copies of $\mathcal{C}^{\triangleleft}$ indexed by (a,i) and (a,j) have been started at distinct positions labeled a to keep track of two different pasts. Now suppose there exists $s_f^{\triangleleft} \in F^{\triangleleft}$ such that $s_f^{\triangleleft}$ is reachable from $s^{\triangleleft}$. Then at $s_f^{\triangleleft}$, this position is related by Prev_a with both starting positions, i.e., when the clocks $z_{a,i}^{\triangleleft}$ and $z_{a,j}^{\triangleleft}$ were last reset. But this is not possible, because there is at most one previous position labeled a for any position. Thus no final state is reachable from $s^{\triangleleft}$. □

This implies that we can safely remove the corresponding indices (a,i) and (a,j) from $\mathrm{dom}(\xi^{\triangleleft})$. Thus, we say that a state $\mathrm{st} = (\overline{s}, \chi, \eta, \xi^{\triangleleft}, \xi^{\triangleright}, \gamma) \in Q_{\mathcal{B}}$ is $\triangleleft$-*safe* if there are no two indices $(a,i), (a,j) \in \mathrm{dom}(\xi^{\triangleleft})$, $i \neq j$, such that $\xi^{\triangleleft}(a,i) = \xi^{\triangleleft}(a,j)$. Otherwise, we say that st is $\triangleleft$-*unsafe* and that $(a,i), (a,j)$ are $\triangleleft$-*unsafe* indices at the state st. A transition from $\mathcal{B}$ is retained in $\mathcal{B}'$ if it is between $\triangleleft$-safe states. Further, every transition $(\mathrm{st}, \varphi, a, R, \mathrm{st}')$ in $\mathcal{B}$ from a $\triangleleft$-safe state st to a $\triangleleft$-unsafe state $\mathrm{st}' = (\overline{s}', \chi', \eta', \xi'^{\triangleleft}, \xi'^{\triangleright}, \gamma')$ is replaced by transition $(\mathrm{st}, \varphi, a, R, \widetilde{\mathrm{st}'})$ between $\triangleleft$-safe states. Hereby, $\widetilde{\mathrm{st}'} = (\overline{s}', \chi', \eta', \widetilde{\xi'^{\triangleleft}}, \xi'^{\triangleright}, \gamma')$ with $\widetilde{\xi'^{\triangleleft}}(b,i) = \xi'^{\triangleleft}(b,i)$ if $(b,i) \in \mathrm{dom}(\xi'^{\triangleleft})$ and there is no $j \neq i$ such that $(b,j) \in \mathrm{dom}(\xi'^{\triangleleft})$ and $\xi'^{\triangleleft}(b,i) = \xi'^{\triangleleft}(b,j)$. Otherwise, $\widetilde{\xi'^{\triangleleft}}(b,i)$ is undefined. By Proposition 7.2, $\mathcal{B}'$ still accepts the same set of timed words as $\mathcal{B}$. This is enough to ensure finiteness in the *previous* case as shown below.

Lemma 7.3 *For any reachable state* $(\overline{s}, \chi, \eta, \xi^{\triangleleft}, \xi^{\triangleright}, \gamma) \in Q_{\mathcal{B}'}$, *we have* $\mathrm{dom}(\xi^{\triangleleft}) \subseteq Act \times \{0, \ldots, |Q^{\triangleleft}|\}$.

Proof Suppose not, then, for some $a \in Act$, $|\mathrm{dom}(\xi^{\triangleleft}, a)| \geq (|Q^{\triangleleft}| + 1)$. But this implies that there must exist at least two indices $(a,i), (a,j) \in \mathrm{dom}(\xi^{\triangleleft})$, $i \neq j$, such that $\xi^{\triangleleft}(a,i) = \xi^{\triangleleft}(a,j)$. By the above definition, they would have been undefined and, hence, cannot be in the domain of $\xi^{\triangleleft}$. Thus, we have a contradiction. □

The remaining source of infinity comes from *next* constraints. The situation is not as easy as for *previous* constraints, since *next* constraints registered at several positions could be matched at the same time. Thus, the number of registered Next_b-constraints may be unbounded. In particular, in some state of $\mathcal{B}$, suppose $(b,i), (b,j) \in \mathrm{dom}(\xi^{\triangleright})$ for some $i \neq j$ such that $\xi^{\triangleright}(b,i) = (s^{\triangleright}, I)$ and $\xi^{\triangleright}(b,j) = (s^{\triangleright}, I')$. Then, the constraints associated with i and j will be matched simultaneously. When matched, the guard on the transition of $\mathcal{B}$ will include both $z_{b,i}^{\triangleright} \in I$ and $z_{b,j}^{\triangleright} \in I'$. Our idea is to keep only the *stronger constraint* and release the other. To determine the stronger constraint we deal separately with upper and lower constraints.

Refining the constraints A clock constraint over $\mathcal{Z}_{\mathcal{B}}$ is called an *upper*-guard if it is of the form $x \sim c$ where $\sim \in \{<, \leq\}$ for some $x \in \mathcal{Z}_{\mathcal{B}}$, $c \in \mathbb{Q}_{\geq 0}$. Similarly $x \sim c$ is a *lower*-guard if $\sim \in \{>, \geq\}$. Note that each "$x \in I$" defines uniquely a lower and an upper guard, depending upon the endpoints of the interval I.

Definition 7.2 Let $x \sim c$ and $x' \sim' c'$ be two upper-guards with $\sim, \sim' \in \{<, \leq\}$ or two lower guards with $\sim, \sim' \in \{>, \geq\}$. We say $x \sim c$ is *stronger than* $x' \sim' c'$ if, when evaluated at the same instant, $x \sim c$ holds implies that $x' \sim' c'$ holds as well.

The stronger constraint can be determined with a diagonal guard: For upper guards, $x \sim c$ is stronger than $x' \sim' c'$ if either $x' - x < c' - c$ or $x' - x \leq c' - c$ and ($\sim = <$ or $\sim' = \leq$). The relation *stronger than* is transitive among upper-guards. It is also total: either $x \sim c$ is stronger than $x' \sim' c'$, or the converse holds, or both in which case we say that the two constraints are equivalent. The constraints are *equivalent* iff $x' - x = c' - c$ and $\sim = \sim'$. The above properties are true for lower guards as well, where we have $x \sim c$ stronger than $x' \sim' c'$ if either $x' - x > c' - c$ or $x' - x \geq c' - c$ and ($\sim = >$ or $\sim' = \geq$).

Restricting the domain of $\xi^{\triangleright}$ Now we get back to our problem and change $\mathcal{B}$ so that the size of $\mathrm{dom}(\xi^{\triangleright})$ in a state $\mathrm{st} = (\overline{s}, \chi, \eta, \xi^{\triangleleft}, \xi^{\triangleright}, \gamma)$ is bounded by $|Act| \cdot (2|Q^{\triangleright}| + 1)$. Note that a transition of $\mathcal{B}$ may initiate at most $|Act|$ new copies of $\mathcal{C}^{\triangleright}$ (one for each $b \in Act$ such that $\mathrm{Next}_b \in \mathrm{dom}(g)$). Hence, we say that state st is $\triangleright$-*safe* if for all $b \in Act$ we have $|\mathrm{dom}(\xi^{\triangleright}(b))| \leq 2|Q^{\triangleright}|$. Only transitions between $\triangleright$-safe states of $\mathcal{B}$ are retained in $\mathcal{B}'$.

For a state st, $b \in Act$, $s^{\triangleright} \in Q^{\triangleright}$, we define $\mathrm{Activ}_{\mathrm{st}}(b, s^{\triangleright}) = \{i \mid \xi^{\triangleright}(b, i) = (s^{\triangleright}, I), I \in \mathcal{I}\}$. Also for $i \in \mathrm{Activ}_{\mathrm{st}}(b, s^{\triangleright})$, we denote by I_i the interval such that $\xi^{\triangleright}(b, i) = (s^{\triangleright}, I_i)$. Now, if the state st is not $\triangleright$-safe, then there exist $b \in Act$ and $s^{\triangleright} \in Q^{\triangleright}$ such that we have $|\mathrm{Activ}_{\mathrm{st}}(b, s^{\triangleright})| > 2$. In this case, we say that st is $\triangleright$-unsafe for $(b, s^{\triangleright})$. Now, for each $(b, s^{\triangleright})$ such that st is $\triangleright$-unsafe for $(b, s^{\triangleright})$, we can find $i_\ell, i_u \in \mathrm{Activ}_{\mathrm{st}}(b, s^{\triangleright})$ such that the lower guard defined by $z^{\triangleright}_{b,i_\ell} \in I_{i_\ell}$ is stronger than all lower guards defined by $z^{\triangleright}_{b,j} \in I_j$ for $j \in \mathrm{Activ}_{\mathrm{st}}(b, s^{\triangleright})$ and the upper guard defined by $z^{\triangleright}_{b,i_u} \in I_{i_u}$ is stronger than all upper guards defined by $z^{\triangleright}_{b,j} \in I_j$ for $j \in \mathrm{Activ}_{\mathrm{st}}(b, s^{\triangleright})$. From the definition of the relation *stronger than*, all constraints $z^{\triangleright}_{b,j} \in I_j$ for $j \in \mathrm{Activ}_{\mathrm{st}}(b, s^{\triangleright})$ are subsumed by the conjunction of $z^{\triangleright}_{b,i_\ell} \in I_{i_\ell}$ and $z^{\triangleright}_{b,i_u} \in I_{i_u}$. Therefore, we can release all *next* constraints associated with (b, j) with $j \in \mathrm{Activ}_{\mathrm{st}}(b, s^{\triangleright}) \setminus \{i_\ell, i_u\}$.

To do this in the automaton, we define guards of the form $\varphi(i_\ell^{b,s^{\triangleright}}, i_u^{b,s^{\triangleright}})$ that evaluate to true if $i_\ell^{b,s^{\triangleright}}$ and $i_u^{b,s^{\triangleright}}$ determine stronger lower and upper guards among those defined by $\mathrm{Activ}_{\mathrm{st}}(b, s^{\triangleright})$. Since the relation *stronger than* can be expressed with diagonal constraints as we have seen above, we have $\varphi(i_\ell^{b,s^{\triangleright}}, i_u^{b,s^{\triangleright}}) \in \mathrm{Form}(\mathcal{Z}_{\mathcal{B}})$.

Thus, for each transition $(\mathrm{st}, \varphi, a, R, \mathrm{st}')$ in $\mathcal{B}$ from a $\triangleright$-safe state st to a state $\mathrm{st}' = (\overline{s}', \chi', \eta', \xi'^{\triangleleft}, \xi'^{\triangleright}, \gamma')$ that is not $\triangleright$-safe, we replace it by a transition $(\mathrm{st}, \varphi', a, R, \widetilde{\mathrm{st}'})$, where $\varphi' = \varphi \wedge (\bigwedge_{|\mathrm{Activ}_{\mathrm{st}'}(b,s^{\triangleright})|>2} \varphi(i_\ell^{b,s^{\triangleright}}, i_u^{b,s^{\triangleright}}))$ and $\widetilde{\mathrm{st}'} = (\overline{s}', \chi', \eta', \xi'^{\triangleleft}, \widetilde{\xi'^{\triangleright}}, \gamma')$ is such that

$$\widetilde{\xi'^{\triangleright}}(b, i) = \begin{cases} \xi'^{\triangleright}(b, i) & \text{if } \exists s^{\triangleright} \in Q^{\triangleright} \text{ s.t., } i \in \mathrm{Activ}_{\mathrm{st}'}(b, s^{\triangleright}) \text{ and } (|\mathrm{Activ}_{\mathrm{st}'}(b, s^{\triangleright})| \leq 2) \\ & \text{or } (|\mathrm{Activ}_{\mathrm{st}'}(b, s^{\triangleright})| > 2 \wedge i \in \{i_\ell^{b,s^{\triangleright}}, i_u^{b,s^{\triangleright}}\}) \\ \text{undefined} & \text{otherwise} \end{cases}$$

Then, observe that $\widetilde{\mathrm{st}'}$ is a $\triangleright$-safe state. From the discussion above, we obtain that $\mathcal{B}$ and $\mathcal{B}'$ still accept the same set of timed words. Hence we may conclude that in $\mathcal{B}'$, we can restrict to the *finite* index set $\mathrm{Ind}_{fin} = Act \times \{0, \ldots, n\}$ where $n = \max\{|Q^{\triangleleft}|, 2|Q^{\triangleright}|\}$. Consequently, $\mathcal{B}'$ uses finitely many states and clocks.

Theorem 7.4 *The timed automaton $\mathcal{B}'$ is finite, and we have $\mathcal{L}_{tw}(\mathcal{B}') = \mathcal{L}_{tw}(\mathcal{B})$. Moreover, $\mathcal{B}'$ has $(B+1)^{\mathcal{O}(|Act|)}$ many clocks.*

Proof We first note that the finiteness of $\mathcal{B}'$ follows immediately from the fact that $\mathrm{Ind}_{fin} = Act \times \{0, \ldots, n\}$ is finite (where $n = \max\{|Q^{\triangleleft}|, 2|Q^{\triangleright}|\}$). Using (13) and (14), we deduce that the number of clocks of $\mathcal{B}'$ is bounded by $\mathcal{O}(|Act| \times 2^{|Act|} \times (B+1)^{2|Proc|^2})$.

To prove that $\mathcal{B}$ and $\mathcal{B}'$ accept the same timed language we will use an alternative definition of an accepting run of a timed automaton, which has moves with regard to time-elapse instead of time stamps.

In the following, we say that $(b,i) \in \mathrm{dom}(\xi_1^{\triangleleft})$ is $\triangleleft$-*live* if there is no $j \neq i$ such that $(b,j) \in \mathrm{dom}(\xi_1^{\triangleleft})$ and $\xi_1^{\triangleleft}(b,i) = \xi_1^{\triangleleft}(b,j)$. Moreover, we call $(b,i) \in \mathrm{dom}(\xi_1^{\triangleright})$ to be $\triangleright$-*live* if $i \in \mathrm{Activ}_{\mathrm{st}}(b, s^{\triangleright})$ and either $(|\mathrm{Activ}_{\mathrm{st}'}(b, s^{\triangleright})| \leq 2)$ or $(|\mathrm{Activ}_{\mathrm{st}'}(b, s^{\triangleright})| > 2$ and $i \in \{i_\ell, i_u\})$.

We define a relation $\rightsquigarrow$ between configurations of $\mathcal{B}$ and $\mathcal{B}'$ and let $(\mathrm{st}_1, \nu_1) \rightsquigarrow (\mathrm{st}_2, \nu_2)$ if the following conditions (i)–(iii) hold:

(i) $\mathrm{st}_1 = (\overline{s}, \chi, \eta, \xi_1^{\triangleleft}, \xi_1^{\triangleright}, \gamma)$ and $\mathrm{st}_2 = (\overline{s}, \chi, \eta, \xi_2^{\triangleleft}, \xi_2^{\triangleright}, \gamma)$.

(ii) For all $\triangleleft$-live $(b,i) \in \mathrm{dom}(\xi_1^{\triangleleft})$, there is k such that $(b,k) \in \mathrm{dom}(\xi_2^{\triangleleft})$, $\xi_2^{\triangleleft}(b,k) = \xi_1^{\triangleleft}(b,i)$, and $\nu_1(z^{\triangleleft}_{(b,i)}) = \nu_2(z^{\triangleleft}_{(b,k)})$. Conversely, for all $(b,k) \in \mathrm{dom}(\xi_2^{\triangleleft})$, there is i such that $(b,i) \in \mathrm{dom}(\xi_1^{\triangleleft})$ is $\triangleleft$-live, $\xi_2^{\triangleleft}(b,k) = \xi_1^{\triangleleft}(b,i)$, and $\nu_1(z^{\triangleleft}_{(b,i)}) = \nu_2(z^{\triangleleft}_{(b,k)})$.

(iii) For all $\triangleright$-live $(b,i) \in \mathrm{dom}(\xi_1^{\triangleright})$, there is k such that $(b,k) \in \mathrm{dom}(\xi_2^{\triangleright})$, $\xi_2^{\triangleright}(b,k) = \xi_1^{\triangleright}(b,i)$, and $\nu_1(z^{\triangleright}_{(b,i)}) = \nu_2(z^{\triangleright}_{(b,k)})$. Conversely, for all $(b,k) \in \mathrm{dom}(\xi_2^{\triangleright})$, there is i such that $(b,i) \in \mathrm{dom}(\xi_1^{\triangleright})$ is $\triangleright$-live, $\xi_2^{\triangleright}(b,k) = \xi_1^{\triangleright}(b,i)$, and $\nu_1(z^{\triangleright}_{(b,i)}) = \nu_2(z^{\triangleright}_{(b,k)})$.

We show that the relation $\rightsquigarrow$ is a bisimulation. That is, if $(\mathrm{st}_1, \nu_1) \rightsquigarrow (\mathrm{st}_2, \nu_2)$, then

- for every move $(\mathrm{st}_1, \nu_1) \xrightarrow{a,\tau} (\mathrm{st}_1', \nu_1')$, there exists a move $(\mathrm{st}_2, \nu_2) \xrightarrow{a,\tau} (\mathrm{st}_2', \nu_2')$ such that $(\mathrm{st}_1', \nu_1') \rightsquigarrow (\mathrm{st}_2', \nu_2')$, and
- conversely, for every move $(\mathrm{st}_2, \nu_2) \xrightarrow{a,\tau} (\mathrm{st}_2', \nu_2')$, there exists a move $(\mathrm{st}_1, \nu_1) \xrightarrow{a,\tau} (\mathrm{st}_1', \nu_1')$ such that $(\mathrm{st}_1', \nu_1') \rightsquigarrow (\mathrm{st}_2', \nu_2')$

To prove the first direction, let $\mathrm{st}_1 = (\overline{s}, \chi, \eta, \xi_1^{\triangleleft}, \xi_1^{\triangleright}, \gamma)$ and $\mathrm{st}_2 = (\overline{s}, \chi, \eta, \xi_2^{\triangleleft}, \xi_2^{\triangleright}, \gamma)$. Then, $(\mathrm{st}_1, \nu_1) \xrightarrow{a,\tau} (\mathrm{st}_1', \nu_1')$ with $\mathrm{st}_1' = (\overline{s}', \chi', \eta', \xi_1'^{\triangleleft}, \xi_1'^{\triangleright}, \gamma')$ if $(\mathrm{st}_1, \varphi, a, R, \mathrm{st}_1') \in \delta_{\mathcal{B}}$ for some φ and R with $\nu_1 + \tau \models \varphi$ and $\nu_1' = (\nu_1 + \tau)[R \mapsto 0]$. We can now define φ' and R' by replacing each occurrence of $z^{\triangleleft}_{(b,i)}$ (and $z^{\triangleright}_{(b,i)}$) in φ by $z^{\triangleleft}_{(b,k)}$ (respectively $z^{\triangleright}_{(b,k)}$) for some k given by condition (ii) (respectively, (iii)). Then, $(\mathrm{st}_2, \varphi', a, R', \mathrm{st}_2') \in \delta_{\mathcal{B}'}$ where $\mathrm{st}_2' = (\overline{s}', \chi', \eta', \xi_2'^{\triangleleft}, \xi_2'^{\triangleright}, \gamma')$ with $\xi_2'^{\triangleleft}$ and $\xi_2'^{\triangleright}$ obtained from the definition of the respective modified transition relation.

Now, by Proposition 7.2, each *previous* clock mentioned in φ has an image in φ' which, by definition, has the same constraint. Again, if φ mentions a *next* clock, then either it itself is in φ' or there exists some other clocks in φ' whose upper guards and lower guards are stronger than it. We can conclude that $\nu_1 + \tau \models \varphi$ iff $\nu_2 + \tau \models \varphi'$ and $\nu_2' = (\nu_2 + \tau)[R' \mapsto 0]$. Thus, we have $(\mathrm{st}_2, \nu_2) \xrightarrow{a,\tau} (\mathrm{st}_2', \nu_2')$ and we are done once we see that $(\mathrm{st}_1', \nu_1') \rightsquigarrow (\mathrm{st}_2', \nu_2')$. Condition (i) is already true. Consider condition (ii). For $(b,i) \in \mathrm{dom}(\xi_1'^{\triangleleft})$, there are two cases to consider. First, it was defined here, i.e., $b = a$ and $i = \min(\mathbb{N} \setminus \mathrm{dom}(\xi_1^{\triangleleft}(a)))$. In this case, (b,i) is not $\triangleleft$-unsafe at st_1' iff (b,k) is not $\triangleleft$-unsafe at st_2', for $k = \min(\mathbb{N} \setminus \mathrm{dom}(\xi_2^{\triangleleft}(a)))$. Second, it was updated from the previous state (i.e., $(b,i) \in \mathrm{dom}(\xi_1^{\triangleleft})$). Now, if it was updated from the previous state st_1 and it is not $\triangleleft$-unsafe at st_1', then it was not $\triangleleft$-unsafe at st_1 either (because, otherwise, there is $(b,j) \in \mathrm{dom}(\xi_1^{\triangleleft})$ such that $\xi_1^{\triangleleft}(b,i) = \xi_1^{\triangleleft}(b,j)$ which implies that $\delta^{\triangleleft}(\xi_1^{\triangleleft}(b,i), (a, \eta(a))) = \delta^{\triangleleft}(\xi_1^{\triangleleft}(b,j), (a, \eta(a)))$ which implies that (b,i) is $\triangleleft$-unsafe at st_1'). Then, as $\mathrm{st}_1 \rightsquigarrow \mathrm{st}_2$, there exists k such that $\xi_2^{\triangleleft}(b,k) = \xi_1^{\triangleleft}(b,i)$. Now, (b,k)

cannot be $\lhd$-unsafe at st_2' since, otherwise, we obtain that (b,i) will be $\lhd$-unsafe at st_1'. Thus, $\xi_2'^{\lhd}(b,k) = \delta^{\lhd}(\xi_2^{\lhd}(b,k)(a,\eta(a))) = \delta^{\lhd}(\xi_1^{\lhd}(b,j)(a,\eta(a))) = \xi_1'^{\lhd}(b,i)$. Conversely, if there exists $(b,k) \in \mathrm{dom}(\xi_2'^{\lhd})$ and $\xi_2'^{\lhd}(b,k) = \xi_1'^{\lhd}(b,i)$, then there cannot exist $(b,j) \in \mathrm{dom}(\xi_1'^{\lhd})$ such that $\xi_1'^{\lhd}(b,j) = \xi_1'^{\lhd}(b,i)$. By similar arguments, we see that condition (iii) also holds.

In the converse direction, for *previous* or *next* index (b,i) of st_1 which does not have a corresponding index in st_2, we use a fresh *previous* or *next* clock and use the same arguments as above.

Thus, $\rightsquigarrow$ is a bisimulation, and now from the fact that the final states of $\mathcal{B}$ and $\mathcal{B}'$ coincide, we conclude that the timed languages are the same. □

From Theorems 7.2, 7.3, and 7.4, we obtain the proof of the main result, i.e., the decidability of emptiness for ECMPA over existentially B-bounded *timed* MSCs. To conclude the proof of Theorem 7.1, it remains to discuss the complexity of our algorithm.

Recall from [2] that the emptiness problem for timed automata is PSPACE-complete. More precisely, the upper bound is obtained with an NLOGSPACE reachability analysis of the region graph, the number of regions being polynomial in the number of states of the automaton but exponential in the number of clocks.

By Theorem 7.4, the number of clocks of $\mathcal{B}'$ is $(B+1)^{\mathcal{O}(|Act|)}$. Each state of $\mathcal{B}'$ is a tuple of the form $(\overline{s}, \chi, \eta, \xi^{\lhd}, \xi^{\rhd}, \gamma)$ as defined in Sect. 7.2 but using the finite set of indices Ind_{fin}. From (13) and (14) we deduce that $|Q^{\lhd}|$, $|Q^{\rhd}|$ and $|\mathrm{Ind}_{fin}|$ are all in $(B+1)^{\mathcal{O}(|Act|)}$. Then, we can check that the number of states of $\mathcal{B}'$ is bounded by $2^{P(|\mathcal{A}|,(B+1)^{|Act|})}$ for some polynomial P.

Finally, combining the bounds above on the number of clocks and the number of states of $\mathcal{B}'$ with the NLOGSPACE reachability analysis of the region graph, we can check emptiness using space polynomial in $|\mathcal{A}|$ and $(B+1)^{|Act|}$ as announced in Theorem 7.1.

8 Conclusion

In this paper, we have introduced ECMPA as an automaton model to deal with timed and distributed systems. These combine the generality of message passing automata (in the distributed untimed setting) with the tractability of event clock automata (in the sequential timed setting). To describe the behaviors of such systems we have used two formalisms of timed partial orders, i.e., TMSCs and TCMSCs. The first main result shows that timed monadic second-order logic formulae and ECMPA are expressively equivalent over TMSCs. Indeed, this equivalence holds without assumptions on the bounds of channels, only when we restrict to the existential fragment of the logic. Further, the proof of this equivalence is constructive, since we are able to formulate an explicit translation between the two. The second main result proves that checking emptiness for ECMPA is decidable in the bounded case. These two results together allow us to check satisfiability for the timed logic.

Acknowledgements We thank the anonymous referees for their constructive suggestions which helped in improving the presentation of the paper.

References

1. Akshay S, Bollig B, Gastin P (2007) Automata and logics for timed message sequence charts. In: FSTTCS. LNCS, vol 4855. Springer, Berlin, pp 290–302
2. Alur R, Dill DL (1994) A theory of timed automata. Theor Comput Sci 126(2):183–235

3. Alur R, Fix L, Henzinger TA (1999) Event-clock automata: A determinizable class of timed automata. Theor Comput Sci 211(1–2):253–273
4. Alur R, Holzmann G, Peled D (1996) An analyser for message sequence charts. In: TACAS. LNCS, vol 1055. Springer, Berlin, pp 35–48
5. Ben-Abdallah H, Leue S (1997) Timing constraints in message sequence chart specifications. In: FORTE. IFIP Conf Proc, vol 107, pp 91–106
6. Berthomieu B, Diaz M (1991) Modeling and verification of time dependent systems using time Petri nets. IEEE Trans Softw Eng 17(3):259–273
7. Bollig B, Leucker M (2006) Message-passing automata are expressively equivalent to EMSO logic. Theor Comput Sci 358(2–3):150–172
8. Bouyer P, Haddad S, Reynier P-A (2006) Timed unfoldings for networks of timed automata. In: ATVA. LNCS, vol 4218. Springer, Berlin, pp 292–306
9. Brand D, Zafiropulo P (1983) On communicating finite-state machines. J ACM 30(2):323–342
10. Büchi J (1960) Weak second order logic and finite automata. Math Log Q 5:66–92
11. Chandrasekaran P, Mukund M (2006) Matching scenarios with timing constraints. In: FORMATS. LNCS, vol 4202. Springer, Berlin, pp 91–106
12. Chatain Th, Jard C (2005) Time supervision of concurrent systems using symbolic unfoldings of time Petri nets. In: FORMATS. LNCS, vol 3829. Springer, Berlin, pp 196–210
13. D'Souza D (2000) A logical study of distributed timed automata. PhD thesis, BITS Pilani
14. D'Souza D (2003) A logical characterisation of event clock automata. Int J Found Comput Sci 14(4):625–640
15. D'Souza D, Thiagarajan PS (1999) Product interval automata: a subclass of timed automata. In: FSTTCS. LNCS, vol 1738. Springer, Berlin, pp 60–71
16. Elgot CC (1961) Decision problems of finite automata design and related arithmetics. Trans Am Math Soc 98:21–52
17. Genest B, Kuske D, Muscholl A (2006) A Kleene theorem and model checking algorithms for existentially bounded communicating automata. Inf Comput 204(6):920–956
18. Godefroid P (1996) Partial-order methods for the verification of concurrent systems—an approach to the state-explosion problem. LNCS, vol 1032. Springer, Berlin
19. Henriksen JG, Mukund M, Narayan Kumar K, Sohoni M, Thiagarajan PS (2005) A theory of regular MSC languages. Inf Comput 202(1):1–38
20. Krcal P, Yi W (2006) Communicating timed automata: the more synchronous, the more difficult to verify. In: CAV. LNCS, vol 4144. Springer, Berlin, pp 243–257
21. Narayan Kumar K (2012) The theory of MSC languages. In: D'Souza D, Shankar P (eds) Modern Applications of Automata Theory. IISc Res Monographs, vol 2. World Scientific, Singapore, pp 289–324
22. Lohrey M, Muscholl A (2004) Bounded MSC communication. Inf Comput 189(2):160–181
23. Ramchandani C (1974) Analysis of asynchronous concurrent systems by timed Petri nets. PhD thesis, Massachusetts Institute of Technology