Automata and Logics for Timed Message Sequence Charts

S. Akshay[1,2], Benedikt Bollig[1], and Paul Gastin[1]

[1] LSV, ENS Cachan, CNRS, France
[2] Institute of Mathematical Sciences, Chennai, India

Abstract. We provide a framework for distributed systems that impose timing constraints on their executions. We propose a timed model of communicating finite-state machines, which communicate by exchanging messages through channels and use event clocks to generate collections of timed message sequence charts (T-MSCs). As a specification language, we propose a monadic second-order logic equipped with timing predicates and interpreted over T-MSCs. We establish expressive equivalence of our automata and logic. Moreover, we prove that, for (existentially) bounded channels, emptiness and satisfiability are decidable for our automata and logic.

1 Introduction

One of the most famous connections between automata theory and classical logic, established in the early sixties by Büchi and Elgot [7], is the equivalence of finite-state machines and monadic second-order logic (MSO) over words. This study of relations between logical formalisms and automata has had many generalizations including extensions and abstractions of the definition of words themselves.

A natural extension, for instance, are timed words which are very important in the context of verification of safety critical timed systems. For this, we have automata models such as timed automata [1] and event-clock automata (ECA) [2]. The latter have implicit clocks allowing them to record or predict time lapses. This is well-suited for real-time specifications (such as bounded response time) and allows for a suitable logical characterization by a timed MSO over timed words as shown in [9].

On the other hand, in a distributed setting, we might have several agents interacting to generate a global behavior. This interaction can be specified using message sequence charts (MSCs) which generalize words and reflect the causality of events in a system execution. MSCs have been known for a long time independently, as they serve as documentation of design requirements that are referred throughout the design process and even in the final system integration and acceptance testing. MSCs are used for describing the behavior of communicating finite-state machines (CFMs) [6], which are a fundamental model for concurrent systems and communicating protocols. These CFMs have communicating channels between the constituent finite-state automata and a single MSC diagram subsumes a whole set of sequential runs of the CFM.

Our goal is to merge the timed and distributed approaches mentioned above. For this, we first consider timed MSCs (T-MSCs) which are just MSCs with time stamps at events (as in timed words). These are ideal to describe real-time system executions, keeping explicitly the causal relation between events. Next, we consider MSCs with timing constraints (TC-MSCs) where we associate lower and upper bounds on the time interval between certain pairs of events. This is more suitable for a specifier and also useful to describe a (possibly infinite) family of T-MSCs in a finite way.

We introduce event clock communicating finite-state machines (EC-CFM) recognizing timed MSCs. These are CFMs equipped with implicit event clocks allowing us to record or predict time lapses as in the ECA. For the logical framework, we use a timed version of monadic second-order logic (TMSO) with additional timing predicates to specify necessary timing constraints. We interpret both EC-CFMs and TMSO over T-MSCs and prove a constructive equivalence between them, with and without bounds on channels. This is done by lifting the corresponding results from the untimed case [12, 11, 5] by using TC-MSCs, since they can be seen as MSCs whose labelings are extended by timing information and also as a representation of infinite sets of T-MSCs.

Further, we prove that, over "existentially bounded" channels, the emptiness checking of our automaton model and, thus, the satisfiability problem of our logic are decidable. Our approach consists of constructing a global finite timed automaton that can simulate the runs of the EC-CFM (which is a distributed machine) and so, reduce the problem to emptiness checking on a timed automaton. The hard part of the construction lies in "cleverly" maintaining the partial-order information (of the T-MSC) along the sequential runs of the global timed automaton, while using only finitely many clocks.

Related Work Past approaches to timing in MSCs with a formal semantics and analysis have been looked at in [3, 4, 8, 13]. While [3] and [4] only consider single MSCs or high-level MSCs, one of the first attempts to study channel automata in the timed setting goes back to Krcal and Yi [13], who provide local timed automata with the means to communicate via FIFO channels. They do not consider MSCs as a semantics of their automata but rather look at restricted channel architectures (e.g., one-channel systems) to transfer decidability of reachability problems from the untimed to the timed setting. A similar automaton model was independently introduced by Chandrasekaran and Mukund in [8], who even define its semantics in terms of timed MSCs. They propose a practical solution to a very specific matching problem using the tool UPPAAL.

Outline We define MSCs in Section 2, together with their timed extensions. Our logic and the automaton model are introduced in Section 3. We describe the equivalence results between our automata and logic over timed MSCs in Section 4. In Section 5, it is shown that emptiness of automata is decidable for existentially bounded channels.

2 Timed Message Sequence Charts

We fix a finite set Ag of at least two *agents* or *processes*. The set of *communication actions* on process p is $Act_p = \{p!q \mid q \in Ag \setminus \{p\}\} \cup \{p?q \mid q \in Ag \setminus \{p\}\}$, where $p!q$ means that process p sends a message to process q and $p?q$ means that process p receives a message from process q. Furthermore, we let $Act = \bigcup_{p \in Ag} Act_p$.

An *Act-labeled partial order* is a triple $M = (E, \preceq, \lambda)$ where $(E, \preceq)$ is a finite partial order (elements from E are called *events*) and $\lambda : E \to Act$ is a labeling function. For $e \in E$, ${\downarrow}e$ denotes $\{e' \in E \mid e' \preceq e\}$. We define a message relation $\mathrm{Msg}^M \subseteq E \times E$ matching send events with their corresponding receives, assuming a FIFO architecture on the channels. That is, $(e, e') \in \mathrm{Msg}^M$ if $\lambda(e) = p!q$ and $\lambda(e') = q?p$ for some $p, q \in Ag$, and $|{\downarrow}e \cap \lambda^{-1}(p!q)| = |{\downarrow}e' \cap \lambda^{-1}(q?p)|$.

A *message sequence chart* (MSC) is an Act-labeled partial order $M = (E, \preceq, \lambda)$ such that (i) for any $p \in Ag$, the restriction of $\preceq$ to process p (denoted $\preceq_p$) is a total order, (ii) the partial order $\preceq$ is the transitive closure of $\mathrm{Msg}^M \cup \bigcup_{p \in Ag} \preceq_p$, and (iii) for any distinct $p, q \in Ag$, the number of send events is equal to the number of receive events, i.e, $|\lambda^{-1}(p!q)| = |\lambda^{-1}(q?p)|$.

Fig. 1 depicts an MSC as a diagram. The events of each process are arranged along the vertical lines and messages are shown as horizontal or downward-sloping directed edges. Note that $\lambda(e_1) = p!q$, $\lambda(e_2) = q?p$, $e_1 \preceq_p e_1'$, $(e_2', e_3) \in \mathrm{Msg}^M$ and $e_1 \preceq e_3$. The linearizations of an MSC form a word language over Act under λ. E.g., $(p!q)(q?p)(q!r)(p!r)(r?q)(r?p)$ is one linearization of the MSC in Fig. 1. An MSC is uniquely determined by one of its linearizations.

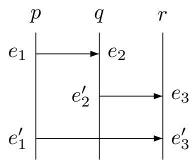


Fig. 1. An MSC

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 action. This approach is quite realistic when we want to model the real-time execution of concurrent systems.

Definition 1. *A* timed MSC *(T-MSC) is a tuple* $(E, \preceq, \lambda, t)$ *where* $(E, \preceq, \lambda)$ *is an MSC and* $t : E \to \mathbb{R}^{\geq 0}$ *is a function such that if* $e_1 \preceq e_2$ *then* $t(e_1) \leq t(e_2)$*. The set of all T-MSCs is denoted* $\mathbb{TMSC}$*.*

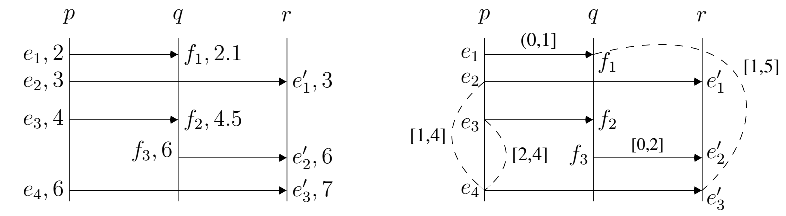
A *timed linearization* of a T-MSC is a possible execution in terms of a word from $(Act \times \mathbb{R}^{\geq 0})^*$, which thus respects both the causal order and the order imposed by the time stamping. A T-MSC is shown in 2(a). Note that it has several timed linearizations as the concurrent events e_4 and f_3 occur at the same time. A possible timed linearization is $(p!q, 2)(q?p, 2.1)(p!r, 3)(r?p, 3)(p!q, 4)(q?p, 4.5)(p!r, 6)(q!r, 6)(r?q, 6)(r?p, 7)$.

Now a family of T-MSCs with the same induced MSC can be specified by timing constraints on pairs of events of the MSC. This approach is better suited to a specifier who can then decide and enforce constraints between occurrences of events. As an example consider Fig. 2(b). The label $(0, 1]$ on message from e_1 to f_1 specifies the lower bound and upper bound on the delay of message delivery. The label $[1, 5]$ from f_1 to e_3' represents the bounds on the delay between f_1 and e_3' and so on.

The question here is how flexible do we want this timing to be, i.e, between which pairs of events do we allow constraints. For an MSC $M = (E, \preceq, \lambda)$, one obvious set of pairs is given by Msg^M which allows us to time messages. A more flexible approach is to allow timing between the next (or previous) event of any action and an event in the MSC. For this, we define the relations $\mathrm{Next}_\sigma^M, \mathrm{Prev}_\sigma^M$ for every $\sigma \in Act$ as follows:

- $\mathrm{Next}_\sigma^M = \{(e, e') \mid \lambda(e') = \sigma,\ e \prec e',\ (e \prec e'' \wedge \lambda(e'') = \sigma) \implies e' \preceq e''\}$
- $\mathrm{Prev}_\sigma^M = \{(e, e') \mid \lambda(e') = \sigma,\ e' \prec e,\ (e'' \prec e \wedge \lambda(e'') = \sigma) \implies e'' \preceq e'\}$

E.g., in Fig. 2(b), $(e_2, e_4) \in \mathrm{Next}_{p!r}^M$, $(f_1, e_3') \in \mathrm{Next}_{r?p}^M$, and $(e_4, e_3) \in \mathrm{Prev}_{p!q}^M$. Note that these relations are in fact partial maps and hence one can also write $f = \mathrm{Next}_\sigma^M(e)$ for $(e, f) \in \mathrm{Next}_\sigma^M$ and similarly for Prev_σ^M. In fact Msg^M can also been seen as a partial function $E \dashrightarrow E$ mapping a send event to its corresponding receive in the MSC M. Further, we remark that these relations can all be defined for a T-MSC T as well. Since they depend only on the underlying partial order, we write Msg^T, Next_σ^T, etc.



(a) (b)

Fig. 2. A T-MSC and a TC-MSC

Let us denote the set of symbols $\{\mathrm{Msg}\} \cup \{\mathrm{Prev}_\sigma \mid \sigma \in \mathit{Act}\} \cup \{\mathrm{Next}_\sigma \mid \sigma \in \mathit{Act}\}$ by TC (for timing constraints). For an MSC (or T-MSC) M, we let $\mathrm{TC}^M = \bigcup_{\alpha \in \mathrm{TC}}(\alpha^M)$ be our set of allowed timing pairs. This is flexible enough to specify what we need. It also generalizes the approach of D'Souza [9] in the timed words case. Further, this is similar to the approach adopted by Alur et al. [3] to time MSCs and so we can use their analysis tool to check consistency of the timing constraints in an MSC.

To specify timing constraints we will use rational bounded intervals over the real line. These can be open or closed intervals but we require them to be nonempty and the bounds to be rational. The set of all such intervals is denoted by $\mathcal{I}$.

Definition 2. *An* MSC with timing constraints *(TC-MSC) is a tuple* $(E, \preceq, \lambda, \tau)$ *where* $M = (E, \preceq, \lambda)$ *is an MSC and* $\tau : \mathrm{TC}^M \dashrightarrow \mathcal{I}$ *is a partial function. The TC-MSC is called* maximally defined *if* τ *is a total function.*

With this definition, TC-MSCs can be considered as abstractions of T-MSCs and timed words. Let $M = (E, \preceq, \lambda, \tau)$ be a TC-MSC. A T-MSC $T = (E, \preceq, \lambda, t)$ is a *realization* of M if, for all $(e, e') \in \mathrm{dom}(\tau)$, we have $|t(e) - t(e')| \in \tau(e, e')$. Thus for instance, the T-MSC in Fig. 2(a) is a realization of the TC-MSC in Fig. 2(b).

3 Logic and Automata for Timed MSCs

Monadic Second-Order Logic We will define several monadic second-order logics as a means to describe sets of T-MSCs. Their syntax depends on a set $\mathcal{R}$ of (binary) relation symbols, which settles the access to the partial-order relation of an MSC or T-MSC. One example is $\mathcal{R}_\preceq = \{\preceq, \mathrm{Msg}\}$ containing symbols for the partial order and the message relation. The formal syntax of our logic $\mathrm{TMSO}(\mathcal{R})$ is given by:

$$\varphi ::= P_\sigma(x) \mid x \in X \mid x = y \mid R(x, y) \mid \delta(x, \alpha(x)) \in I \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \varphi \mid \exists X \varphi$$

where $\sigma \in \mathit{Act}$, $R \in \mathcal{R}$, $\alpha \in \mathrm{TC}$, $I \in \mathcal{I}$, x, y are individual (or first-order) variables, and X is a set (or second-order) variable (each from an infinite supply of variables).

Let $T = (E, \preceq, \lambda, t)$ be a T-MSC and let $\mathbb{I}$ be an *interpretation* that maps first-order variables to elements in E and second-order variables to subsets of E. Let us define when $T, \mathbb{I} \models \varphi$ for $\varphi \in \mathrm{TMSO}(\mathcal{R})$. As usual, $P_\sigma(x)$ expresses that x is labeled with σ, i.e., $\lambda(\mathbb{I}(x)) = \sigma$. The novelty is the timing predicate $\delta(x, \alpha(x)) \in I$ by which we mean that the time difference between x and $\alpha^T(x)$ is contained in I, i.e., $T, \mathbb{I} \models \delta(x, \alpha(x)) \in I$ if $\mathbb{I}(x) \in \mathrm{dom}(\alpha^T)$ and $|t(\mathbb{I}(x)) - t(\alpha^T(\mathbb{I}(x)))| \in I$. For the set $\mathcal{R}$ of binary relation symbols we will use $\mathcal{R}_{\preceq} = \{\preceq, \mathrm{Msg}\}$ or $\mathcal{R}_{\prec} = \{\prec_p \mid p \in Ag\} \cup \{\mathrm{Msg}\}$. The interpretation of $\lessdot_p$ is the immediate successor relation on process p: $\lessdot_p := \prec_p \setminus \prec_p{}^2$. The interpretation of Msg is indeed Msg^T. The rest of the semantics is classical for MSO logics. For sentences φ in this logic, we define $\mathcal{L}_{time}(\varphi) = \{T \in \mathbb{TMSC} \mid T \models \varphi\}$. The *existential* fragment of $\mathrm{TMSO}(\mathcal{R})$, which is denoted by $\mathrm{ETMSO}(\mathcal{R})$, comprises all formulas $\exists X_1 \ldots \exists X_n \varphi$ such that φ does not contain any set quantifier.

We will give TMSO formulas a natural semantics in terms of TC-MSCs, too. The only noteworthy difference is in the atomic predicate $\delta(x, \alpha(x)) \in I$. For a TC-MSC $M = (E, \preceq, \lambda, \tau)$, we define $M, \mathbb{I} \models \delta(x, \alpha(x)) \in I$ if $\tau(\mathbb{I}(x), \alpha^M(\mathbb{I}(x))) \subseteq I$, which implicitly implies $\mathbb{I}(x) \in \mathrm{dom}(\alpha^M)$ and $(\mathbb{I}(x), \alpha^M(\mathbb{I}(x))) \in \mathrm{dom}(\tau)$. The set of TC-MSCs that satisfy a TMSO sentence φ is denoted by $\mathcal{L}_{TC}(\varphi)$. The following implication is easy to see. Its converse holds in a restricted case, as we will see later.

Lemma 3. *Let a T-MSC T be a realization of some TC-MSC M and let φ be a TMSO formula. Then, $M \in \mathcal{L}_{TC}(\varphi)$ implies $T \in \mathcal{L}_{time}(\varphi)$.*

Event-Clock Communicating Finite-State Machines (EC-CFMs) A natural model of communication protocols are communicating finite-state machines [6], which consist of finite-state machines with message channels between any pair of them. To introduce the timed model we attach recording and predicting *clocks* (as in [2]) to these machines.

Definition 4. *An* EC-CFM *is a tuple $\mathcal{A} = (C, (\mathcal{A}_p)_{p \in Ag}, F)$ where C is a finite set of control messages, $\mathcal{A}_p = (Q_p, \rightarrow_p, \iota_p)$ is a finite transition system over $Act_p \times [\mathrm{TC} \dashrightarrow \mathcal{I}] \times C$ (i.e., $\iota_p \in Q_p$ is the initial state and $\rightarrow_p$ is a* finite *subset of $Q_p \times Act_p \times [\mathrm{TC} \dashrightarrow \mathcal{I}] \times C \times Q_p$) with $[\mathrm{TC} \dashrightarrow \mathcal{I}]$ denoting the set of partial maps from TC to $\mathcal{I}$, and $F \subseteq \prod_{p \in Ag} Q_p$ is a set of global final states.*

The input of an EC-CFM $\mathcal{A}$ is a T-MSC $T = (E, \preceq, \lambda, t)$. Consider a map $r : E \rightarrow \bigcup_{p \in Ag} Q_p$ labeling each event of process p with a state from Q_p. Define $r^- : E \rightarrow \bigcup_{p \in Ag} Q_p$ as follows: For event e in process p, if there is an event e' in process p such that $e' \lessdot_p e$, then we set $r^-(e) = r(e')$. Otherwise, we set $r^-(e) = \iota_p$. Then r is said to be a *run* of $\mathcal{A}$ on T if, for all $(e, e') \in \mathrm{Msg}^T$ with e in process p and e' in process q, there are guards $g, g' \in [\mathrm{TC} \dashrightarrow \mathcal{I}]$ and a control message $c \in C$ such that

(1) $(r^-(e), \lambda(e), g, c, r(e)) \in \rightarrow_p$ and $(r^-(e'), \lambda(e'), g', c, r(e')) \in \rightarrow_q$,
(2) for all $\alpha \in \mathrm{dom}(g)$, we have $e \in \mathrm{dom}(\alpha^T)$ and $|t(e) - t(\alpha^T(e))| \in g(\alpha)$, and
(3) for all $\alpha \in \mathrm{dom}(g')$, we have $e' \in \mathrm{dom}(\alpha^T)$ and $|t(e') - t(\alpha^T(e'))| \in g'(\alpha)$.

Let r be a run of $\mathcal{A}$ on T. We define $s_p = r(e_p)$, where e_p is the maximal event in process p. If there are no events on process p, we set $s_p = \iota_p$. Then run r is *successful* if the tuple $(s_p)_{p \in Ag}$ belongs to F. A T-MSC is *accepted* by an EC-CFM $\mathcal{A}$ if it admits a successful run. We denote by $\mathcal{L}_{time}(\mathcal{A})$ the set of T-MSCs that are accepted by $\mathcal{A}$.

As in the logic, we can give EC-CFMs a semantics in terms of TC-MSCs as well. For defining a run on TC-MSC $M = (E, \preceq, \lambda, \tau)$ we just replace condition (2) above by saying that, for all $\alpha \in \mathrm{dom}(g)$, we must have $e \in \mathrm{dom}(\alpha^M)$ and $\tau(e, \alpha^M(e)) \subseteq g(\alpha)$. We do the same for condition (3). Then, with the same notion of acceptance as above, we can denote the set of all TC-MSCs accepted by a given EC-CFM $\mathcal{A}$ as $\mathcal{L}_{TC}(\mathcal{A})$.

Lemma 5. *Let T be a realization of some TC-MSC M and let $\mathcal{A}$ be an EC-CFM. Then, $M \in \mathcal{L}_{TC}(\mathcal{A})$ implies $T \in \mathcal{L}_{time}(\mathcal{A})$.*

4 Equivalence of EC-CFMs and MSO Logic

In [5], the equivalence between EMSO formulas (with restricted signature) and CFMs over MSCs has been established. In [11], the equivalence between full MSO formulas and CFMs over MSCs has been described in the context of bounded channels. We will lift these theorems to the timed setting, using the concepts from the previous sections.

Theorem 6. *Let L be a set of T-MSCs. The following are equivalent:*

1. *There is an EC-CFM $\mathcal{A}$ such that $\mathcal{L}_{time}(\mathcal{A}) = L$.*
2. *There is $\varphi \in \mathrm{ETMSO}(\mathcal{R}_{\prec})$ such that $\mathcal{L}_{time}(\varphi) = L$.*

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

The rest of this section is devoted to the construction of an EC-CFM from an ETMSO formula, whose size is elementary in the size of the formula. The basic idea is to reduce this to an analogous untimed result, which has also been applied in the settings of words and traces [9, 10]. For this, we establish a connection between TMSO and ordinary MSO logic without timing predicate, and between EC-CFMs and their untimed variant. Usually, these untimed formalisms are parametrized by a finite alphabet Σ to speak about structures whose labelings are provided by Σ. Hence, in our framework, we need to find a finite abstraction of the infinite set of possible time stamps. Applying this finite abstraction, we move from T-MSCs to TC-MSCs and establish the converse of Lemmas 3 and 5 in Lemmas 8 and 9, resp. This finally allows us to translate ETMSO formulas into EC-CFMs. We provide more details below.

First, we define proper interval sets. We call a set of intervals $\mathcal{S} \subseteq \mathcal{I}$ *proper* if it forms a finite partition of $\mathbb{R}^{\geq 0}$. We say that an interval set *refines* another interval set if every interval of the latter is the union of some collection of intervals of the former. For any finite interval set, we can easily obtain a proper interval set refining it.

Let $T = (E, \preceq, \lambda, t)$ be a T-MSC and $\mathcal{S}$ be a proper interval set. We introduce the TC-MSC $M_T^{\mathcal{S}} := (E, \preceq, \lambda, \tau)$ where, for any $(e, e') \in \mathrm{TC}^T$, $\tau(e, e')$ is defined to be the unique interval of $\mathcal{S}$ containing $|t(e) - t(e')|$.

Lemma 7. *Let T be a T-MSC and let $\mathcal{S}$ be a proper interval set. Then, $M_T^{\mathcal{S}}$ is the unique maximally defined TC-MSC that uses intervals from $\mathcal{S}$ and admits T as realization.*

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 $\delta(x, \alpha(x)) \in I$. Similarly, for any EC-CFM $\mathcal{A}$, we have a *finite* set, denoted $\mathrm{Int}(\mathcal{A})$, of intervals occurring in $\mathcal{A}$ as guards. Now look at any proper interval set $\mathcal{S}$ that refines $\mathrm{Int}(\varphi)$. We can translate the TMSO formula φ to another TMSO formula $\varphi^{\mathcal{S}}$ by replacing each sub-formula of the form $\delta(x, \alpha(x)) \in I$ by the formula $\bigvee_{J \in \mathcal{S} : J \subseteq I} \delta(x, \alpha(x)) \in J$. Using Lemma 7, we can show the following Lemmas, which then enable us to prove the reverse direction of Theorem 6.

Lemma 8. *Given a T-MSC T, a TMSO formula φ, and a proper interval set $\mathcal{S}$ that refines $\mathrm{Int}(\varphi)$, we have $T \models \varphi$ iff $M_T^{\mathcal{S}} \models \varphi$ iff $M_T^{\mathcal{S}} \models \varphi^{\mathcal{S}}$.*

Lemma 9. *Let $\mathcal{A}$ be an EC-CFM and let $\mathcal{S}$ be a proper interval set that refines* $\mathrm{Int}(\mathcal{A})$. *For a T-MSC T, we have $T \in \mathcal{L}_{time}(\mathcal{A})$ iff $M_T^{\mathcal{S}} \in \mathcal{L}_{TC}(\mathcal{A})$.*

Proof (of Theorem 6, (2) $\rightarrow$ (1)). Observe that any TC-MSC can be viewed as an MSC with an additional labeling by removing the intervals from pairs of events and attaching them to the corresponding events. More precisely, a TC-MSC $M = (E, \preceq, \lambda, \tau)$ can be represented as an MSC $\overline{M} = (E, \preceq, \lambda, \gamma)$ with additional labeling $\gamma : E \to (\mathrm{TC} \dashrightarrow \mathcal{I})$ describing the timing constraints, i.e., $\gamma(e)(\alpha) = \tau(e, \alpha^M(e))$ if $e \in \mathrm{dom}(\alpha^M)$ and $(e, \alpha^M(e)) \in \mathrm{dom}(\tau)$; otherwise, $\gamma(e)(\alpha)$ is undefined. This view will allow us to apply equivalences between logic and automata in the untimed case. So far, however, the additional labeling γ is over an infinite alphabet, as there are infinitely many intervals that might act as constraints. So, for any proper interval set $\mathcal{S}$, we define $\mathbb{TCMSC}(\mathcal{S})$ as the set of TC-MSCs $M = (E, \preceq, \lambda, \tau)$ such that $\tau(e, e') \in \mathcal{S}$ for any $(e, e') \in \mathrm{dom}(\tau)$. Note that, if $M \in \mathbb{TCMSC}(\mathcal{S})$ and $I \in \mathcal{S}$ then $M, \mathbb{I} \models \delta(x, \alpha(x)) \in I$ iff $\tau(\mathbb{I}(x), \alpha^M(\mathbb{I}(x))) = I$ iff $\gamma(\mathbb{I}(x))(\alpha) = I$. Hence a timing predicate can be transformed into a labeling predicate: for any $\varphi \in \mathrm{TMSO}$ such that $\mathrm{Int}(\varphi) \subseteq \mathcal{S}$, there is an *untimed* MSO formula $\overline{\varphi}$ such that $M, \mathbb{I} \models \varphi$ iff $\overline{M}, \mathbb{I} \models \overline{\varphi}$ for all $M \in \mathbb{TCMSC}(\mathcal{S})$. In the following, we denote by $\mathcal{L}_u(\varphi)$ the set of MSCs with additional labeling γ that satisfy an untimed MSO sentence φ. We can build an untimed $\mathrm{MSO}(\mathcal{R}_\prec)$ sentence $\mu_\prec^{\mathcal{S}}$ such that $\mathcal{L}_u(\mu_\prec^{\mathcal{S}})$ is the set of *maximally defined* MSCs $M = (E, \preceq, \lambda, \gamma)$ with additional labeling γ using intervals from $\mathcal{S}$, i.e., for all $e \in E$, we have $\alpha \in \mathrm{dom}(\gamma(e))$ iff $e \in \mathrm{dom}(\alpha^M)$ and in this case $\gamma(e)(\alpha) \in \mathcal{S}$.

Similarly, an EC-CFM $\mathcal{A}$ can be interpreted over MSCs with the additional labeling γ by replacing conditions (2) and (3) of runs by $\gamma(e) = g$ and $\gamma(e') = g'$, resp. We denote by $\mathcal{L}_u(\mathcal{A})$ the untimed MSCs with additional labeling γ that are accepted by $\mathcal{A}$. Here, for a TC-MSC $M \in \mathbb{TCMSC}(\mathcal{S})$ and an automaton $\mathcal{A}$ with guards in $[\mathrm{TC} \dashrightarrow \mathcal{S}]$, we have $\overline{M} \in \mathcal{L}_u(\mathcal{A})$ implies $M \in \mathcal{L}_{\mathrm{TC}}(\mathcal{A})$. The converse does not hold in general.

Let $\varphi \in \mathrm{ETMSO}(\mathcal{R}_\prec)$ be the given formula and let $\mathcal{S}$ be a proper interval set that refines $\mathrm{Int}(\varphi)$. Consider the untimed $\mathrm{MSO}(\mathcal{R}_\prec)$-formula $\psi = \overline{\varphi}^{\mathcal{S}} \wedge \mu_\prec^{\mathcal{S}}$. By [5], there is an EC-CFM $\mathcal{A}$ with guards from $[\mathrm{TC} \dashrightarrow \mathcal{S}]$ such that $\mathcal{L}_u(\mathcal{A}) = \mathcal{L}_u(\psi)$. We will show that $\mathcal{L}_{time}(\varphi) = \mathcal{L}_{time}(\mathcal{A})$.

Let T be a T-MSC. By Lemma 8 we have $T \models \varphi$ iff $M_T^{\mathcal{S}} \models \varphi^{\mathcal{S}}$. Since $\mathrm{Int}(\varphi^{\mathcal{S}}) \subseteq \mathcal{S}$ and $M_T^{\mathcal{S}} \in \mathbb{TCMSC}(\mathcal{S})$ we have $M_T^{\mathcal{S}} \models \varphi^{\mathcal{S}}$ iff $\overline{M}_T^{\mathcal{S}} \models \overline{\varphi}^{\mathcal{S}}$. Now, $M_T^{\mathcal{S}}$ is maximally defined, hence we obtain $\overline{M}_T^{\mathcal{S}} \models \mu_\prec^{\mathcal{S}}$. Therefore, $T \in \mathcal{L}_{time}(\varphi)$ iff $\overline{M}_T^{\mathcal{S}} \in \mathcal{L}_u(\psi) = \mathcal{L}_u(\mathcal{A})$. We have seen above that this implies $M_T^{\mathcal{S}} \in \mathcal{L}_{TC}(\mathcal{A})$. We show that here the converse holds, too. If $M_T^{\mathcal{S}} \in \mathcal{L}_{TC}(\mathcal{A})$ we can build a TC-MSC $M' = (E, \preceq, \lambda, \tau')$ such that $\mathrm{dom}(\tau') \subseteq \mathrm{dom}(\tau)$, $\tau'(e, e') = \tau(e, e')$ for all $(e, e') \in \mathrm{dom}(\tau')$, and $\overline{M}' \in \mathcal{L}_u(\mathcal{A})$. Now, $\mathcal{L}_u(\mathcal{A}) \subseteq \mathcal{L}_u(\mu_\prec^{\mathcal{S}})$ hence $\overline{M}'$ is maximally defined and we obtain $M' = M_T^{\mathcal{S}}$. To summarize, we have shown that $T \in \mathcal{L}_{time}(\varphi)$ iff $M_T^{\mathcal{S}} \in \mathcal{L}_{TC}(\mathcal{A})$, and we conclude with Lemma 9 that this is equivalent to $T \in \mathcal{L}_{time}(\mathcal{A})$. $\square$

To characterize EC-CFMs in terms of full TMSO, we need to define restrictions on the channel size. For an integer $B > 0$, a word $w \in Act^*$ is *B-bounded* if, for any $p, q \in Ag$ and any prefix u of w, the number of occurrences of $p!q$ in u exceeds that of $q?p$ by at most B. An MSC M is said to be *existentially B-bounded* ($\exists$-B-bounded) if it has some B-bounded linearization. A T-MSC $(E, \preceq, \lambda, t)$ is said to be *untimed-$\exists$-B-bounded* if $(E, \preceq, \lambda)$ is $\exists$-B-bounded. Note that, directly lifting the definition of bounds from MSCs to T-MSCs is not completely intuitive: there are untimed-$\exists$-1-bounded T-MSCs whose minimal channel capacity for a timed linearization exceeds 1.

Following the same lines as in the proof of Theorem 6 but using the equivalence result from [11], we can show the following theorem.

Theorem 10. *Let $B > 0$ and let L be a set of untimed-$\exists$-B-bounded T-MSCs. There is an EC-CFM $\mathcal{A}$ with $\mathcal{L}_{time}(\mathcal{A}) = L$ iff there is $\varphi \in \mathrm{TMSO}(\mathcal{R}_{\preceq})$ with $\mathcal{L}_{time}(\varphi) = L$. Both directions are effective.*

5 Deciding Emptiness of EC-CFMs

In this section, we investigate emptiness checking for EC-CFMs. While the problem is of course undecidable in its full generality, we give a partial solution to it.

Theorem 11. *The following problem is decidable:*
INPUT*: An EC-CFM $\mathcal{A}$ and an integer $B > 0$.*
QUESTION*: Is there $T \in \mathcal{L}_{time}(\mathcal{A})$ such that T has a B-bounded timed linearization?*

Here, a timed linearization of T is B-bounded if the channel size never exceeds B during its execution.

We fix an EC-CFM $\mathcal{A} = (C, (\mathcal{A}_p)_{p \in Ag}, F)$, with $\mathcal{A}_p = (Q_p, \rightarrow_p, \iota_p)$, and $B > 0$. From $\mathcal{A}$, we build a *(finite) timed automaton* that accepts a timed word $w \in (Act \times \mathbb{R}^{\geq 0})^*$ iff w is a B-bounded timed linearization of some T-MSC in $\mathcal{L}_{time}(\mathcal{A})$. As emptiness is decidable for finite timed automata [1], we have shown Theorem 11.

Let us first recall the basic notion of a timed automaton. For a set $\mathcal{Z}$ of *clocks*, the set $\mathrm{Form}(\mathcal{Z})$ of *clock formulas* over $\mathcal{Z}$ is given by the grammar $\varphi ::= \mathrm{true} \mid \mathrm{false} \mid x \sim c \mid x - y \sim c \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2$ where $x, y \in \mathcal{Z}$, $\sim \in \{<, \leq, >, \geq, =\}$, and c ranges over $\mathbb{R}^{\geq 0}$.

A *timed automaton (with ε-transitions)* over Σ is a tuple $\mathcal{B} = (Q, \mathcal{Z}, \delta, \iota, F)$ where Q is a set of *states*, $\mathcal{Z}$ is a set of *clocks*, $\iota \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *final states*, and $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \mathrm{Form}(\mathcal{Z}) \times 2^{\mathcal{Z}} \times Q$ is the transition relation. The definition of a run of $\mathcal{B}$ and its language $\mathcal{L}(\mathcal{B}) \subseteq (\Sigma \times \mathbb{R}^{\geq 0})^*$ are as usual.

To keep track of the clock constraints used in $\mathcal{A}$, we need to recover a partial order from a word. Firstly, the partial order of an MSC can be recovered from any of its linearizations. If w is a linearization of MSC M, then M is isomorphic to the unique MSC $(E, \preceq, \lambda)$ such that $E = \{u \in Act^* \mid u \neq \varepsilon$ and $w = uv$ for some $v\}$ (i.e., E is the set of nonempty prefixes of w), $\lambda(u\sigma) = \sigma$ for $u \in Act^*$ and $\sigma \in Act$, and $\preceq_p = \{(u, v) \in E \times E \mid u$ is a prefix of v and $\lambda(u), \lambda(v) \in Act_p\}$. Thus, we might consider the partial-order relation of M to be a relation over prefixes of a given linearization of M. We go further to describe deterministic finite automata (DFA) that actually run on words that are linearizations of an MSC and accept if the first and last letter of it are related under $\preceq$, $\mathrm{Prev}_{\sigma}^{M}$, or $\mathrm{Next}_{\sigma}^{M}$. More precisely, our finite automata will run on linearizations of MSCs with additional labelings in $\{0, \ldots, B-1\}$. We say that such an MSC $(E, \preceq, \lambda, \rho)$ (with $\rho : E \rightarrow \{0, \ldots, B-1\}$) is *$B$-well-stamped* if, for any $e \in E$, $\rho(e) = |{\downarrow}e \cap \lambda^{-1}(\lambda(e))| \mod B$.

Lemma 12. *There are DFA $\mathcal{C}^{\triangleleft} = (Q^{\triangleleft}, \delta^{\triangleleft}, s_0^{\triangleleft}, F^{\triangleleft})$ and $\mathcal{C}^{\triangleright} = (Q^{\triangleright}, \delta^{\triangleright}, s_0^{\triangleright}, F^{\triangleright})$ over $Act \times \{0, \ldots, B-1\}$ with $|Q^{\triangleleft}| = |Q^{\triangleright}| = B^{\mathcal{O}(|Ag|^2)}$ (for $B \geq 2$) such that, for any $w = (\sigma, m)w'(\tau, n) \in (Act \times \{0, \ldots, B-1\})^*$ and $u, v \in (Act \times \{0, \ldots, B-1\})^*$, the following holds: If uwv is a linearization of some B-well-stamped MSC M, then*

- $w \in L(\mathcal{C}^{\triangleleft})$ *iff* $(u(\sigma, m)w'(\tau, n)\ ,\ u(\sigma, m)) \in \mathrm{Prev}_{\sigma}^{M}$ *and*
- $w \in L(\mathcal{C}^{\triangleright})$ *iff* $(u(\sigma, m)\ ,\ u(\sigma, m)w'(\tau, n)) \in \mathrm{Next}_{\tau}^{M}$.

From now on, we suppose $\mathcal{C}^{\triangleleft} = (Q^{\triangleleft}, \delta^{\triangleleft}, s_0^{\triangleleft}, F^{\triangleleft})$ and $\mathcal{C}^{\triangleright} = (Q^{\triangleright}, \delta^{\triangleright}, s_0^{\triangleright}, F^{\triangleright})$ from the above lemma to be fixed. We moreover suppose that the *previous* automaton $\mathcal{C}^{\triangleleft}$ has a unique sink state $s_{sink}^{\triangleleft}$, from which there is no final state reachable anymore.

The Timed Automaton Let us describe a timed automaton $\mathcal{B}$ that simulates the EC-CFM $\mathcal{A}$. To simplify the presentation, we allow infinitely many clocks and infinitely many states, though on any run only finitely many states and clocks will be seen. Later, we will modify this automaton in order to get down to finitely many states and clocks.

We use $\mathrm{Ind} = Act \times \mathbb{N}$ as (an infinite) index set. A state of the timed automaton $\mathcal{B} = (Q_{\mathcal{B}}, \mathcal{Z}, \delta, \iota_{\mathcal{B}}, F_{\mathcal{B}})$ will be a tuple $\mathrm{st} = (\overline{s}, \chi, \eta, \xi^{\triangleleft}, \xi^{\triangleright}, \gamma^{\triangleright}, \gamma^{m})$ where

- $\overline{s} = (s_p)_{p \in Ag} \in \prod_{p \in Ag} Q_p$ is a tuple of local states,
- $\chi : Ag^2 \to C^{\leq B}$ describes the contents of the channels,
- $\eta : Act \to \{0, \ldots, B-1\}$ gives the number that should be assigned to the next occurrence of an action,
- $\xi^{\triangleleft} : \mathrm{Ind} \dashrightarrow Q^{\triangleleft}$ and $\xi^{\triangleright} : \mathrm{Ind} \dashrightarrow Q^{\triangleright}$ associate with "active" indices, states in the *previous* and *next* automata as given by Lemma 12,
- $\gamma^{\triangleright} : \mathrm{Ind} \dashrightarrow \mathrm{Int}(\mathcal{A})$ associates *next* constraints with active indices, and
- $\gamma^{m} : Ag^2 \times \{0, \ldots, B-1\} \dashrightarrow \mathrm{Int}(\mathcal{A})$ describes the guards attached to messages.

The initial state is $\iota_{\mathcal{B}} = ((\iota_p)_{p \in Ag}, \chi_0, \eta_0, \xi_0^{\triangleleft}, \xi_0^{\triangleright}, \gamma_0^{\triangleright}, \gamma_0^{m})$ where χ_0 and η_0 map any argument to the empty word and 0, resp., and the partial maps $\xi_0^{\triangleleft}$, $\xi_0^{\triangleright}$, $\gamma_0^{\triangleright}$, and γ_0^{m} are nowhere defined. We will use clocks from the (infinite) set $\mathcal{Z} = \{z_{\sigma,i}^{\triangleleft}, z_{\sigma,i}^{\triangleright} \mid (\sigma, i) \in \mathrm{Ind}\} \cup \{z_{p,q,i}^{m} \mid (p, q, i) \in Ag^2 \times \{0, \ldots, B-1\}\}$. Then, $\delta \subseteq Q_{\mathcal{B}} \times Act \times \mathrm{Form}(\mathcal{Z}) \times 2^{\mathcal{Z}} \times Q_{\mathcal{B}}$ contains $((\overline{s}, \chi, \eta, \xi^{\triangleleft}, \xi^{\triangleright}, \gamma^{\triangleright}, \gamma^{m}), \tau, \varphi, R, (\overline{s}', \chi', \eta', \xi'^{\triangleleft}, \xi'^{\triangleright}, \gamma'^{\triangleright}, \gamma'^{m}))$ if there is a local transition $(s_p, \tau, g, c, s_p') \in \to_p$ on process p such that

- $s_r' = s_r$ for all $r \in Ag \setminus \{p\}$.
- if $\tau = p!q$, then $\chi'(p, q) = c \cdot \chi(p, q)$ and $\chi'(r, s) = \chi(r, s)$ for $(r, s) \neq (p, q)$.
- if $\tau = p?q$, then $\chi(q, p) = \chi'(q, p) \cdot c$ and $\chi'(r, s) = \chi(r, s)$ for $(r, s) \neq (q, p)$.
- $\eta'(\tau) = (\eta(\tau) + 1) \bmod B$ and η, η' coincide on all other actions.
- The states of the *previous* automata are updated. We initialize a new copy starting on the current position in order to be able to determine which latter positions are related with the current one by Prev_{τ}^{T}. We also reset a corresponding new clock $z_{\tau,i}^{\triangleleft}$ (see below). Indeed, all existing copies of $\mathcal{C}^{\triangleleft}$ are updated except those that would reach the $s_{sink}^{\triangleleft}$ state which are released since they will not be needed anymore.

$$\xi'^{\triangleleft}(\sigma, i) = \begin{cases} \delta^{\triangleleft}(s_0^{\triangleleft}, (\tau, \eta(\tau))) & \text{if } \sigma = \tau \ \wedge\ i = \min(\mathbb{N} \setminus \mathrm{dom}(\xi^{\triangleleft}(\sigma))) \\ \delta^{\triangleleft}(\xi^{\triangleleft}(\sigma, i), (\tau, \eta(\tau))) & \text{if } (\sigma, i) \in \mathrm{dom}(\xi^{\triangleleft}) \ \wedge \\ & \qquad \delta^{\triangleleft}(\xi^{\triangleleft}(\sigma, i), (\tau, \eta(\tau))) \neq s_{sink}^{\triangleleft} \\ \text{undefined} & \text{otherwise,} \end{cases}$$

- The states of the *next* automata are updated similarly, starting a new copy of $\mathcal{C}^{\triangleright}$ for each action σ such that there is a Next_σ constraint on the local transition. We also reset corresponding new clocks $z^{\triangleright}_{\sigma,i}$ (see below).

$$\xi'^{\triangleright}(\sigma,i) = \begin{cases} \delta^{\triangleright}(s_0^{\triangleright},(\tau,\eta(\tau))) & \text{if } \mathrm{Next}_\sigma \in \mathrm{dom}(g) \ \wedge\ i = \min(\mathbb{N}\setminus \mathrm{dom}(\xi^{\triangleright}(\sigma))) \\ \delta^{\triangleright}(\xi^{\triangleright}(\sigma,i),(\tau,\eta(\tau))) & \text{if } (\sigma,i)\in \mathrm{dom}(\xi^{\triangleright}) \ \wedge\ (\sigma \neq \tau\ \vee \\ & \qquad\qquad \delta^{\triangleright}(\xi^{\triangleright}(\sigma,i),(\tau,\eta(\tau))) \notin F^{\triangleright}) \\ \text{undefined} & \text{otherwise.} \end{cases}$$

- The *next* guards are updated. Each guard generating a new copy of $\mathcal{C}^{\triangleright}$ is recorded with the same new index. Guards that were registered before and are matched by the current action are released. All other recorded guards are kept unchanged.

$$\gamma'^{\triangleright}(\sigma,i) = \begin{cases} g(\mathrm{Next}_\sigma) & \text{if } \mathrm{Next}_\sigma \in \mathrm{dom}(g) \ \wedge\ i = \min(\mathbb{N}\setminus \mathrm{dom}(\xi^{\triangleright}(\sigma))) \\ \text{undefined} & \text{if } \sigma = \tau \ \wedge\ \xi'^{\triangleright}(\tau,i) \in F^{\triangleright} \\ \gamma^{\triangleright}(\sigma,i) & \text{otherwise.} \end{cases}$$

- The guards attached to message constraints are updated similarly.

$$\gamma'^{m}(r,s,i) = \begin{cases} g(\mathrm{Msg}) & \text{if } \mathrm{Msg} \in \mathrm{dom}(g) \ \wedge\ \tau = r!s \ \wedge\ i = \eta(\tau) \\ \text{undefined} & \text{if } \tau = s?r \ \wedge\ i = \eta(\tau) \\ \gamma^{m}(r,s,i) & \text{otherwise.} \end{cases}$$

- The guard φ makes sure that all constraints that get *matched* at the current event are satisfied. E.g., if the local transition contains a Prev_σ constraint, then we have to check $z^{\triangleleft}_{\sigma,i} \in g(\mathrm{Prev}_\sigma)$ for the (unique) i such that $\xi'^{\triangleleft}(\sigma,i) \in F^{\triangleleft}$. If there is no such i then there is no σ in the past of the current event and the Prev_σ constraint of the local transition cannot be satisfied. In this case, we set φ to false.

$$\begin{aligned} \varphi = & \bigwedge_{\substack{(\sigma,i)\ \mid\ \mathrm{Prev}_\sigma \in \mathrm{dom}(g) \\ \text{and } \xi'^{\triangleleft}(\sigma,i)\in F^{\triangleleft}}} z^{\triangleleft}_{\sigma,i} \in g(\mathrm{Prev}_\sigma) \quad \wedge \bigwedge_{\substack{\sigma\ \mid\ \mathrm{Prev}_\sigma \in \mathrm{dom}(g) \\ \text{and } \{i \mid \xi'^{\triangleleft}(\sigma,i)\in F^{\triangleleft}\} = \emptyset}} \text{false} \\ & \wedge \bigwedge_{\substack{i \in \mathrm{dom}(\gamma^{\triangleright}(\tau))\ \mid \\ \xi'^{\triangleright}(\tau,i)\in F^{\triangleright}}} z^{\triangleright}_{\tau,i} \in \gamma^{\triangleright}(\tau,i) \quad \wedge \bigwedge_{\substack{(q,p,i)\in \mathrm{dom}(\gamma^{m})\ \mid \\ \tau = p?q,\ \eta(\tau)=i}} z^{m}_{q,p,i} \in \gamma^{m}(q,p,i) \end{aligned}$$

- All newly defined clocks have to be reset, so we set R to be the union of sets $\{z^{\triangleleft}_{\tau,i} \mid i = \min(\mathbb{N}\setminus \mathrm{dom}(\xi^{\triangleleft}(\tau)))\}$, $\{z^{m}_{p,q,i} \mid \tau = p!q \text{ and } i = \eta(\tau)\}$, and $\{z^{\triangleright}_{\sigma,i} \mid \mathrm{Next}_\sigma \in \mathrm{dom}(g) \text{ and } i = \min(\mathbb{N}\setminus \mathrm{dom}(\xi^{\triangleright}(\sigma)))\}$.

Finally, the set of accepting states $F_{\mathcal{B}}$ consists of all tuples $(\overline{s},\chi,\eta,\xi^{\triangleleft},\xi^{\triangleright},\gamma^{\triangleright},\gamma^{m})$ in $Q_{\mathcal{B}}$ such that $\overline{s} \in F$, $\chi = \chi_0$, and the partial maps $\gamma^{\triangleright}$ and γ^{m} are nowhere defined. This ensures that each registered guard has been checked. Indeed, a constraint registered in $\gamma^{\triangleright}$ or γ^{m} 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.

Theorem 13. *$\mathcal{B}$ accepts precisely the B-bounded timed linearizations of $\mathcal{L}_{time}(\mathcal{A})$.*

A Finite Version of $\mathcal{B}$ To get down to a finite timed automaton that is equivalent to $\mathcal{B}$, we have to bound the number of copies of the automata $\mathcal{C}^{\triangleleft}$ and $\mathcal{C}^{\triangleright}$ that are active along a run. We can show that the number of active copies of $\mathcal{C}^{\triangleleft}$ is already bounded:

Proposition 14. *Assume that $(\overline{s}, \chi, \eta, \xi^{\triangleleft}, \xi^{\triangleright}, \gamma^{\triangleright}, \gamma^m)$ is a reachable state of $\mathcal{B}$. Then, $\mathrm{dom}(\xi^{\triangleleft}) \subseteq Act \times \{0, \ldots, |Q^{\triangleleft}|\}$.*

We deduce that, for the *previous* constraints, we can restrict to the *finite* index set $\mathrm{Ind}^{\triangleleft} = Act \times \{0, \ldots, |Q^{\triangleleft}|\}$: in a reachable state, $\xi^{\triangleleft}$ is a partial map from $\mathrm{Ind}^{\triangleleft}$ to $Q^{\triangleleft}$. This also implies that $\mathcal{B}$ uses finitely many *previous* clocks from $\{z^{\triangleleft}_{\sigma,i} \mid (\sigma, i) \in \mathrm{Ind}^{\triangleleft}\}$.

The remaining source of infinity comes from *next* constraints. The situation is not as easy as for *previous* constraints. The problem is that the number of registered Next_σ constraints, $|\mathrm{dom}(\gamma^{\triangleright})|$, may be unbounded. Assume that $(\sigma, i), (\sigma, j) \in \mathrm{dom}(\gamma^{\triangleright})$ for some $i \neq j$. Then, also $(\sigma, i), (\sigma, j) \in \mathrm{dom}(\xi^{\triangleright})$ and the clocks $z^{\triangleright}_{\sigma,i}$ and $z^{\triangleright}_{\sigma,j}$ have been reset. If we have $\xi^{\triangleright}(\sigma, i) = \xi^{\triangleright}(\sigma, j)$ 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 $z^{\triangleright}_{\sigma,i} \in \gamma^{\triangleright}(\sigma, i)$ and $z^{\triangleright}_{\sigma,j} \in \gamma^{\triangleright}(\sigma, j)$. The idea is to keep the stronger constraint and to release the other one. To determine the stronger constraint we have to deal separately with the upper parts and the lower parts of the constraints. An additional difficulty comes from the fact that the two clocks have not been reset simultaneously.

Let $x \sim c$ and $x' \sim' c'$ be two *upper*-guards which means that $\sim, \sim' \in \{<, \leq\}$. We say that $x \sim c$ is *stronger than* $x' \sim' c'$ if, when evaluated at the same instant, $x \sim c$ holds implies $x' \sim' c'$ holds as well. The stronger constraint can be determined with a diagonal guard: $x \sim c$ is stronger than $x' \sim' c'$ if either $x' - x < c' - c$ or else $x' - x \leq c' - c$ and ($\sim = <$ or $\sim' = \leq$). The relation *stronger than* is transitive and total among upper-guards. We can define similarly *stronger than* for *lower*-guards, i.e, when $\sim, \sim' \in \{>, \geq\}$. We have $x \sim c$ stronger than $x' \sim' c'$ if either $x' - x > c' - c$ or else $x' - x \geq c' - c$ and ($\sim = >$ or $\sim' = \geq$).

Now, we get back to our problem and show how to change $\mathcal{B}$ so that the size of $\mathrm{dom}(\xi^{\triangleright})$ in a state $\mathrm{st} = (\overline{s}, \chi, \eta, \xi^{\triangleleft}, \xi^{\triangleright}, \gamma^{\triangleright}, \gamma^m)$ can be bounded by $|Act| \cdot (2|Q^{\triangleright}| + 1)$. Note that $\mathrm{dom}(\gamma^{\triangleright}) = \mathrm{dom}(\xi^{\triangleright})$. A transition of $\mathcal{B}$ may initiate at most $|Act|$ new copies of $\mathcal{C}^{\triangleright}$ (one for each $\sigma \in Act$ such that $\mathrm{Next}_\sigma \in \mathrm{dom}(g)$. Hence, we say that state st is *safe* if for all $\sigma \in Act$ we have $|\mathrm{dom}(\xi^{\triangleright}(\sigma))| \leq 2|Q^{\triangleright}|$. The transitions of $\mathcal{B}$ are kept in the new automaton $\mathcal{B}'$ only when they start in a safe state.

If st is not safe, then $|\{i \mid \xi^{\triangleright}(\sigma, i) = q\}| > 2$ for some $\sigma \in Act$ and $q \in Q^{\triangleright}$. In this case, we say that st is unsafe for (σ, q) and let $\mathrm{Active}(\sigma, q) = \{i \mid \xi^{\triangleright}(\sigma, i) = q\}$.

If $\mathrm{Active}(\sigma, q) \neq \emptyset$, let $i_u \in \mathrm{Active}(\sigma, q)$ be such that the upper-guard defined by $z^{\triangleright}_{\sigma,i_u} \in \gamma^{\triangleright}(\sigma, i_u)$ is stronger than all upper-guards defined by $z^{\triangleright}_{\sigma,j} \in \gamma^{\triangleright}(\sigma, j)$ for $j \in \mathrm{Active}(\sigma, q)$. Further, let $i_\ell \in \mathrm{Active}(\sigma, q)$ be defined similarly for lower-guards.

From the definition of the relation *stronger than* we know that all constraints $z^{\triangleright}_{\sigma,j} \in \gamma^{\triangleright}(\sigma, j)$ for $j \in \mathrm{Active}(\sigma, q)$ are subsumed by the conjunction of $z^{\triangleright}_{\sigma,i_\ell} \in \gamma^{\triangleright}(\sigma, i_\ell)$ and $z^{\triangleright}_{\sigma,i_u} \in \gamma^{\triangleright}(\sigma, i_u)$. Therefore, we can release all *next* constraints associated with (σ, j) with $j \in \mathrm{Active}(\sigma, q) \setminus \{i_\ell, i_u\}$.

To do this, we add to $\mathcal{B}'$ an ε-transition $(\mathrm{st}, \varphi(\sigma, q, i_\ell, i_u), \varepsilon, \emptyset, \mathrm{st}')$. The guard should evaluate to true if i_ℓ and i_u determine stronger lower- and upper-constraints among those defined by $\mathrm{Active}(\sigma, q)$. Since the relation *stronger than* can be expressed with diagonal constraints, we have $\varphi(\sigma, q, i_\ell, i_u) \in \mathrm{Form}(\mathcal{Z})$. We have that, in state $\mathrm{st}' = (\overline{s}, \chi, \eta, \xi^{\lhd}, \xi'^{\rhd}, \gamma'^{\rhd}, \gamma^m)$, only the *next* information is changed:

$$\gamma'^{\rhd}(\tau, i) = \begin{cases} \text{undefined} & \text{if } \tau = \sigma \text{ and } i \in \mathrm{Active}(\sigma, q) \setminus \{i_\ell, i_u\} \\ \gamma^{\rhd}(\tau, i) & \text{otherwise} \end{cases}$$

$$\xi'^{\rhd}(\tau, i) = \begin{cases} \text{undefined} & \text{if } \tau = \sigma \text{ and } i \in \mathrm{Active}(\sigma, q) \setminus \{i_\ell, i_u\} \\ \xi^{\rhd}(\tau, i) & \text{otherwise.} \end{cases}$$

Then, $\{i \mid \xi'^{\rhd}(\sigma, i) = q\} = \{i_\ell, i_u\}$ and st' is safe for (σ, q).

We deduce that in the automaton $\mathcal{B}'$, we can restrict to the *finite* index set $\mathrm{Ind}^{\rhd} = Act \times \{0, \ldots, 2|Q^{\rhd}|\}$ for the partial maps $\xi^{\rhd}$ and $\gamma^{\rhd}$ used for the *next* constraints. Consequently, $\mathcal{B}'$ uses finitely many *next* clocks from $\{z^{\rhd}_{\sigma,i} \mid (\sigma, i) \in \mathrm{Ind}^{\rhd}\}$. The following proves Theorem 11, from which we deduce a decidability result for our logic.

Theorem 15. *The timed automaton $\mathcal{B}'$ is finite. It has $B^{\mathcal{O}(|Ag|^2)}$ many clocks (for $B \geq 2$), and we have $\mathcal{L}(\mathcal{B}') = \mathcal{L}(\mathcal{B})$.*

Corollary 16. *The following problem is decidable:*
INPUT*: $\varphi \in \mathrm{TMSO}(\mathcal{R}_{\preceq})$ and an integer $B > 0$.*
QUESTION*: Is there $T \in \mathcal{L}_{time}(\varphi)$ such that T has a B-bounded timed linearization?*

Acknowledgment We thank Martin Leucker for motivating discussions.

References

1. R. Alur and D. L. Dill. A theory of timed automata. *TCS*, 126(2):183–235, 1994.
2. R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: A determinizable class of timed automata. *TCS*, 211(1-2):253–273, 1999.
3. R. Alur, G. Holzmann, and D. Peled. An analyser for message sequence charts. In *TACAS 1996*, pages 35–48, 1996.
4. H. Ben-Abdallah and S. Leue. Timing constraints in message sequence chart specifications. In *Proc. of FORTE 1997*, pages 91–106, 1997.
5. B. Bollig and M. Leucker. Message-passing automata are expressively equivalent to EMSO logic. *TCS*, 358(2-3):150–172, 2006.
6. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2), 1983.
7. J. Büchi. Weak second order logic and finite automata. *Z. Math. Logik, Grundlag. Math.*, 5:66–62, 1960.
8. P. Chandrasekaran and M. Mukund. Matching scenarios with timing constraints. In *FORMATS 2006*, pages 91–106, 2006.
9. D. D'Souza. A logical characterisation of event clock automata. *International Journal of Foundations of Computer Science*, 14(4):625–640, 2003.
10. D. D'Souza and P. S. Thiagarajan. Product interval automata: A subclass of timed automata. In *FSTTCS 1999*, pages 60–71. Springer-Verlag, 1999.
11. B. Genest, D. Kuske, and A. Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *IC*, 204(6):920–956, 2006.
12. J. G. Henriksen, M. Mukund, K. N. Kumar, M. Sohoni, and P. S. Thiagarajan. A theory of regular MSC languages. *IC*, 202(1):1–38, 2005.
13. P. Krcal and W. Yi. Communicating timed automata: The more synchronous, the more difficult to verify. In *CAV 2006*, volume 4144 of *LNCS*, pages 243–257. Springer, 2006.