CONCUR 2007, 18th International Conference on Concurrency Theory
Proceedings: Luis Caires, Vasco T. Vasconcelos (eds.)
Springer Lecture Notes in Computer Science (2007), *to appear*

Checking Coverage for Infinite Collections of Timed Scenarios*

S. Akshay[1,3], Madhavan Mukund[2], and K. Narayan Kumar[2]

[1] LSV, ENS Cachan, France
akshay@lsv.ens-cachan.fr
[2] Chennai Mathematical Institute, Chennai, India
{madhavan,kumar}@cmi.ac.in
[3] Institute of Mathematical Sciences, Chennai, India

Abstract. We consider message sequence charts enriched with timing constraints between pairs of events. As in the untimed setting, an infinite family of time-constrained message sequence charts (TC-MSCs) is generated using an HMSC—a finite-state automaton whose nodes are labelled by TC-MSCs. A timed MSC is an MSC in which each event is assigned an explicit time-stamp. A timed MSC *covers* a TC-MSC if it satisfies all the time constraints of the TC-MSC. A natural recognizer for timed MSCs is a message-passing automaton (MPA) augmented with clocks. The question we address is the following: given a timed system specified as a time-constrained HMSC H and an implementation in the form of a timed MPA $\mathcal{A}$, is every TC-MSC generated by H covered by some timed MSC recognized by $\mathcal{A}$? We give a complete solution for locally synchronized time-constrained HMSCs, whose underlying behaviour is always regular. We also describe a restricted solution for the general case.

1 Introduction

In a distributed system, several agents interact with each other to generate a global behaviour. The interaction between these agents is usually specified in terms of scenarios, using message sequence charts (MSCs) [7].

We consider scenarios extended with timing constraints, called time-constrained MSCs (TC-MSCs). In a TC-MSC, we associate lower and upper bounds on the time interval between certain pairs of events. TC-MSCs are a natural and useful extension of the untimed notation for scenarios, because protocol specifications typically include timing requirements for message exchanges, as well as descriptions of how to recover from timeouts.

As an implementation model for timed distributed systems, we use communicating finite-state machines equipped with clocks, called timed message-passing automata (timed MPAs). Clock constraints are used to guard transitions and specify location invariants, as in other models of timed automata [3]. Just as the

* Partially supported by *Timed-DISCOVERI*, a project under the Indo-French Networking Programme.

runs of timed automata can be described in terms of timed words, the interactions exhibited by timed MPAs can be described using timed MSCs—MSCs in which each event is assigned an explicit timestamp.

Scenario specifications are typically incomplete and can be classified into two categories, positive and negative. Positive scenarios are those that the system should execute while negative scenarios indicate undesirable behaviours. This leads to the *scenario matching* problem: given a distributed system and a set of positive (negative) scenarios, does the system exhibit (avoid) these scenarios? In the untimed setting, efficient algorithms for the scenario matching problem have been identified in [10]. An automated approach is proposed in [5].

The timed analogue of scenario matching is *coverage*. A timed MSC T *covers* a TC-MSC specification M if the timestamps on the events in T satisfy the constraints specified in M. In general, a TC-MSC is covered by infinitely many timed MSCs. The coverage problem is to check whether the set of timed MSCs generated by a timed MPA cover all the TC-MSCs in the specification.

For finite sets of TC-MSCs, this problem reduces to the intersection of timed regular languages. In this case, checking coverage can be automated using the modelchecker Uppaal for timed systems, as described in [4].

In this paper, we consider the coverage problem for infinite collections of TC-MSCs. A standard way to generate an infinite set of MSCs is to use a High-level Message Sequence Chart (HMSC) [8]. In its most basic form, an HMSC is a finite directed graph, called a Message Sequence Graph (MSG), with each node labelled by an MSC. We label nodes in an MSGs with TC-MSCs and add time constraints across edges, resulting in a structure called a time-constrained MSG (TC-MSG). A TC-MSG defines a collection of TC-MSCs by concatenating the TC-MSCs labeling each path from an initial node to a terminal node.

Formally, coverage asks whether for every TC-MSC M generated by a TC-MSG, there is a timed MSC exhibited by the system that covers M. Since the set of TC-MSCs generated by a TC-MSG is infinite, it turns out that coverage can no longer be reduced to a simple intersection of timed regular languages.

We describe an algorithm to solve the coverage problem for *locally synchronized* TC-MSGs—those for which the underlying behaviour is guaranteed to be regular [6]. Our approach consists of "guiding" the timed MPA implementation to follow the TC-MSG specification in such a way that we can reduce the problem to *untimed* language inclusion. This allows us to solve the coverage problem both for positive and negative specifications. For arbitrary TC-MSGs, a fully general guided simulation can result in non-regular behaviours. However, we can adapt our approach to solve the coverage problem for TC-MSCs that are executed node by node in the underlying TC-MSG.

The paper is organized as follows. We begin with some preliminaries about MSCs and MSGs. In the next section, we describe how to attach timing information to scenario specifications. Section 4 formally describes timed message-passing automata. With this background, we formally describe the coverage problem in Section 5. Our solution is described in Section 6. We conclude with a brief discussion.

2 Preliminaries on MSCs

2.1 Message sequence charts

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a finite set of processes (agents) that communicate through messages via reliable FIFO channels using a finite set of message types $\mathcal{M}$. For $p \in \mathcal{P}$, let $Act_p = \{p!q(m), p?q(m) \mid p \neq q \in \mathcal{P}, m \in \mathcal{M}\}$ be the set of communication actions for 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 $Act = \bigcup_{p\in\mathcal{P}} Act_p$. We also denote the set of *channels* by $Ch = \{(p, q) \mid p \neq q\}$.

Labelled posets An Act-labelled poset is a structure $M = (E, \leq, \lambda)$ where $(E, \leq)$ is a poset and $\lambda : E \to Act$ is a labelling function.

For $e \in E$, let $\downarrow e = \{e' \mid e' \leq e\}$. For $X \subseteq E$, $\downarrow X = \cup_{e\in X} \downarrow e$. We call $X \subseteq E$ a *prefix* of M if $X = \downarrow X$. For $p \in \mathcal{P}$ and $a \in Act$, we set $E_p = \{e \mid \lambda(e) \in Act_p\}$ and $E_a = \{e \mid \lambda(e) = a\}$, respectively.

For each $(p, q) \in Ch$, we define a relation $<_{pq}$ as follows, to captures 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.

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

Finally, for each $p \in \mathcal{P}$, we define the relation $\leq_{pp} = (E_p \times E_p) \cap \leq$, with $<_{pp}$ standing for the largest irreflexive subset of $\leq_{pp}$.

Definition 1. *An MSC (over $\mathcal{P}$) is a* finite *Act-labelled poset $M = (E, \leq, \lambda)$ that satisfies the following conditions.*

1. *Each relation $\leq_{pp}$ is a linear order.*
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 $|{\downarrow}e \cap \left(\bigcup_{m\in\mathcal{M}} E_{p!q(m)}\right)| = |{\downarrow}e' \cap \left(\bigcup_{m\in\mathcal{M}} E_{q?p(m)}\right)|$.*
4. *The partial order $\leq$ is the reflexive, transitive closure of the relation $\bigcup_{p,q\in\mathcal{P}} <_{pq}$.*

The second condition ensures that every message sent along a channel is received. The third condition says that every channel is FIFO across all messages.

In diagrams, the events of an MSC are presented in *visual order*. The events of each process are arranged in a vertical line and messages are displayed as horizontal or downward-sloping directed edges. Fig. 1 shows an example with three processes $\{p, q, r\}$ and six events $\{e_1, e'_1, e_2, e'_2, e_3, e'_3\}$ corresponding to three messages—m_1 from p to q, m_2 from q to r and m_3 from p to r.

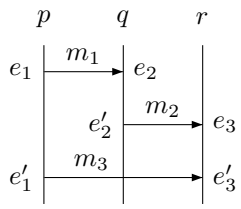

Fig. 1. An MSC

For an MSC $M = (E, \leq, \lambda)$, we let $\text{lin}(M) = \{\lambda(\pi) \mid \pi \text{ is a linearization of } (E, \leq)\}$. 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 of the MSC in Fig. 1.

MSC languages An *MSC language* is a set of MSCs.
We can also regard an MSC language $\mathcal{L}$ as a word language L over Act consisting of all linearizations of the MSCs in $\mathcal{L}$. For an MSC language $\mathcal{L}$, we set $\text{lin}(\mathcal{L}) = \bigcup\{\text{lin}(M) \mid M \in \mathcal{L}\}$.

Definition 2. *An MSC language $\mathcal{L}$ is said to be a* regular MSC language *if the word language* $\mathrm{lin}(\mathcal{L})$ *is a regular language over Act.*

Let M be an MSC and $B \in \mathbb{N}$. We say that $w \in \mathrm{lin}(M)$ is B-bounded if for every prefix v of w and for every channel $(p,q) \in Ch$, $\sum_{m\in\mathcal{M}} |\pi_{p!q(m)}(v)| - \sum_{m\in\mathcal{M}} |\pi_{q?p(m)}(v)| \leq B$, where $\pi_\Gamma(v)$ denotes the projection of v on $\Gamma \subseteq Act$. This means that along the execution of M described by w, no channel ever contains more than B-messages. We say that M is (universally) B-bounded if every $w \in \mathrm{lin}(M)$ is B-bounded. An MSC language $\mathcal{L}$ is B-bounded if every $M \in \mathcal{L}$ is B-bounded. Finally, $\mathcal{L}$ is bounded if it is B-bounded for some B.

We then have the following result [6].

Theorem 3. *If an MSC language $\mathcal{L}$ is regular then it is bounded.*

2.2 Message sequence graphs

Message sequence graphs (MSGs) are finite directed graphs with designated initial and terminal vertices. Each vertex in an MSG is labelled by an MSC. The edges represent (asynchronous) MSC concatenation, in which one MSC is "pasted" below the other. Formally, MSC concatenation is defined as follows.

Let $M_1 = (E^1, \leq^1, \lambda_1)$ and $M_2 = (E^2, \leq^2, \lambda_2)$ be a pair of MSCs such that E^1 and E^2 are disjoint. The *(asynchronous) concatenation* of M_1 and M_2 yields the MSC $M_1 \circ M_2 = (E, \leq, \lambda)$ where $E = E^1 \cup E^2$, $\lambda(e) = \lambda_i(e)$ if $e \in E^i$, $i \in \{1,2\}$, and $\leq = (\leq^1 \cup \leq^2 \cup \bigcup_{p\in\mathcal{P}} E^1_p \times E^2_p)^*$.

A *Message Sequence Graph* is a structure $G = (Q, \rightarrow, Q_{in}, Q_F, \Phi)$, where Q is a finite and nonempty set of states, $\rightarrow \subseteq Q \times Q$, $Q_{in} \subseteq Q$ is a set of initial states, $Q_F \subseteq Q$ is a set of final states and Φ labels each state with an MSC.

A *path* π through an MSG G is a sequence $q_0 \rightarrow q_1 \rightarrow \cdots \rightarrow q_n$ such that $(q_{i-1}, q_i) \in \rightarrow$ for $i \in \{1, 2, \ldots, n\}$. The MSC generated by π is $M(\pi) = M_0 \circ M_1 \circ M_2 \circ \cdots \circ M_n$, where $M_i = \Phi(q_i)$. A path $\pi = q_0 \rightarrow q_1 \rightarrow \cdots \rightarrow q_n$ is a *run* if $q_0 \in Q_{in}$ and $q_n \in Q_F$. The language of MSCs accepted by G is $L(G) = \{M(\pi) \mid \pi \text{ is a run through } G\}$. We say that an MSC language $\mathcal{L}$ is *MSG-definable* if there exists and MSG G such that $\mathcal{L} = L(G)$.

An example of an MSG is depicted in Fig. 2. The initial state is marked $\Rightarrow$ and the final state has a double circle. The language $\mathcal{L}$ defined by this MSG is *not* regular: $\mathcal{L}$ projected to $\{p!q(m), r!s(m)\}^*$ consists of $\sigma \in \{p!q(m), r!s(m)\}^*$ such that $|\pi_{p!q(m)}(\sigma)| = |\pi_{r!s(m)}(\sigma)| \geq 1$, which is not a regular string language.

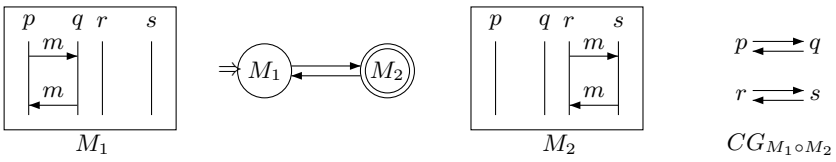

Fig. 2. A message sequence graph

In general, it is undecidable whether an MSG describes a regular MSC language [6]. However, a sufficient condition for the MSC language of an MSG to be regular is that the MSG be *locally synchronized*.

Communication graph For an MSC $M = (E, \leq, \lambda)$, let CG_M, *the communication graph of* M, be the directed graph $(\mathcal{P}, \mapsto)$ where:

- $\mathcal{P}$ is the set of processes of the system.
- $(p, q) \in \mapsto$ iff there exists an $e \in E$ with $\lambda(e) = p!q(m)$.

M is said to be *com-connected* if CG_M consists of one nontrivial strongly connected component and isolated vertices.

Locally synchronized MSGs The MSG G is *locally synchronized* [9] (or *bounded* [2]) if for every loop $\pi = q \to q_1 \to \cdots \to q_n \to q$, the MSC $M(\pi)$ is com-connected. In Fig. 2, $CG_{M_1 \circ M_2}$ is not com-connected, so the MSG is not locally synchronized. We have the following result for MSGs [2].

Theorem 4. *If G is locally synchronized, $L(G)$ is a regular MSC language.*

One of the factors contributing to the non-regularity of MSG-definable languages is that there is, in general, no bound on the asynchrony between processes. For instance, in Fig. 2, if we traverse the loop k times, we can identify a prefix of the MSC $\underbrace{M_1 \circ M_2 \circ \cdots \circ M_1 \circ M_2}_{k \text{ copies}}$ in which r and s are currently in the final copy of M_2 while p and q are in the first copy of M_1, at a distance $2k$. In locally synchronized MSGs, the gap between the first and last processes is always bounded. To formalize this, we define the active suffix of a path.

Active suffix Let G be an MSG and $\pi = q_0 q_1 \ldots q_k$ a path through G. Let X be a prefix of $M(q_0) \circ M(q_1) \circ \cdots \circ M(q_k)$. For each process p, let $i_p \in \{0, 1, \ldots, k\}$ be the node such that the maximum p-event in X lies in $M(q_{i_p})$. By convention, if X has no p-events, $i_p = 0$. Let $i_{\min} = \min_{p \in \mathcal{P}} i_p$. The *active suffix* of π is defined to be the path $q_{i_{\min}} q_{i_{\min}+1} \ldots q_k$.

The following two facts about locally synchronized MSGs will be useful. The first follows from Theorems 3 and 4. The second fact is the key to the proof of Theorem 4 (see [6], Appendix A).

Corollary 5. *Let G be a locally synchronized MSG. Then we can effectively compute bounds $B, K \in \mathbb{N}$ such that:*

- *Every MSC in $L(G)$ is B-bounded.*
- *For every MSC $M \in L(G)$, for every prefix X of M, the length of the active suffix of X is bounded by K.*

3 Adding time to scenarios

3.1 Time-constrained MSCs

A time-constrained MSC is an MSC annotated with time intervals between some pairs of events. For instance, consider the interaction between a user, an ATM

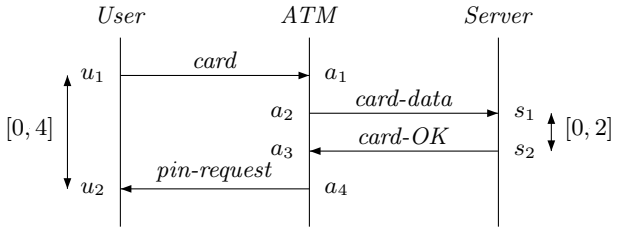

Fig. 3. A TC-MSC describing interaction with an ATM.

and a server depicted in Fig. 3. This MSC has eight events generated by four messages, with time constraints between the event pairs (u_1, u_2) and (s_1, s_2). The constraint $[0, 2]$ on (s_1, s_2) specifies that the server is expected to respond to a request to authenticate an ATM card within 2 time units. Similarly, the constraint $[0, 4]$ on (u_1, u_2) specifies that a user will be asked to enter his PIN within 4 time units of inserting the card.

For simplicity, we assume that time intervals are bounded by natural numbers. For $m, n \in \mathbb{N}$, $[m, n]$ is the closed interval $\{x \in \mathbb{R}_{\geq 0} \mid m \leq x \leq n\}$, while (m, n) is the open interval $\{x \in \mathbb{R}_{\geq 0} \mid m < x < n\}$. As usual, we permit half-open intervals of the form $[m, n)$ and $(m, n]$. To specify an interval without an upper bound we write $[m, \infty)$ or (m, ∞). Let $\mathcal{I}$ denote the set of intervals.

Definition 6. *Let $M = (E, \leq, \lambda)$ be an MSC. An* interval constraint *is a tuple $\langle (e_1, e_2), I \rangle$ where $e_1, e_2 \in E$ with $e_1 \leq_{pp} e_2$ for some $p \in \mathcal{P}$ or $e_1 <_{pq} e_2$ for some channel $(p, q) \in Ch$ and $I \in \mathcal{I}$.*

The restrictions on e_1 and e_2 ensure that an interval constraint is either local to a process or describes a bound on the delivery time of a single message.

Definition 7. *A* time-constrained MSC (TC-MSC) *is a pair $\mathcal{T} = (M, \mathcal{EC})$ where $M = (E, \leq, \lambda)$ is an MSC and $\mathcal{EC} \subseteq (E \times E) \times \mathcal{I}$ is a set of interval constraints such that each pair (e_1, e_2) is mapped to at most one interval.*

3.2 Timed MSCs

In a timed MSC, events are explicitly time-stamped so that the ordering on the time-stamps respects the partial order on the events.

Definition 8. *A* timed MSC *is pair (M, τ) where $M = (E, \leq, \lambda)$ is an MSC and $\tau : E \to \mathbb{R}_{\geq 0}$ assigns a nonnegative time-stamp to each event, such that for all $e_1, e_2 \in E$, if $e_1 \leq e_2$ then $\tau(e_1) \leq \tau(e_2)$.*

A timed MSC *covers* a TC-MSC if the time-stamps assigned to events respect the interval constraints specified in the TC-MSC. Let $r \in \mathbb{R}_{\geq 0}$ and $I \in \mathcal{I}$. We write $r \models I$ to denote that r lies in the interval specified by I.

Definition 9. *Let $M = (E, \leq, \lambda)$ be an MSC, $\mathcal{T} = (M, \mathcal{EC})$ a TC-MSC and $M_\tau = (M, \tau)$ a timed MSC. M_τ is said to* cover *$\mathcal{T}$ if for each $\langle (e_1, e_2), I \rangle \in \mathcal{EC}, \tau(e_2) - \tau(e_1) \models I$.*

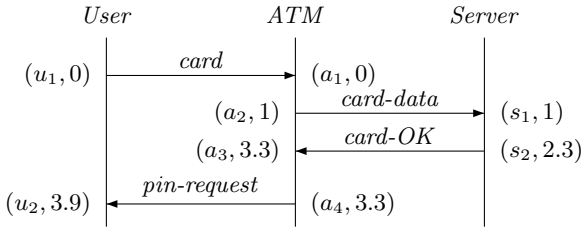

Fig. 4. A timed MSC describing interaction with an ATM.

Fig. 4 shows a timed MSC that covers the TC-MSC in Fig. 3.

Let $M_\tau = (M, \tau)$ be a timed MSC, where $M = (E, \leq, \lambda)$. A *timed linearization* of M_τ is a sequence $(e_0, \tau(e_0))(e_1, \tau(e_1)) \cdots (e_n, \tau(e_n))$ where $e_0 e_1 \ldots e_m$ is a linearization of $(E, \leq)$ and $\tau(e_0) \leq \tau(e_1) \leq \cdots \leq \tau(e_n)$. As is the case with untimed MSCs, under the FIFO assumption for channels, a timed MSC can be faithfully reconstructed from any one of its timed linearizations.

3.3 Time-constrained MSGs

A natural way to describe infinite families of TC-MSCs is to label the nodes of an MSG with TC-MSCs instead of normal MSCs. In addition, we permit local (process-wise) timing constraints along the edges of the MSG. A constraint for process p along an edge $q \to q'$ specifies a constraint between the final p-event of $M(q)$ and the initial p-event of $M(q')$, provided p actively participates in both these nodes. If p does not participate in either of these nodes, the constraint is ignored. We can think of each node in a TC-MSG as describing one phase of a communication protocol, with timing constraints along the edges specifying how long each process can wait between phases.

Definition 10. *A* time-constrained MSG (TC-MSG) *is a structure* $G = (Q, \to, Q_{in}, Q_F, \Phi, EdgeC)$*, where*

- Q *is a finite non-empty set of states with sets of initial and final states* Q_{in} *and* Q_F*, respectively, and* $\to \subseteq Q \times Q$ *is a transition relation, as in an MSG.*
- Φ *labels each node with a TC-MSC.*
- $EdgeC \subseteq Q \times Q \times \mathcal{P} \times \mathcal{I}$ *describes local constraints on the edges, with the restriction that* $(q, q', p, I) \in EdgeC$ *only if* $q \to q'$ *and each triple* (q, q', p) *is mapped to at most one interval.*

The definitions of *paths* and *runs* are the same for TC-MSGs as for MSGs. If $\pi = q_0 q_1 \ldots q_n$ is a path through G, the TC-MSC generated by π is denoted $M(\pi)$. To define $M(\pi)$, we begin with the TC-MSC $M_0 \circ M_1 \circ \cdots \circ M_n$, where $M_i = \Phi(q_i)$ for $i \in \{0, 1, \ldots, n\}$. For each edge $q_i \to q_{i+1}$, $0 \leq i < n$, if $(q_i, q_{i+1}, p, I) \in EdgeC$ we add a constraint I between the last p-event in M_i and the first p-event in M_{i+1}, provided p participates in both M_i and M_{i+1}.

The language $L(G)$ of TC-MSCs accepted by G is defined to be the set of TC-MSCs generated by the runs of G.

We define com-connectedness for TC-MSCs based on the communication graph just as we do for untimed MSCs. From this, we can define locally synchronized TC-MSGs in the same way as we do for MSGs.

4 Timed Message-Passing Automata

Message-passing automata are a natural machine model for recognizing MSCs. We extend the definition used in [6] to include clocks.

Definition 11. *Let $\mathcal{C}$ denote a finite-set of real-valued variables called* clocks. *A* clock constraint *is a conjunctive formula of the form $x \sim n$ or $x - y \sim n$ for $x, y \in \mathcal{C}$, $n \in \mathbb{N}$ and $\sim \in \{\leq, <, =, >, \geq\}$. Let $Form(\mathcal{C})$ denote the set of clock constraints over the set of clocks $\mathcal{C}$.*

Clock constraints will be used as guards in timed message-passing automata.

Definition 12. *A* clock assignment *for a set of clocks $\mathcal{C}$ is a function $v : \mathcal{C} \to \mathbb{R}_{\geq 0}$ that assigns a nonnegative real value to each clock in $\mathcal{C}$.*

A clock assignment v satisfies *a clock constraint φ, denoted $v \models \varphi$, if φ evaluates to true when we replace each clock x in φ by the value $v(x)$.*

Let $v : \mathcal{C} \to \mathbb{R}_{\geq 0}$ be a clock assignment. For $d \in \mathbb{R}_{\geq 0}$, $v + d$ denotes the clock assignment that maps each $x \in \mathcal{C}$ to $v(x) + d$. For $X \subseteq \mathcal{C}$, $v[X \leftarrow 0]$ is the clock assignment that agrees with v for $x \in \mathcal{C} \setminus X$ and maps all clocks in X to 0.

Definition 13. *A* timed message-passing automaton (timed MPA) *over Act with a set of clocks $\mathcal{C}$ is a structure $\mathcal{A} = (\{\mathcal{A}_p\}_{p \in \mathcal{P}}, Act, \mathcal{C})$. Each component $\mathcal{A}_p$ is of the form $(S_p, S_{in}^p, \to_p)$, where:*

- *S_p is a finite set of p-local states.*
- *$S_{in}^p \subseteq S_p$, is a set of initial states for p.*
- *$\to_p \subseteq S_p \times Form(\mathcal{C}) \times Act_p \times 2^{\mathcal{C}} \times S_p$ is the p-local transition relation.*

The local transition relation $\to_p$ specifies how the process p sends and receives messages. The transition $(s, \varphi, p!q(m), X, s')$ says that in state s, p can send the message m to q and move to state s'. This transition is *guarded* by the clock constraint φ—the transition is enabled only when the current values of all the clocks satisfy φ. The set X specifies the clocks whose values are reset to 0 when this transition is taken. Similarly, the transition $(s, \varphi, p?q(m), X, s')$ signifies that at state s, p can receive the message m from q and move to state s' provided the current clock values satisfy φ. Once again, all clocks in X are reset to 0.

To simplify the presentation, we have not included location invariants in our model. Location invariants are clock constraints attached to states that constrain the duration for which the automaton can remain in each state. Our results extend smoothly to timed MPAs equipped with location invariants.

Like timed automata, timed MPA can perform two types of moves: moves where the automaton does not change state and time elapses, and moves where some local component p changes state instantaneously as permitted by $\to_p$.

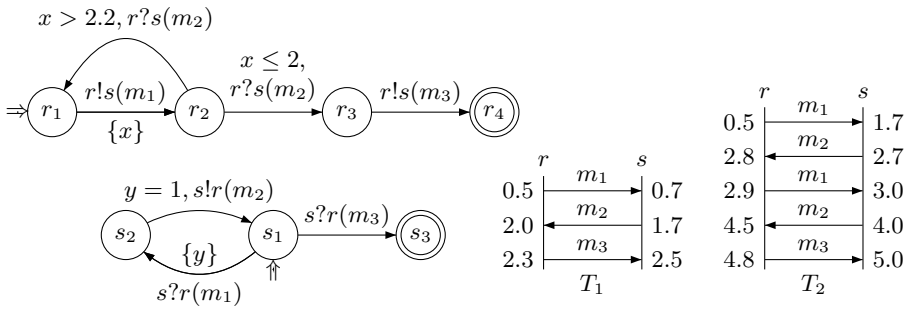

Fig. 5. A timed MPA and some timed MSCs that it recognizes

A global state of $\mathcal{A}$ is an element of $\prod_{p\in\mathcal{P}} S_p$. For a global state $\overline{s}$, $\overline{s}_p$ denotes the pth component of $\overline{s}$. A *configuration* is a triple $(\overline{s}, \chi, v)$ where $\overline{s}$ is a global state, $\chi : Ch \to \mathcal{M}^*$ is the *channel state* describing the message queue in each channel c and $v : \mathcal{C} \to \mathbb{R}_{\geq 0}$ is a clock assignment. An *initial configuration* of $\mathcal{A}$ is of the form $(\overline{s}_{in}, \chi_\varepsilon, v_0)$ where $\overline{s}_{in} \in \prod_{p\in\mathcal{P}} S^p_{in}$, $\chi_\varepsilon(c)$ is the empty string ε for every channel c and $v_0(x) = 0$ for every $x \in \mathcal{C}$.

The set of reachable configurations of $\mathcal{A}$, $Conf_\mathcal{A}$, is defined inductively, together with a transition relation $\Longrightarrow \subseteq Conf_\mathcal{A} \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times Conf_\mathcal{A}$.

- Every initial configuration $(\overline{s}_{in}, \chi_\varepsilon, v_0)$ is in $Conf_\mathcal{A}$.
- If $(\overline{s}, \chi, v) \in Conf_\mathcal{A}$ and $d \in \mathbb{R}_{\geq 0}$, then there is a global move $(\overline{s}, \chi, v) \stackrel{d}{\Longrightarrow} (\overline{s}, \chi, v + d)$ and $(\overline{s}, \chi, v + d) \in Conf_\mathcal{A}$.
- If $(\overline{s}, \chi, v) \in Conf_\mathcal{A}$ and $(\overline{s}_p, \varphi, p!q(m), X, \overline{s}'_p) \in \to_p$ such that v satisfies φ, there is a global move $(\overline{s}, \chi, v) \stackrel{p!q(m)}{\Longrightarrow} (\overline{s}', \chi', v[X \leftarrow 0])$ with $(\overline{s}', \chi', v[X \leftarrow 0]) \in Conf_\mathcal{A}$, where, for $r \neq p$, $\overline{s}_r = \overline{s}'_r$, $\chi'((p,q)) = \chi((p,q)) \cdot m$, and for $c \neq (p,q)$, $\chi'(c) = \chi(c)$.
- Similarly, if $(\overline{s}, \chi, v) \in Conf_\mathcal{A}$ and $(\overline{s}_p, \varphi, p?q(m), X, \overline{s}'_p) \in \to_p$ such that v satisfies φ, there is a global move $(\overline{s}, \chi, v) \stackrel{p?q(m)}{\Longrightarrow} (\overline{s}', \chi', v[X \leftarrow 0])$ with $(\overline{s}', \chi', v[X \leftarrow 0]) \in Conf_\mathcal{A}$, where, for $r \neq p$, $\overline{s}_r = \overline{s}'_r$, $\chi((q,p)) = m \cdot \chi'((q,p))$, and for $c \neq (q,p)$, $\chi'(c) = \chi(c)$.

Let $\mathrm{prf}(\sigma)$ denote the set of prefixes of a timed word $\sigma = (a_1, t_1)(a_2, t_2)\ldots(a_k, t_k) \in (Act \times \mathbb{R}_{\geq 0})^*$. A run of $\mathcal{A}$ over σ is a map $\rho : \mathrm{prf}(\sigma) \to Conf_\mathcal{A}$ such that $\rho(\varepsilon)$ is assigned an initial configuration $(\overline{s}_{in}, \chi_\varepsilon, v_0)$ and for each $\sigma' \cdot (a_i, t_i) \in \mathrm{prf}(\sigma)$, $\rho(\sigma') \stackrel{d_i}{\Longrightarrow}\stackrel{a_i}{\Longrightarrow} \rho(\sigma' \cdot (a_i, t_i))$ with $t_i = t_{i-1} + d_i$ and t_0 implicitly set to 0.

The run ρ is *complete* if $\rho(\sigma) = (s, \chi_\varepsilon, v)$ is a configuration in which all channels are empty. When a run on σ is complete, σ is a timed linearization of a timed MSC. We define $L(\mathcal{A}) = \{\sigma \mid \mathcal{A} \text{ has a complete run over } \sigma\}$. Thus, $L(\mathcal{A})$ corresponds to the set of timed linearizations of a collection of timed MSCs. Let $\mathcal{L}(\mathcal{A})$ be the language of timed MSCs corresponding to $L(\mathcal{A})$.

Fig. 5 shows a timed MPA along with two of the timed MSCs that it recognizes. In the timed MSCs, we have only written the time-stamps associated with

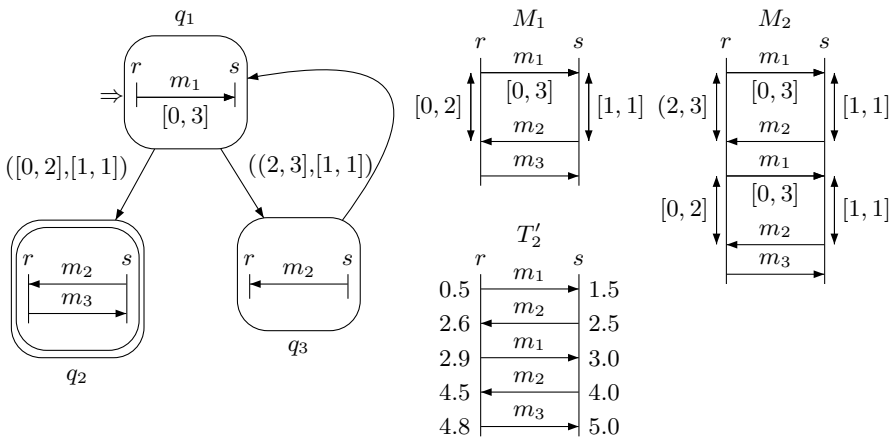

Fig. 6. The coverage problem

the events and not the event names themselves. In this timed MPA, r sends a message m_1 to s. Process s replies with m_2 exactly 1 time unit after it receives m_1. If m_2 is received by r within 2 time units of its sending m_1, it sends m_3 and quits. Otherwise, if at least 2.2 time units go by before r receives m_2, it resends m_1. Note that there is no transition enabled in r for the interval $2 < x \leq 2.2$.

5 The coverage problem

We are interested in the following verification problem for timed scenario specifications. Let G be a TC-MSG and $\mathcal{A}$ a timed MPA. The *coverage problem* for G and $\mathcal{A}$ is to determine whether for each TC-MSC $M \in L(G)$, there is a timed MSC $T \in \mathcal{L}(\mathcal{A})$ such that T covers M. This is a natural verification problem when we interpret TC-MSGs as incomplete positive specifications.

For instance, consider the TC-MSG G in Fig. 6. In G, r sends a message m_1 to s that could be delayed by upto 3 time units, to which s replies after exactly 1 time unit. If the response m_2 from s reaches r within 2 time units from the time r sent m_1, r sends a final message m_3 and quits. Otherwise, r resends m_1. M_1 and M_2 (Fig. 6) are TC-MSCs in $L(G)$, generated by paths q_1q_2 and $q_1q_3q_1q_2$, respectively. These are covered by the timed MSCs T_1 and T_2 (Fig. 5) in $\mathcal{L}(\mathcal{A})$.

In the untimed case, scenario matching asks whether $L(G) \subseteq \mathcal{L}(\mathcal{A})$, where G is an MSG and $\mathcal{A}$ is an MPA. In the timed case, we cannot reduce coverage to language inclusion of timed MSCs. A TC-MSC M represents an infinite family of timed MSCs, each of which covers M. However, the implementation need not, in general, permit all these realizations. For instance, the timed MPA in Fig. 5 will not exhibit any timed MSC covering M_2 from Fig. 6 where the time difference between the first two p-events is 2.1, such as in the timed MSC T_2' in Fig. 6.

Another plausible approach is to treat this as a timed game between Spoiler, who picks a path in the TC-MSG G, and Duplicator, who picks a timed MSC in

$\mathcal{L}(\mathcal{A})$ that covers the TC-MSC generated along the path chosen by Spoiler. At each step, Spoiler adds a node to the path in G. Duplicator has to match this move by extending the current timed MSC so that it stays in $\mathcal{L}(\mathcal{A})$ and covers the TC-MSC described by the extended path. However, a winning strategy in this game would have the following property: if two paths π_1 and π_2 have a common prefix π, then the timed MSC generated by Duplicator for the prefix π must be the same for the plays in which Spoiler generates π_1 and π_2. Notice that the paths that generate M_1 and M_2 in Fig. 6 share a common prefix, namely q_1. In any timed MSC that covers M_1, message m_1 must be delivered within 1 time unit whereas in any timed MSC that covers M_2, m_1 can only reach after 1 time unit. Hence, any timed MSC that covers $M(q_1)$ and can be extended to cover M_1 cannot simultaneously be extended to cover M_2. In other words, the game-theoretic formulation introduces too strict a correlation between the timed MSCs covering different paths through the TC-MSG.

These observations suggest that traditional approaches for scenario matching in the untimed case do not generalize to the coverage problem in the timed case. In the next section, we formulate a solution to the coverage problem for locally synchronized TC-MSGs.

6 Coverage for locally synchronized TC-MSGs

Theorem 14. *Let G be a locally synchronized TC-MSG and $\mathcal{A}$ a timed MPA. The coverage problem for G and $\mathcal{A}$ is decidable.*

Proof. Our proof strategy is as follows. From $\mathcal{A}$, we construct a timed automaton $\mathcal{B}$ that simulates the global behaviour of $\mathcal{A}$, restricted to runs that are consistent with paths through $L(G)$. In addition to the communication actions in Act, the timed automaton $\mathcal{B}$ also has moves labelled by nodes from G, indicating the path that it is following. As usual, we can use the region construction [1] to obtain an untimed regular language $\mathit{Untime}(L(\mathcal{B})) \subseteq (Act \cup Q)^*$. It will then turn out that $\mathit{Untime}(L(\mathcal{B}))$ projected onto Q precisely describes the set of paths through G that are covered by some timed MSC in $\mathcal{L}(\mathcal{A})$.

To check coverage, we just need to verify that the *node language* of G, $L_Q(G) = \{q_0 q_1 \dots q_n \in Q^* \mid q_0 \to q_1 \to \cdots \to q_n \text{ is a run}\}$, is included in $L_Q(\mathcal{B}) = \pi_Q(\mathit{Untime}(L(\mathcal{B})))$. This would imply that for every path in π through G, the TC-MSC $M(\pi)$ is covered by some timed MSC in $L(\mathcal{A})$.

There is, however, a complication. Some paths in G may define TC-MSCs that cannot be covered, because of self-contradictory timing constraints. These paths need not be covered by timed MSCs from $L(\mathcal{A})$. We cannot, therefore, directly compare $L_Q(G)$ with $L_Q(\mathcal{B})$. The solution is straightforward: we start with the trivial automaton $\mathcal{A}_U$ that recognizes Act^*, which can be regarded as a degenerate timed automaton with no timing constraints. To $\mathcal{A}_U$, we apply the same construction as we have done for $\mathcal{A}$. The resulting timed automaton $\mathcal{B}_U$ will mark out all paths π through G for which $M(\pi)$ can be covered by some timed MSC. We can now check whether $L_Q(\mathcal{B}_U) = \pi_Q(\mathit{Untime}(L(\mathcal{B}_U)))$ is included in $L_Q(\mathcal{B})$. Since both $L_Q(\mathcal{B}_U)$ and $L_Q(\mathcal{B})$ are regular languages, the result follows.

Constructing $\mathcal{B}$ Recall that $\mathcal{A} = (\{\mathcal{A}_p\}_{p\in\mathcal{P}}, Act, \mathcal{C})$, where each component $\mathcal{A}_p$ is of the form $(S_p, S^p_{in}, \rightarrow_p)$, as described in Section 4. The structure of the TC-MSG is given by $G = (Q, \rightarrow, Q_{in}, Q_F, \Phi, EdgeC)$, as described in Section 3. Without loss of generality, we assume that all the events that occur in the TC-MSCs labelling nodes of G have distinct names, so that when we refer to a constraint $\langle (e, e'), I\rangle$, there is no ambiguity.

Alphabet The alphabet of $\mathcal{B}$ is $Act \cup Q$, where Q is the set of nodes in G.

States A state of $\mathcal{B}$ consists of the following components:

- $\overline{s} \in \prod_{p\in\mathcal{P}} S_p$, a global state of $\mathcal{A}$.
- $\chi : Ch \rightarrow \mathcal{M}^*$, the state of the channels.
- $\eta : Act \rightarrow \{0, \ldots, B-1\}$, a function to count occurrences of each action in Act modulo B.
- $\pi \in Q^*$, (the active suffix of) a path in G.
- $\rho : E_{M(\pi)} \rightarrow \{0, 1, 2\}$, a labelling function for the events in the TC-MSC $M(\pi)$. (Events labelled 0 are yet to be executed. Events labelled 1 represent the "active" frontier of events that will be executed next along each process. Events labelled 2 are those that have already been executed.)

The state space of $\mathcal{B}$ is finite because of the bounds B and K that we can compute for a locally synchronized TC-MSG G (Corollary 5). Since every TC-MSC in $L(G)$ is B-bounded, the channel state χ is bounded. Moreover, since the length of the active suffix along any run is at most K, the length of π is bounded, and hence so is ρ.

An initial state of $\mathcal{B}$ is one in which the global state of the timed MPA $\mathcal{A}$ is $\overline{s}_{in} \in \prod_{p\in\mathcal{P}} S^p_{in}$, all channels are empty, η maps each action to 0, the active suffix $\pi = q$ for some initial node $q \in Q_{in}$ and ρ maps the minimal events along each process in $M(q)$ to 1 and all other events to 0.

Clocks of $\mathcal{B}$ Interval constraints in G will be monitored using additional clocks in $\mathcal{B}$. The set of clocks of $\mathcal{B}$ consists of all clocks of $\mathcal{A}$ along with a clock z^{lc} for each constraint lc local to a process in G, a clock z^{EC} for each edge constraint EC in G, and a set of clocks z^{mesg}_i, $1 \leq i \leq B$, one for each potentially active copy of a message constraint $mesg$ in G.

If $lc = \langle (e, e'), I\rangle$ is a local constraint on process p, the clock z^{lc} is reset to zero when $\mathcal{B}$ simulates e and is checked against the interval I when $\mathcal{B}$ simulates e'. Each clock z^{EC} is used to check edge constraints in an analogous manner. For message constraints, we may have multiple copies of the same message in a channel. We have to maintain and check the constraint independently for each copy of the message. However, since channels are B-bounded, there can be no more than B active copies of a message at any point. Thus, we can use the clocks z^{mesg}_i in conjunction with the state information η that assigns a number modulo B to each communication action in order to check message constraints.

Transitions We can now define the transition function for $\mathcal{B}$. Each move of $\mathcal{B}$ consists of one of the following:

- Pick a process p and perform a legal move in $\mathcal{A}$ that executes the (unique) p-event e_p labelled 1 by ρ. This move is labelled by $\lambda(e_p) \in Act$.
- If there is no active event for some process, extend the active suffix by adding a node q. This move is labelled q.

Transitions on Act Since $\mathcal{B}$ incorporates the global state, the channel state and the clocks of $\mathcal{A}$, it can faithfully simulate every move of $\mathcal{A}$. We use the remaining components of the state of $\mathcal{B}$ to restrict the moves of $\mathcal{A}$ to follow a path in G.

Formally, we have a move $(s, \varphi, \alpha, X, s')$ in $\mathcal{B}$, where φ is a clock constraint, $\alpha \in Act$ and X is a set of clocks to be reset if the following hold. Let us assume that $\alpha \in Act_r$ is an r-action. Then:

- The projection of $(s, \varphi, \alpha, X, s')$ onto components from $\mathcal{A}$ defines a valid (global) transition of $\mathcal{A}$.
- The action α must match the label of the r-event e_r currently labelled 1 by ρ in s.
- If $lc = \langle (e, e_r), I \rangle$ is a local constraint, we must have in φ a constraint checking that z^{lc} satisfies I. For instance, if $I = [m, n)$, we have a constraint $m \leq z^{lc} \wedge z^{lc} < n$ in φ.
- If $EC = \langle (e, e_r), I \rangle$ is an edge constraint, e is the maximum r-event in $M(q_i)$ and e_r is the minimum r-event in $M(q_{i+1})$ for some nodes q_i, q_{i+1} along π, then we must have in φ a constraint checking that z^{EC} satisfies I.
- If $\alpha = r?s(m)$, $\eta(\alpha) = k$ and there is a message constraint $mesg = \langle (e_s, e_r), I \rangle$ from the corresponding send event e_s, we must have in φ a constraint checking that $z^{mesg}_{(k+1) \bmod m}$ satisfies I.

The new state of $\mathcal{B}$, s', is obtained from s as follows.

- The global state of $\mathcal{A}$ is updated according to the transition simulated by $\mathcal{B}$.
- The channel state of $\mathcal{A}$ is updated in the obvious way.
- $\eta(\alpha)$ is updated to $(\eta(\alpha) + 1) \bmod B$.
- $\rho(e_r)$ is set to 2 and the next r event in $M(\pi)$, if any, is labelled 1.
- Let $\pi = q_0 q_1 \dots q_m$. Let $i_{\max}$ be the maximum index in the set $\{i \mid \forall e \in M(q_i).\ \rho(e) = 2\}$. Update π to $q_{i_{\max}} q_{i_{\max}+1} \dots q_m$. Note that this deletes all completed nodes from the active suffix except the last one. We need to retain $q_{i_{\max}}$ in the active suffix to check edge constraints.

Finally, we compute the set X of clocks to be reset on this transition as follows:

- If $lc = \langle (e_r, e'), I \rangle$ is a local constraint involving e_r, add z^{lc} to X.
- If e_r is the maximum r-event in $M(q)$, for each edge $q \rightarrow q' \in G$, if e' is the minimum r-event in $M(q')$ and $EC = \langle (e_r, e'), I \rangle$ is an edge constraint, add z^{EC} to X.
  Note that we activate clocks for edge constraints along *all* possible extensions of the path when we encounter a maximal event in a node. However, when we

reach a minimal event in the next node, we ensure that we only enforce the constraint for the edge that was actually traversed. Each time we traverse an edge with constraint EC, the clock z^{EC} will be reset, so there is no danger when we check an edge constraint that the clock value is stale.
- If $\alpha = r!s(m)$ and there is a message constraint $mesg = \langle (e_r, e_s), I \rangle$ involving e_r, add $z^{EC}_{\eta(\alpha)}$ to X.

Transitions on Q If $\mathcal{B}$ is in a state $s = (\overline{s}, \chi, \eta, \pi, \rho)$ where for some subset $P \subseteq \mathcal{P}$, for each $p \in P$ there is no p-event labelled 1 by ρ, we need to extend $\pi = q_0 q_1 \dots q_m$. In such a situation, for each q such that there is an edge $q_m \to q$ in G, we add a transition $s \xrightarrow{q} s'$ in $\mathcal{B}$ where $s' = (\overline{s}, \chi, \eta, \pi \cdot q, \rho')$, such that $\rho'(e)$ agrees with $\rho(e)$ for all $e \in M(\pi)$, $\rho'(e) = 1$ if e is a minimum p-event in $M(q)$ for $p \in P$, and $\rho'(e) = 0$ for all other events $e \in M(q)$.

Accepting states $\mathcal{B}$ A state $s = (\overline{s}, \chi, \eta, \pi, \rho)$ of $\mathcal{B}$ is accepting if $\overline{s}$ is an accepting state of $\mathcal{A}$, $\chi = \chi_\varepsilon$, the channel state in which there are no messages in any channel, $\pi = q_0 q_1 \dots q_m$ with $q_m \in Q_F$ and $\rho(e) = 2$ for all $e \in M(\pi)$.

From the definition of $\mathcal{B}$, it is routine, though tedious, to prove that $\mathcal{B}$ simulates precisely those runs of $\mathcal{A}$ that cover some path in G. Moreover, for each such run, $\mathcal{B}$ records the sequence of nodes in G traversed along the run. Thus $L_Q(\mathcal{B}) = \pi_Q(\mathit{Untime}(L(\mathcal{B})))$ is a regular language describing the set of paths in G covered by runs of $\mathcal{A}$. As described earlier, we can apply the same construction to the trivial automaton $\mathcal{A}_U$ recognizing Act^* to get a timed automaton $\mathcal{B}_U$ that marks out all feasible paths in G. It then follows that checking coverage is equivalent to checking that $L_Q(\mathcal{B}_U) \subseteq L_Q(\mathcal{B})$.

From locally synchronized to arbitrary TC-MSGs

If we drop the assumption that the TC-MSG we start with is locally synchronized, the automaton $\mathcal{B}$ fails to be finite-state because we cannot guarantee a bound on either the channel capacities or the length of the active suffix.

However, in such a situation, we can still solve a restricted version of the problem. For each MSC of the form $M = M_1 \circ M_2 \circ \dots \circ M_k \in L(G)$ generated by a path $q_1 q_2 \dots q_k$ with $M_i = M(q_i)$ for $i \in \{1, 2, \dots, k\}$, we ask whether there is a timed MSC $T \in \mathcal{L}(\mathcal{A})$ such that $\mathrm{lin}(T) = w_1 w_2 \dots w_k$ where $w_i \in \mathrm{lin}(M_i)$ for $i \in \{1, 2, \dots, n\}$. In other words, we restrict our attention to runs of $\mathcal{A}$ that cover G one node at a time.

We can suitably modify the construction of the automaton $\mathcal{B}$ to achieve this. In the new version of $\mathcal{B}$, there is always only one active node, from the way we have defined the simulation. Channel capacities are bounded by the maximum capacity exhibited by the individual TC-MSCs labelling the nodes of G. We can then obtain an analogous result for coverage in terms of $L_Q(\mathcal{B}_U)$ and $L_Q(\mathcal{B})$. Due to lack of space, we omit further details.

Negative specifications

For negative specifications, the verification problem is dual—given a TC-MSG G and a timed MPA $\mathcal{A}$, we want to ensure that there is no timed MSC $T \in \mathcal{L}(\mathcal{A})$ that covers any TC-MSC $M \in L(G)$. Let us call this problem *avoidance*.

Unlike coverage, avoidance does reduce directly a problem involving timed languages. We want $\mathrm{lin}(L(G))$, the set of timed linearizations of $L(G)$, to be disjoint from $L(\mathcal{A}) = \mathrm{lin}(\mathcal{L}(\mathcal{A}))$, the set of timed linearizations of $\mathcal{L}(\mathcal{A})$. However, there is still the problem of finding an effective presentation of $\mathrm{lin}(L(G))$.

Notice, however, that the construction used to prove Theorem 14 also solves the avoidance problem. Using the terminology introduced above, avoidance is equivalent to checking that $L_Q(\mathcal{B}_U) \cap L_Q(\mathcal{B}) = \emptyset$.

7 Discussion

We have shown how to solve the timed analogue of the scenario matching problem for specifications consisting of infinite sets of time-constrained MSCs. Though our result is stated in the context of timed MPAs, our construction works even if the system is presented as a global timed automaton. Our solution to the coverage problem for positive specifications also yields a solution to the avoidance problem for negative specifications. An interesting problem to be tackled is realizability for time-constrained MSGs—given a TC-MSG G, construct a timed MPA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A})$ covers $L(G)$.

References

1. R. Alur and D. Dill: A Theory of Timed Automata. *Theor. Comput. Sci.*, **126** (1994) 183–225.
2. R. Alur and M. Yannakakis: Model checking of message sequence charts. *Proc. CONCUR'99*, Springer Lecture Notes in Computer Science **1664**, (1999) 114–129
3. J. Bengtsson and Wang Yi: Timed Automata: Semantics, Algorithms and Tools, *Lectures on Concurrency and Petri Nets 2003*, LNCS **3098**, Springer-Verlag (2003) 87–124.
4. P. Chandrasekaran and M. Mukund: Matching Scenarios with Timing Constraints. *Proc. FORMATS 2006*, Springer LNCS 4202 (2006) 98–112.
5. D. D'Souza and M. Mukund: Checking consistency of SDL+MSC specifications, *Proc. SPIN Workshop 2003*, LNCS **2648**, Springer-Verlag (2003) 151–165.
6. J.G. Henriksen, M. Mukund, K. Narayan Kumar, M. Sohoni and P.S. Thiagarajan: A Theory of Regular MSC Languages. *Inf. Comp.*, **202(1)** (2005) 1–38.
7. ITU-T Recommendation Z.120: *Message Sequence Chart (MSC)*. ITU, Geneva (1999).
8. S. Mauw and M.A. Reniers: High-level message sequence charts. *Proc. SDL'97*, Elsevier (1997) 291–306.
9. A. Muscholl and D. Peled: Message sequence graphs and decision problems on Mazurkiewicz traces. *Proc. MFCS'99*, Springer Lecture Notes in Computer Science **1672**, (1999) 81–91
10. A. Muscholl, D. Peled, and Z. Su: Deciding properties for message sequence charts. *Proc. FOSSACS'98*, LNCS **1378**, Springer-Verlag (1998) 226–242.