Regular Set of Representatives for Time-Constrained MSC Graphs[☆]

S. Akshay[a], Blaise Genest[a,b], Loïc Hélouët[a], Shaofa Yang[c]

[a] *IRISA, ENS Cachan Bretagne - CNRS - INRIA, Rennes, France*
[b] *CNRS, UMI IPAL joint with NUS and A*STAR/I2R, Singapore*
[c] *SIAT, Chinese Academy of Sciences, China*

Abstract

Systems involving both time and concurrency are notoriously difficult to analyze. Existing decidability results apply when clocks on different processes cannot be compared or when the set of timed executions is regular. We prove new decidability results for timed concurrent systems, requiring neither restriction. We consider the formalism of time-constrained MSC-graphs (TC-MSC graphs for short) introduced in [1], and study if the set of timed executions generated by a TC-MSC graph is empty. This emptiness problem is known to be undecidable in general [9]. Our approach consists of finding a regular set R of *representative* timed executions, i.e., such that every timed execution of the system has an equivalent, up to commutation, timed execution in R. This allows us to solve the emptiness problem under the assumption that the TC-MSC graph is prohibited from (1) *forcing* any basic scenario to take an arbitrarily long time to complete and (2) *enforcing* unboundedly many events to occur within one unit of time.

Keywords: timed automata, partial order languages, MSC graphs, set of representatives

1. Introduction

In a distributed system, several processes interact to implement a collection of global behaviors. These processes are often equipped with timing information and protocol specifications include timing requirements for messages and descriptions of how to recover from timeouts. Thus, a system designer has to deal with situations where time and concurrency influence each other. One way to describe these interactions is through scenarios, formalized using Message Sequence Charts (MSCs) [12]. The timing information is captured by adding timing constraints between pairs of events, yielding time-constrained MSCs (TC-MSCs). Further, infinite collections of MSCs can be modelled using High-level Message Sequence Charts, or more basic forms called MSC-graphs: directed graphs whose nodes are labelled by MSCs. MSC-graphs can be generalized to time-constrained MSC graphs (TC-MSC graphs)[9], whose nodes are labelled by TC-MSCs and edges have additional timing constraints. In this paper, we focus on decidability results for the analysis of TC-MSC graphs.

Obtaining decidability in the presence of both time and concurrency is a challenging issue. Even the simple yet fundamental question of checking if there exists a timed execution of a TC-MSC graph consistent with all the constraints is undecidable in general [9]. This is the *emptiness problem*, which in the case of (sequential) timed automata is known to be decidable [3]. Extending such results to distributed systems has been done only in two particular and limited settings. In [13, 8], clocks are local to a process. But then, one cannot specify time taken by a communication (message or synchronisation). This limitation makes the specification formalism very weak. The second setting can relate clocks from different processes and specify how long a communication can take [1, 2, 6, 7]. However, they restrict the concurrency in a structural way, for instance by considering only locally synchronized ([15, 4, 11]) MSC-graphs (in [1, 2]) or only safe Petri Nets (in [6, 7]). This forces the set of timed executions defined by the specification to

[☆] This work was supported by the DST INRIA associated team, CNRS PEPS AABS, and ANR IMPRO project.

Email addresses: akshay@irisa.fr (S. Akshay), blaise.genest@irisa.fr (Blaise Genest), loic.helouet@irisa.fr (Loïc Hélouët), sf.yang@siat.ac.cn (Shaofa Yang)

be (timed) regular, which is a significant restriction in a concurrent setting where even simple examples may not be regular (e.g., the producer-consumer protocol).

In this paper, we propose a first decidability result for (globally) timed concurrent systems having a possibly *non-regular* set of behaviors. More specifically, we tackle the emptiness problem for TC-MSC graphs (which is undecidable in general [9]) by coming up with mild restrictions which are practically motivated and yet sufficient to prove decidability. Our technique to obtain decidability is to use a *regular set of representatives*. A set of representatives is a subset of executions such that every execution of the system has an equivalent execution (up to commutation) in this subset. This technique has been used previously in untimed settings [14, 10] and with the particular set of *existentially bounded executions* [10] as the regular set of representatives. In Section 3, it is formalized as a general technique on *timed* languages and applied to the set of *well-behaved timed executions*, i.e., timed executions where two events from the same scenario do not occur at dates that are arbitrarily apart (*non-drifting*) and only a limited number of events can occur in a unit of time (*non-Zeno*).

We state our main theorems in Section 4: the set of well-behaved timed executions of a TC-MSC graph is regular, and it is a set of representatives under the assumption that the TC-MSC graph is *well-formed*. Together, these imply that the emptiness problem is decidable for well-formed TC-MSC graphs. Intuitively, being well-formed prohibits specifications in which (1) events from the same scenario are forced to occur arbitrarily apart from each other (drifting), which is undesirable as it goes against the MSC-graph design, and (2) an unbounded number of events are forced to happen within one unit of time, which is unimplementable.

Proofs are detailed in Section 5. Regularity of the set of well-behaved executions exploits the fact that if node x appears sufficiently before node y in a path, then all events of x must occur before any event of y in any well-behaved execution of this path. Proving representativity for a well-formed TC-MSC graph is not trivial, as for each execution, we need to find a representative which is *both* non-drifting and has a limited number of events per unit of time, while being well-formed guarantees only the existence of two representatives, one of each kind. Further discussion regarding significance and practicality of our assumptions can be found in [17].

2. Time-Constrained MSC graphs

We fix a finite non-empty set of processes $\mathcal{P}$ that communicate through messages via reliable FIFO channels. For $p, q \in \mathcal{P}$, the communication alphabet is $\Sigma = \{p!q, p?q \mid p \neq q\}$ where the send action $p!q$ denotes a message sent from process p to q and the receive action $q?p$ denotes a message received by process q from p. Let $\mathbb{N}$ denote the set of natural numbers and $\mathcal{I}(\mathbb{N})$ denote the set of open and closed intervals whose end points are in $\mathbb{N}$, plus the intervals of the form $[c, \infty)$, (c, ∞), where $c \in \mathbb{N}$. We shall use intervals in $\mathcal{I}(\mathbb{N})$ to constrain the lower and upper bounds on the difference of occurrence times of events in a scenario. We remark that in what follows, intervals involving non-negative rationals can be easily simulated by scaling them to integers. We adopt the basic definitions from [1].

Definition 2.1. *A* time-constrained message sequence chart *(TC-MSC) over $\mathcal{P}$ and Σ is a tuple $T = (E, (<_p)_{p\in\mathcal{P}}, \lambda, \mu, \delta)$ where E is a finite non-empty set of* events*; $\lambda : E \to \Sigma$ labels each event with an action type in Σ such that:*

(i) Each $<_p \subseteq E_p \times E_p$ is a total order, where $E_p = \lambda^{-1}(\{p\} \times \{!, ?\} \times \mathcal{P})$. Members of E_p are termed p-events.

(ii) The message relation μ is a bijection from $E_{send} = \lambda^{-1}(\mathcal{P} \times \{!\} \times \mathcal{P})$ (send events) to $E_{recv} = \lambda^{-1}(\mathcal{P} \times \{?\} \times \mathcal{P})$ (receive events). For any e, f with $\mu(e) = f$, for some p, q, we have $\lambda(e) = p!q$ and $\lambda(f) = q?p$. For each e, e' with $\lambda(e) = \lambda(e') = p!q$ for some $p, q \in \mathcal{P}$, we have $e <_p e'$ iff $\mu(e) <_q \mu(e')$. (FIFO)

(iii) Writing $<$ for the transitive closure of $(\bigcup_{p\in\mathcal{P}} <_p) \cup \mu$, the time constraint labelling function δ associates an interval in $\mathcal{I}(\mathbb{N})$ to each pair of events $(e, f) \in E \times E$ with $e < f$.

With a slight abuse of notation, we write a TC-MSC as $T = (E, <, \lambda, \mu, \delta)$, with $<$ as above. A *linearization* of T is a sequence $\sigma = a_1 \dots a_\ell$ over Σ^*, where $\ell = |E|$ and such that E can be enumerated as $e_1 \dots e_\ell$ with $a_i = \lambda(e_i)$, and $e_i < e_j$ implies $i < j$ for any i, j in $\{1, \dots, \ell\}$. Note that due to the FIFO condition (see Condition (ii) in the definition above), the enumeration $e_1 \dots e_\ell$ is uniquely determined by $a_1 \dots a_\ell$. A TC-MSC T defines a collection of linearizations augmented with occurrence times such that the relative delay between each pair of causally ordered events falls within the interval

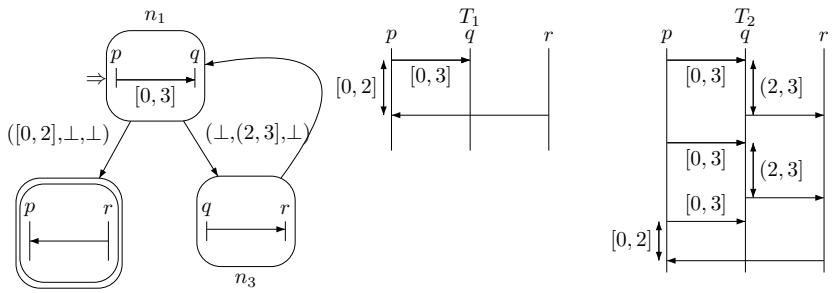

Figure 1: A TC-MSC graph G_1 and two TC-MSCs it generates

dictated by δ. To avoid confusion, we shall term occurrence times as *dates*: A *timed execution* w of T is a sequence $(a_1, d_1) \dots (a_\ell, d_\ell)$, where $a_1 \dots a_\ell$ is a linearization of T, each date d_i is a non-negative real for $i = 1, \dots, \ell$, and $d_1 \leq \dots \leq d_\ell$. Let $e_1 \dots e_\ell$ be the enumeration corresponding to the linearization $a_1 \dots a_\ell$. Then $e_i < e_j$ implies $d_j - d_i$ is in the interval $\delta(e_i, e_j)$.

To describe infinite collections of TC-MSCs, we use TC-MSC graphs:

Definition 2.2. *Let $\mathcal{T}$ be a finite non-empty set of TC-MSCs. A* TC-MSC graph *over $\mathcal{T}$ is a tuple $G = (N, \longrightarrow, n_{ini}, N_{fin}, \Lambda, \Delta)$ where N is a finite set of nodes, $\longrightarrow \subseteq N \times N$ a transition relation, $n_{ini} \in N$ the initial node, $N_{fin} \subseteq N$ the subset of final nodes, and $\Lambda : N \to \mathcal{T}$ labels each node with a TC-MSC from $\mathcal{T}$. Further, the mapping Δ associates each transition (n, n') in $\longrightarrow$ with a $\mathcal{P}$-indexed family of intervals in $\mathcal{I}(\mathbb{N})$, such that if either n or n' has no p-event, then the p-component of $\Delta(n, n')$ is $[0, \infty)$.*

For each p, we write $\Delta_p(n, n')$ for the p-th component of $\Delta(n, n')$. The interval $\Delta_p(n, n')$ specifies the range of relative delay on p when moving from n to n'. We write $\perp$ for the interval $[0, \infty)$. Figure 1 displays a TC-MSC graph G_1 whose nodes are n_1, n_2, n_3, with n_1 being the initial node and n_2 the final node. In n_1, the relative delay between the send event of p and the receive event of q is constrained to lie within $[0, 3]$. The $([0, 2], \perp, \perp)$ on transition (n_1, n_2) indicates $\Delta_p(n_1, n_2) = [0, 2]$, $\Delta_q(n_1, n_2) = \perp$, $\Delta_r(n_1, n_2) = \perp$. It asserts that the relative delay between the last event of p of n_1 and the first event of p of n_2 should be in $[0, 2]$. To reduce clutter in the figures, we omit time constraints of the form $\perp$ inside a TC-MSC labeling a node, and $(\perp)^{|\mathcal{P}|}$ on transitions.

We fix a TC-MSC graph $G = (N, \longrightarrow, n_{ini}, N_{fin}, \Lambda, \Delta)$. We write $n \longrightarrow n'$ for $(n, n') \in \longrightarrow$ and speak interchangeably of a node n and its associated TC-MSC $\Lambda(n)$. A TC-MSC graph defines a collection of TC-MSCs arising from concatenating TC-MSCs in paths of G. First, for a TC-MSC $T = (E, <, \lambda, \mu, \delta)$, we call the $<_p$-minimal event in E_p the *first* p-event, and the $<_p$-maximal event in E_p the *last* p-event. Simply put, for a transition (n, n'), the concatenation of n with n' is the TC-MSC resulting from placing n' after n, and for each process p, take $\Delta_p(n, n')$ to be the time constraint between the last p-event of n and the first p-event of n'. Formally, letting $\Lambda(n) = (E, <, \lambda, \mu, \delta)$ and $\Lambda(n') = (E', <', \lambda', \mu', \delta')$, the *concatenation* of $\Lambda(n)$ and $\Lambda(n')$, denoted $\Lambda(n) \circ \Lambda(n')$, is the TC-MSC $(E'', <'', \lambda'', \mu'', \delta'')$ detailed as follows. Firstly, E'' is the disjoint union of E and E'; λ'' agrees with λ on events in E, and with λ' on events in E'. Secondly, for each p, $<''_p$ is $<_p \cup <'_p \cup E_p \times E'_p$; μ'' is the union of μ and μ'. Lastly, for $e, f \in E''$ with $e <'' f$, $\delta''(e, f)$ is given as follows: (i) if $e, f \in E$, then $\delta''(e, f) = \delta(e, f)$; (ii) if $e, f \in E'$, then $\delta''(e, f) = \delta'(e, f)$; (iii) suppose $e \in E$, $f \in E'$. If for some p, e is the last p-event of n and f the first p-event of n', $\delta''(e, f) = \Delta_p(n, n')$, otherwise, $\delta(e, f) = \perp$. Note that the restriction $\Delta_p(n, n') = \perp$ whenever $E^n_p = \emptyset$ or $E^{n'}_p = \emptyset$ in Definition 2.2 is equivalent to the restrictions in [1, 9, 2] . It implies that $\circ$ is associative.

A *path* of G is a sequence of nodes $\rho = n_0 \dots n_\ell$ of G such that $n_0 = n_{ini}$ and $n_i \longrightarrow n_{i+1}$ for $i = 0, \dots, \ell - 1$. Since $\circ$ is associative, we can unambiguously define the TC-MSC induced by ρ, denoted T^ρ, to be $\Lambda(n_0) \circ \dots \circ \Lambda(n_\ell)$. The path ρ is *final* if $n_\ell \in N_{fin}$. The *TC-MSC language* of G is the set of TC-MSCs induced by final paths of G. For a TC-MSC T, let $L(T)$ denote its set of timed executions. For TC-MSC graph G, the *timed execution language* of G, denoted $L(G)$, is the union of $L(T^\rho)$ over final paths ρ of G. We say that a TC-MSC T (resp. a path ρ) is *consistent* iff $L(T) \neq \emptyset$ (resp. $L(T^\rho) \neq \emptyset$). In what follows, timed executions of the TC-MSC T^ρ are sometimes refered to as timed executions of ρ and $L(\rho)$ refers to $L(T^\rho)$.

We tackle the *emptiness problem for TC-MSC graphs*, which is: given a TC-MSC graph G, determine whether $L(G)$ is empty. The emptiness of $L(G)$ implies that for any TC-MSC T^ρ induced by a final path ρ of G, no assignment of dates to events in T^ρ can satisfy all the time constraints in T^ρ. Thus, such a G with $L(G) = \emptyset$ should be considered ill-specified, and should be checked for. However, it is known from [9] that the emptiness problem for TC-MSC graphs is undecidable. In [1, 2], decidability is obtained for locally-synchronized TC-MSC graphs. This syntactical restriction limits concurrency, and implies that the timed execution language is regular, which is a severe restriction. Indeed, even simple examples, such as G_1 from Figure 1 or the producer-consumer protocol, do not have regular timed execution languages.

3. Regular Set of Representatives

We advocate a technique of using *regular sets of representatives* (defined below) for obtaining decidability of the emptiness problem for TC-MSC graphs. This is a partial order reduction technique (since not all timed executions will be considered), which can handle TC-MSC graphs with *non-regular* timed execution languages. In this paper, regular will always stand for *timed regular*, i.e., languages accepted by finite timed automata [3]. Notice that timed regularity implies regularity of the untimed projection of the timed language.

Definition 3.1. *Let G be a TC-MSC graph. A subset R of $L(G)$ is called* a set of representatives *for G if for each consistent* final *path ρ of G, $R \cap L(T^\rho) \neq \emptyset$. If further R is (timed) regular, then R is called a* regular set of representatives.

It immediately follows that if R is a set of representatives for G, then $L(G) = \emptyset$ iff $R = \emptyset$. Now, many timed executions of a TC-MSC graph G are equivalent, in the sense that they are timed executions of the TC-MSC induced by the same final path of G. To check for emptiness of $L(G)$, it suffices to consider emptiness of a set R of representatives for G, instead of $L(G)$ itself. If R turns out to be regular and effective, then the emptiness problem for TC-MSC graphs can be decided. For example, consider G_2 in Figure 2. The language $L(G_2)$ is not regular. However, the set $\{\sigma_0, \sigma_0\sigma_1, \sigma_0\sigma_1\sigma_2, \ldots\}$, where $\sigma_i = (p!q, 4i)(q?p, 4i+1)(s!r, 4i+2)(r?s, 4i+3)$ for all $i \in \mathbb{N}$, is a regular set of representatives for G_2.

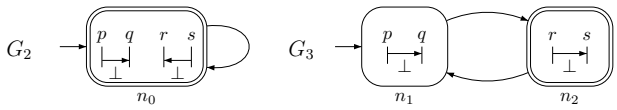

Figure 2: Two TC-MSC graphs G_2, G_3. Specification G_2 is scenario-connected and G_3 is not.

Thus, there are three elements in the technique of regular set of representatives: **(1)** choose a subset R of $L(G)$, **(2)** show that R is a set of representatives for G and **(3)** prove that R is regular.

We fix TC-MSC graph $G = (N, \longrightarrow, n_{ini}, N_{fin}, \Lambda, \Delta)$, a path $\rho = n_0 \ldots n_\ell$ of G, a timed execution $w = (a_1, d_1) \ldots (a_h, d_h)$ of ρ, and $e_1 \ldots e_h$ the enumeration of E associated with $a_1 \ldots a_h$. We start by giving a first set of representatives.

Definition 3.2. *Let K be an integer. We call w* K-drift-bounded *if for each $0 \leq u \leq \ell$, and $i, j \in \{1, \ldots, h\}$, if e_i, e_j are in $\Lambda(n_u)$, then $|d_i - d_j| \leq K$.*

Thus w is K-drift-bounded if the difference between the first and last date associated with an event of any TC-MSC $\Lambda(n_u)$ is bounded by K. Interpreting the scenario in each node of a TC-MSC graph as one phase or transaction of a distributed protocol, it is realistic to believe that at least some (but not necessarily all) executions of an implemented system are K-drift-bounded.

Now, for a TC-MSC graph G and an integer K, we say that G is *K-drift-bounded* if for every consistent path ρ of G, there *exists* a K-drift-bounded timed execution in $L(\rho)$. We emphasize that *all* timed executions of $L(\rho)$ are not required to be K-drift-bounded. Observe that, G being K-drift bounded implies that the set $L_K(G)$ of K-drift-bounded executions of G is a set of representatives of G. Unfortunately this set may not be regular. For example, G_2 in Figure 2 is K-drift-bounded for $K = 1$, but $L_K(G_2)$ is not regular for any K. Indeed, for any K, the untimed projections of $L_K(G_2)$ and of $L^{t0}(G_2)$, the timed language of G_2 where every event occurs at date 0, are the same. As the untimed projection of $L^{t0}(G_2)$ is not regular, $L_K(G_2)$ is not (timed) regular.

For $K' \in \mathbb{N}$, w has *at most K' events per unit of time* if for any $i, j \in \{1, \ldots, h\}$, $d_j - d_i \leq 1$ implies $j - i < K'$. A language L is *strongly non-Zeno* [5] if there exists $K' \in \mathbb{N}$ such that every execution of L has at most K' events per unit of time. It turns out that by imposing the following syntactical condition, a TC-MSC graph has a strongly non-Zeno set of representatives (this is one consequence of Theorem 4.2 below). We say that a transition (n, n') of

G is *positively constrained* if for every p, $\Delta_p(n, n')$ is *not* $[0, 0]$ (but can be $[0, 1)$, $[3, 3]$, $[2, \infty) \ldots$). G is *positively constrained* if every transition of G is positively constrained. This restriction does not imply that $L(G)$ is itself strongly non-Zeno: consider the positively constrained TC-MSC graph G_2 of Figure 2 (where transitions without labels are implicitly labeled by $\Delta_p = \perp$ for all p). $L(G_2)$ is not strongly non-Zeno since unboundedly many events can occur at date 0 (and hence within a unit of time). However, there exist timed executions where time elapses between positively constrained transitions.

We now present our regular set of representatives, namely the set $L_{K,K'}(G)$ of (K, K')-well-behaved timed executions, as well as the restriction needed on G for representativity.

Definition 3.3. *For $K, K' \in \mathbb{N}$, we say that w is* (K, K')-well-behaved *if w is K-drift-bounded and has at most K' events per unit of time. Further a TC-MSC graph is* K-well-formed *if it is K-drift-bounded and positively constrained.*

4. Main results

We can now state our main results. The first two theorems below hold with one more technical restriction imposed on TC-MSC graphs. However, the third theorem will establish decidability of the emptiness problem for TC-MSC graphs even without this technical restriction. A transition (n, n') of G is said to be *scenario-connected* if there exists a process p, s.t. both n and n' have at least one p-event. G is *scenario-connected* if every transition of G is scenario-connected. For instance, in Figure 2, G_2 is scenario-connected while G_3 is not.

Theorem 4.1. *Let $K, K' \in \mathbb{N}$. If G is scenario-connected, then $L_{K,K'}(G)$ is (timed) regular.*

For representativity, we need G to be well-formed.

Theorem 4.2. *Let $K \in \mathbb{N}$. If G is K-well-formed and scenario-connected, then $L_{K,K'}(G)$ is a set of representatives of G, with $K' = (4(|\mathcal{P}| + 1) + 2) \cdot |\mathcal{P}| \cdot M$, where M is the max number of events in a node of G.*

Subsequently, in Proposition 5.3, we will prove that we can lift the the scenario-connected assumption, which along with Theorems 4.1 and 4.2 implies:

Theorem 4.3. *Given a K-well-formed TC-MSC graph G for some integer K, it is decidable to determine whether $L(G) = \emptyset$.*

An immediate question is if, given a TC-MSC graph G and an integer K, one can decide whether G is K-well-formed. In fact, it turns out that this question is decidable. However, its proof involves vastly different techniques, and will be dealt with in a subsequent paper (see [16] for a draft).

5. Proofs of Main Results

Regularity: We prove Theorem 4.1 by constructing a timed automaton $\mathcal{A}$ which recognizes $L_{K,K'}(G)$. As in [15], $\mathcal{A}$ keeps track of nodes that have not yet been fully executed, plus unexecuted events of these nodes. The property proved in Lemma 5.2 shows that it suffices to remember finitely many nodes and events (and thus finitely many clocks).

Throughout this section, let $K, K' \in \mathbb{N}$, and $\rho = n_0 \ldots n_z$ a (non necessarily final) path of a scenario-connected TC-MSC graph G. Let $w = (a_1, d_1) \ldots (a_\ell, d_\ell)$ be a (K, K')-well-behaved timed execution of ρ. We denote by $e_1, \ldots, e_\ell$ the enumeration of events of T^ρ associated with $a_1, \ldots, a_\ell$, and $d(e_i) = d_i$ for all i. We also fix two constants $C_K = (|\mathcal{P}| + 1) \cdot K$ and $C_{K,K'} = 2 \cdot |\mathcal{P}| \cdot K' \cdot C_K$.

Lemma 5.1. *Let g be an event of node n_j and g' of node $n_{j'}$ of ρ, with $j \leq j'$. Then $d(g) \leq d(g') + C_K$.*

Proof. Since G is scenario-connected, it follows that for each n_t of ρ, there exists a process p_t, an event f_t in n_t and an event g_t in n_{t+1} such that both f_t, g_t are on process p_t. Thus, there exists a subsequence of $(f_t, g_t)_{j \leq t \leq j'}$ consisting of $k \leq |\mathcal{P}|$ pairs $(f_{\alpha(i)}, g_{\alpha(i)})_{1 \leq i \leq k}$ satisfying the following: (1) $f_{\alpha(1)}$ and g are in the same node, (2) $f_{\alpha(i+1)}$ and $g_{\alpha(i)}$ are in the same node for each $i < k$, (3) g' and $g_{\alpha(k)}$ are in the same node and (4) $f_{\alpha(i)}$ and $f_{\alpha(i')}$ are on different processes for all $i \neq i'$. Then, we have $f_{\alpha(i)} < g_{\alpha(i)}$ for each $i \leq k$, as they are on the same process. Hence $d(f_{\alpha(i)}) \leq d(g_{\alpha(i)})$. Also, $|d(g) - d(f_{\alpha(1)})| \leq K$ since g, $f_{\alpha(1)}$ are in the same node. Similarly, $|d(f_{\alpha(i+1)}) - d(g_{\alpha(i)})| \leq K$ for each $i \leq k$, and $|d(g_{\alpha(k)}) - d(g')| \leq K$. Thus, $d(g) \leq d(g') + (|\mathcal{P}| + 1) \cdot K$ (since $k \leq |\mathcal{P}|$). □

The next lemma is the key for the regularity.

Lemma 5.2. *Let n_x be a node of ρ such that $z - x \geq C_{K,K'}$. For all e in n_x and f in n_z, $d(e) < d(f)$.*

Proof. Recall that w has at most K' events per unit of time, and that each node of ρ contains at least one event. As $z - x \geq C_{K,K'}$, there are $m \geq 2K' \times C_K$ events $f_1 <_p \ldots <_p f_m$ in $n_x \ldots n_z$ and

on the same process p, for some $p \in \mathcal{P}$. Since w has at most K' events per unit of time, it follows that $d(f_m) > d(f_1) + 2C_K$. Applying Lemma 5.1 to e, f_1 and then again to f_m, f, we obtain $d(e) \leq d(f_1) + C_K < d(f_m) - 2C_K + C_K \leq d(f)$. □

This lemma implies that taking a_i the first event of node n_z appearing in w, every node n_x with $x \leq z - C_{K,K'}$ has been fully executed before a_i: for every event a_k of n_x, we have $k < i$. We can now sketch the construction of the timed automaton $\mathcal{A}$:

- States of $\mathcal{A}$ are sequences $(n_1, S_1) \cdots (n_k, S_k)$, such that $k \leq C_{K,K'}$, $n_1 \cdots n_k$ is a (not necessarily initial or final) path of G, for all i, S_i is a suffix (for $<_{n_i}$) of $\Lambda(n_i)$ and $S_1 \neq \emptyset$. $(n_0, \Lambda(n_0))$ is initial and $(n_f, \emptyset)$ is final. Intuitively, $n_1 \cdots n_k$ are the last nodes of the path seen during the execution, and S_i is the set of events not yet executed in n_i. Since $S_1 \neq \emptyset$, we can restrict to $k \leq C_{K,K'}$ using Lemma 5.2. There is a clock associated with every event e in $\bigcup_{i \in \{1,\ldots,k\}} \Lambda(n_i)$, called the e-clock, and every process $p \in \mathcal{P}$, called the p-clock.

- $(n_1, S_1) \cdots (n_k, S_k) \to (n'_1, S'_1) \cdots (n'_{k'}, S'_{k'})$ is a transition of $\mathcal{A}$ if one of the following holds:

  action: $k = k'$, $n'_j = n_j$ for all j, there exists i with $S_i = e \cdot S'_i$ and $S'_j = S_j$ for all $j \neq i$. Further, $S_1 \cdots S_{i-1}$ has no event on $p(e)$. The guard states that for all f in n_i, the f-clock is in $\delta(f, e)$, and if e is the first event of n_i on p, then the p-clock is in $\Delta_p(n_{i-1}, n_i)$. The transition resets the e-clock. Further, if e is the last event of n_i on p, then it resets the p-clock.

  new_node: $k' = k + 1 \leq C_{K,K'}$ $(n'_1, S'_1) \cdots (n'_k, S'_k) = (n_1, S_1) \cdots (n_k, S_k)$, $n_k \to n'_{k'}$ and $S'_{k'} = \Lambda(n'_{k'})$,

  deletion: $k > 1, k' = k - 1, S_1 = \emptyset$ and $(n'_1, S'_1) \cdots (n'_{k'}, S'_{k'}) = (n_2, S_2) \cdots (n_k, S_k)$.

  It is easy but tedious to show $L(\mathcal{A}) = L_{K,K'}(G)$.

Representativity: Now, we prove Theorem 4.2. Let $\rho = n_0 \ldots n_z$ be a consistent path of G. As G is K-drift-bounded, there is a K-drift-bounded timed execution $w = (a_1, d_1) \ldots (a_\ell, d_\ell)$ of T^ρ. We construct another timed execution $w' = (a_1, d'_1) \ldots (a_\ell, d'_\ell)$ from w by suitably modifying the dates such that w' is still an execution of T^ρ and w' is K-drift-bounded with at most $K' = (4C_K + 2) \cdot |\mathcal{P}| \cdot M$ events per unit of time, where M is the largest number of events in a node of G. The key idea in the construction of w' is to inductively postpone (when needed) the dates of all events of $n_x \cdots n_z$. By postponing, we ensure that there will exist some process p such that the difference between the date of the last p-event of n_{x-1} and the date of the first p-event of n_x is at least $1/2$. We use $1/2$ since all intervals have integer bounds. We do not postpone if it is already the case.

As before, let $e_1 \ldots e_\ell$ be an enumeration of events in T^ρ corresponding to $a_1 \ldots a_\ell$, and let us write $d(e)$ (resp. $d'(e)$) for the date d_i (resp d'_i), when $e = e_i$. To construct w', we first initialize $d'(e) = d(e)$ for each event e. Next, consider node n_1. Let Q be the set of processes p that participate in both n_0 and n_1. As G is scenario-connected, $Q \neq \emptyset$. For each p in Q, let $le_p(n_0)$ denote the last p-event of n_0 and $fe_p(n_1)$ denote the first p-event of n_1. If for some $p \in Q$, $d(fe_p(n_1)) - d(le_p(n_0)) \geq 1/2$, then for each event e in n_1, do not modify $d'(e)$ (it will not be modified later either). Otherwise, let $\theta_{max} < 1/2$ be the maximum of $d(fe_p(n_1)) - d(le_p(n_0))$ where p ranges over Q. For each event e in $n_1 \ldots n_z$, set $d'(e) = d(e) + 1/2 - \theta_{max}$. We emphasize that when considering node n_1, the above procedure postpones dates of events in n_1, *and* dates of events in $n_2 \ldots n_z$, by the same amount. Since G is positively constrained, the timed execution resulting from the above procedure is still in $L(T^\rho)$ and is still K-drift-bounded. We inductively carry on the above procedure to consider each of the nodes $n_2, \ldots, n_z$. The timed execution w' is obtained after considering all the nodes $n_0, \ldots, n_z$. It follows that w' is K-drift-bounded and is in $L(\rho)$.

It remains to show that w' has at most K' events per unit of time. It suffices to show that every pair of events e, f from two nodes n_x, n_z with $z - x > C = (4C_K + 2) \cdot |\mathcal{P}|$ satisfies $d'(f) > d'(e) + 1$. Indeed, then, for each t, the set of events e with $d(e) \in [t, t+1)$ is included into the set of events of $n_y \cdots n_{y+C}$ for some y. There are at most $K' = C \times M$ such events, hence there are at most K' events per unit of time.

If there are more than $C = (4C_K + 2) \cdot |\mathcal{P}|$ nodes between n_x and n_z, then there is some process p and more than $4C_K + 2$ pairs (f_i, g_i) of events in $n_x \cdots n_z$ with f_i the last p-event on some node $n_{\alpha i}$, g_i is the first event on node $n_{\alpha(i)+1}$, $d'(g_i) - d'(f_i) \geq \frac{1}{2}$, and α is stricly increasing. This is by construction of w'. That is, $d'(f_1) + 2C_K + 1 < d'(g_{C+1})$. Now, using Lemma 5.1 twice, for any event e of n_x and f of n_z, we obtain $d'(e) + 1 \leq d'(f_1) + C_K + 1 < d'(g_{C+1}) - 2C_K - 1 + C_K + 1 \leq d'(f)$.

Decidability: From Theorems 4.1 and 4.2, we readily conclude that, if G is scenario-connected, and K-well-formed, then one can decide whether $L(G)$ is empty. It remains to lift the scenario-connected restriction to prove Theorem 4.3. Suppose G is not scenario-connected. Let *NSC* denote the set of transitions of G that are not scenario-connected. Proposition 5.3 states the crucial observation that for any path $\rho = n_0 \cdots n_\ell$ with (n_i, n_{i+1}) in *NSC* for some i, the dates of events in $n_{i+1} \ldots n_\ell$ are *not* constrained in any way by the dates of events in $n_0 \cdots n_i$. This fact was also used in [9] along the same lines. Recall that in a transition (n, n'), if some $p \in \mathcal{P}$ does not participate in either n or n', then $\Delta_p(n, n') = \perp = [0, \infty)$.

Proposition 5.3. *Let $\rho = n_0 \cdots n_\ell$ a path of G. If (n_i, n_{i+1}) is in NSC then ρ is consistent iff both $n_0 \cdots n_i$ and $n_{i+1} \cdots n_\ell$ are consistent. If ρ is consistent, (n_i, n_{i+1}), (n_j, n_{j+1}) are both in NSC* and *$n_i = n_j$, then $n_0 \ldots n_i n_{j+1} \ldots n_\ell$ is also consistent.*

We now decompose G into a finite collection $\mathcal{H}$ of TC-MSC graphs, each of which is scenario-connected. We will decide the non-emptiness of $L(G)$ by considering the non-emptiness of $L(H)$ for every H in $\mathcal{H}$, which is decidable as shown earlier. Let N_1 be the subset of nodes n of G such that $(n', n) \in NSC$ for some node n' of G. Let N_2 be the subset of nodes n' of G such that $(n', n) \in NSC$ for some node n of G. For each $n \in N_1 \cup \{n_{ini}\}$, each $n' \in N_2 \cup N_{fin}$, we build the scenario-connected TC-MSC graph $H_{n,n'}$ from G as follows. The set of nodes of $H_{n,n'}$ is the same as G. $H_{n,n'}$ has n as initial node, and has one single final node n'. The transitions of $H_{n,n'}$ consist of all scenario-connected transitions of G. Let $\mathcal{H}$ be the collection of all such $H_{n,n'}$. For each $H_{n,n'}$ in $\mathcal{H}$, we can decide whether $L(H_{n,n'})$ is not empty since it is scenario-connected. From Proposition 5.3(1), $L(G)$ is *not empty* iff there exist a sequence H_{n_0,n_1}, H_{n_2,n_3}, ..., $H_{n_{2\ell},n_{2\ell+1}}$ in $\mathcal{H}$ such that $n_0 = n_{ini}$, $n_{2\ell+1} \in N_{fin}$, and for each $i \leq \ell$, $L(H_{n_{2i},n_{2i+1}})$ is not empty and (n_{2i+1}, n_{2i+2}) is in *NSC*. We can choose $n_0, n_2 \ldots, n_{2\ell}$ to be distinct according to Proposition 5.3(2). In particular, ℓ is at most the number of nodes of G.

6. Conclusion

In this paper, we have proved decidability of the language emptiness problem for a subclass of TC-MSC graphs. This problem was known to be undecidable in general and decidable for regular TC-MSC graphs. The subclass considered in this paper contains non-regular specifications and thus non-trivially extends the boundary of decidability. It is characterized in terms of bounds on the time a basic scenario takes, and disallows the constraint $[0, 0]$ on transitions. We believe (see also [17]) that these two requirements do not impair implementability, and meet what designers have in mind when designing a TC-MSC graph: event execution takes time, and the specification is split in phases.

[1] S. Akshay, M. Mukund, and K. Narayan Kumar. Checking coverage for infinite collections of timed scenarios. In *CONCUR 2007, LNCS 4703, pp. 181–196*. Springer.

[2] S. Akshay, P. Gastin, K. Narayan Kumar, and M. Mukund. Model checking time-constrained scenario-based specifications. In *FSTTCS 2010, LNCS 4855, pp. 290–302*. Springer.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Comp. Sci.*, 126(2):183–235, 1994.

[4] R. Alur and M. Yannakakis. Model checking of message sequence charts. In *CONCUR 1999, LNCS 1664, pp. 114–129*. Springer.

[5] C. Baier and T. Brihaye N. Bertrand, P. Bouyer. When are timed automata determinizable? In *ICALP (2) 2009, LNCS 5556, pp. 43–54*. Springer.

[6] P. Bouyer, S. Haddad, and P.-A. Reynier. Timed unfoldings for networks of timed automata. In *ATVA 2006, LNCS 4218, pp. 292–306*. Springer.

[7] F. Cassez, T. Chatain, and C. Jard. Symbolic unfoldings for networks of timed automata. In *ATVA 2006, LNCS 4218, pp. 307–321*. Springer.

[8] C. Dima and R. Lanotte. Distributed time-asynchronous automata. In *ICTAC 2007, LNCS 4711, 185–200*. Springer.

[9] P. Gastin, K. Narayan Kumar, and M. Mukund. Reachability and boundedness in time-constrained MSC graphs. In *Perspectives in Concurrency – A Festschrift for P. S. Thiagarajan*. Universities Press, 2009.

[10] B. Genest, D. Kuske, and A. Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inf. and Comp.*, 204(6):920–956, 2006.

[11] J. G. Henriksen, M. Mukund, K. N. Kumar, M. Sohoni, and P. S. Thiagarajan. A theory of regular MSC languages. *Inf. and Comp.*, 202(1):1–38, 2005.

[12] ITU-TS Recommendation Z.120: Message Sequence Chart (MSC '99), 1999.

[13] D. Lugiez, P. Niebert, and S. Zennou. A partial order semantics approach to the clock explosion problem of timed automata. *TCS*, 345(1):27–59, 2005.

[14] P. Madhusudan and B. Meenakshi. Beyond message sequence graphs. In *FSTTCS 2001, LNCS 2245, pp. 256–267*. Springer.

[15] A. Muscholl and D. Peled. Message sequence graphs and decision problems on mazurkiewicz traces. In *MFCS 1999, LNCS 1672, pp. 81–91*. Springer.

[16] S. Akshay, B. Genest, L. Hélouët, and S. Yang. Symbolically bounding the drift in time-constrained MSC graphs. Manuscript available at *http://www.crans.org/~genest/AGHY12.pdf*.

[17] S. Akshay, B. Genest, L. Hélouët, and S. Yang. Regular set of representatives for time-constrained MSC graphs. Technical Report RR-7823, HAL-INRIA, 2011.

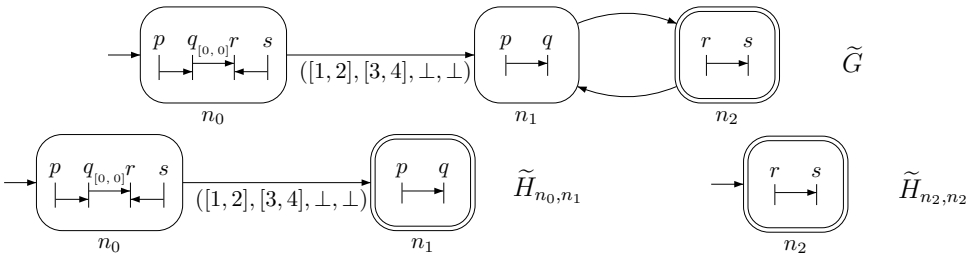

Figure 3: A TC-MSC graph $\widetilde{G}$ with unconstrained edges and its graph decomposition

Appendix

We give here the proof of Proposition 5.3.

Proposition 5.3. *Let $\rho = n_0 \cdots n_\ell$ a path of G. (1) If (n_i, n_{i+1}) is in NSC then ρ is consistent iff both $n_0 \cdots n_i$ and $n_{i+1} \cdots n_\ell$ are consistent. (2)If ρ is consistent, (n_i, n_{i+1}), (n_j, n_{j+1}) are both in NSC* and $n_i = n_j$, *then $n_0 \dots n_i n_{j+1} \dots n_\ell$ is also consistent.*

We start by assuming wlog that a given TC-MSC graph G has a unique initial node n_0 and final node n_f (else we can add a dummy initial node and edges from it to the old initial nodes and a dummy final node with edges from the old final nodes to this one). Recall that $(n_1, n_2) \in NSC$ if the processes that participate in n_1 and n_2 are disjoint. Observe that such edges are *unconstrained* by definition, i.e., they can only have the timing constraint $\perp = [0, \infty)$.

Proof. (1) One direction is straightforward. If ρ is consistent, then it means that we have a dated execution (w, d) corresponding to T^ρ. Now, projecting (w, d) to events of $n_0 \cdots n_i$ and $n_{i+1} \cdots n_\ell$, we respectively obtain a dated execution generated by $T^{n_0 \cdots n_i}$ and $T^{n_{i+1} \cdots n_\ell}$. Thus, both $T^{n_0 \cdots n_i}$ and $T^{n_{i+1} \cdots n_\ell}$ are consistent.

For the other direction, assume that both $T^{n_0 \cdots n_i}$ and $T^{n_{i+1} \cdots n_\ell}$ are consistent, with timed executions (w_1, d_1) and (w_2, d_2). We can then obtain the timed execution $(w_1 w_2, d)$ of T^ρ by delaying the timings sufficiently. More precisely, the timing d is obtained as $d(e) = d_1(e)$ for $e \in n_0 \cdots n_i$, and $d(e) = d_2(e) + \max$ for $e \in n_{i+1} \cdots n_\ell$, where max is the latest timing in d_1. Indeed, such a delay in the events is possible because we know that the edge $(n_i n_{i+1})$ is unconstrained and so a constant delay in all the events will not affect any constraint in this execution T^ρ. Thus $(w_1 w_2, d)$ is a timed execution of T^ρ completing part(1) of the proof.

(2) Finally, assume that ρ is consistent, (n_i, n_{i+1}), (n_j, n_{j+1}) are both in *NSC and* $n_i = n_j$. Consider a timed execution (w, d) of T^ρ. We first project it to events of $n_1 \cdots n_i$ to obtain the timed execution (w_1, d_1). We also project it to events of $n_j + 1 \cdots n_\ell$ to obtain the timed executions (w_2, d_2). Now, as above, the timed execution $(w_1 w_2, d')$ with $d'(e) = d_1(e)$ for $e \in n_0 \cdots n_i$, and $d'(e) = d_2(e) + \max$ for $e \in n_{i+1} \cdots n_\ell$, where max is the latest timing in d_1 is an execution of $T^{n_0 \dots n_i n_{j+1} \dots n_\ell}$, which is thus consistent. □

With the above proof of Proposition 5.3, let us now demonstrate how the proof of Theorem 5.3 (decidability) works on a simple example. Consider the graph $\widetilde{G}$ in Figure 3. Then (n_1, n_2) and (n_2, n_1) are unconstrained edges and $N_1 = \{n_0, n_2\}$ and $N_2 = \{n_1, n_2\}$ (the definitions of the sets N_1 and N_2 are as in the last paragraph of Section 5). The graph decomposition consists of $\widetilde{H}_{n_0,n_1}$ and $\widetilde{H}_{n_2,n_2}$ in the picture. Now observe that both $\widetilde{H}_{n_0,n_1}$ and $\widetilde{H}_{n_2,n_2}$ are consistent (that it there exist accepting paths) and (n_1, n_2) is unconstrained in $\widetilde{G}$. Thus by Proposition 5.3 $n_0 n_1 n_2$ is consistent and hence $L(\widetilde{G})$ is non-empty, which indeed is also clear from the picture.