

# Regular Set of Representatives for Time-Constrained MSC Graphs

S. Akshay<sup>1</sup>, Blaise Genest<sup>2</sup>, Loïc Hélouët<sup>3</sup>, Shaofa Yang<sup>4</sup>

<sup>1</sup> National University of Singapore

<sup>2</sup> CNRS, UMI IPAL joint with NUS and A\*STAR/I2R, Singapore.

<sup>3</sup> INRIA Rennes, France.

<sup>4</sup> SIAT, Chinese Academy of Sciences, China.

## ABSTRACT.

Systems involving both time and concurrency are notoriously difficult to analyze. Existing decidability results apply in settings where clocks on different processes cannot be compared or where 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 whether the set of timed executions generated by a TC-MSC graph is empty or not. This emptiness problem is known to be undecidable in general [10].

Our approach for obtaining decidability consists of two steps: (i) find a subset  $R$  of *representative* timed executions, that is, for which every timed execution of the system has an equivalent, up to commutation, timed execution in  $R$ , and (ii) prove that  $R$  is regular. This allows us to solve the emptiness problem under the assumption that the TC-MSC graph  $G$  is *well-formed*. In particular, a well-formed TC-MSC graph is prohibited from *forcing* any basic scenario to take an arbitrarily long time to complete. Secondly, it is forbidden from *enforcing* unboundedly many events to occur within a single unit of time. We argue that these restrictions are indeed practically sensible.

## 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) [13]. The timing information is captured by adding timing constraints between pairs of events, yielding time-constrained MSCs (denoted TC-MSCs). Further, infinite collections of MSCs are typically described using High-level Message Sequence Charts, or more basic forms called MSC-graphs, i.e. directed graphs whose nodes are labelled by MSCs. MSC-graphs can be easily generalized to time-constrained MSC graphs (TC-MSC graphs)[10], whose nodes are labelled by TC-MSCs and edges have additional timing constraints. In this paper, we are interested in obtaining decidability results for the analysis of TC-MSC graphs.

Obtaining decidability in the presence of both time and concurrency is a challenging issue. In particular, even the simple question of checking if there exists a timed execution

of a TC-MSC graph consistent with all the constraints is undecidable in general [10]. This is the *emptiness problem*, which in the case of (sequential) timed automata is known to be decidable [3]. Extending such decidability results to distributed systems has been done only in two particular and limited settings. In the first setting, [14, 9] consider clocks that 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 or must take [1, 2, 6, 7]. However, these papers restrict the concurrency in a structural way, for instance considering only locally synchronized (see [15, 4, 12]) MSC-graphs (in [1, 2]) or only safe Petri Nets (in [6, 7]). The language of the specification is then forced to be regular, which is a significant restriction in a concurrent setting where even simple behaviors may not be regular (e.g., the producer-consumer protocol).

In this paper, we propose the first decidability result for timed concurrent systems with global clocks having a possibly *non-regular* set of behaviors. More specifically, we tackle the emptiness problem for TC-MSC graphs (which is undecidable in general [10]) by coming up with mild restrictions which are practically motivated and yet sufficient to prove decidability. Indeed, this is important as the only non trivial way to have an empty language is that every possible timing of the events conflict with some constraint. Such specification should then be considered as ill-formed, which has to be reported to the designer.

Our technique to obtain decidability of the emptiness problem 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- up to commutation- execution in this subset. This technique has been used previously in untimed settings [11, 8] and with the fixed set of *existentially bounded executions* [11] as the regular set of representatives. In Section 3, it is formalized as a general technique on timed languages, and is applied on a new regular set of representatives called the set of *well-behaved timed executions*. This is the subset of timed executions where two events from the same scenario do not occur at dates that are arbitrarily apart from each other (*non drifting*), and where there are a limited number of events done in one unit of time (*non Zeno*). This demonstrates the versatility of the technique.

We state our main theorems in Section 4: the set of well-behaved timed executions 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 forbids 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.

The proofs of these Theorems are detailed in Section 5. The 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 non trivial, as for each execution, one needs 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. Finally, we discuss in Section 6 the significance and practicality of our assumptions as well as related work.

## 2 Time-Constrained MSC graphs

We begin by fixing a finite nonempty set of processes  $\mathcal{P}$  that communicate through messages via reliable FIFO channels. Let  $p, q$  range over  $\mathcal{P}$ . The communication alphabet is given by  $\Sigma = \{p!q, p?q \mid p \neq q\}$  where the sending action  $p!q$  denotes a message sent from process  $p$  to  $q$  and the receiving action  $p?q$  denotes a message received by process  $q$  from  $p$ . Let  $\mathbb{N}$  denote the set of natural numbers. Further, let  $\mathcal{I}(\mathbb{N})$  denote the set of open and closed intervals whose end points are in  $\mathbb{N}$ , plus the intervals of the form  $[c, \infty)$ ,  $(c, \infty)$ , 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. Note that intervals involving non-negative rationals can be easily simulated by scaling them to integers. We adopt the basic definitions from [1].

**DEFINITION 1.** A time-constrained message sequence chart (TC-MSC) over  $\mathcal{P}$  and  $\Sigma$  is a tuple  $T = (E, (\prec_p)_{p \in \mathcal{P}}, \lambda, \mu, \delta)$  where  $E$  is a finite nonempty set of events;  $\lambda : E \rightarrow \Sigma$  labels each event with a letter in  $\Sigma$ . The following conditions are satisfied:

- Each  $\prec_p \subseteq E_p \times E_p$  is a total order, where  $E_p = \lambda^{-1}(\{p\} \times \{!, ?\} \times \mathcal{P})$ . Members of  $E_p$  are termed  $p$ -events.
- The message relation  $\mu$  is a bijection from  $E_{\text{send}} = \lambda^{-1}(\mathcal{P} \times \{!\} \times \mathcal{P})$  (the sending events) to  $E_{\text{recv}} = \lambda^{-1}(\mathcal{P} \times \{?\} \times \mathcal{P})$  (the receiving events). For any  $e, f$  with  $\mu(e) = f$ , it is the case that for some  $p, q$ , we have  $\lambda(e) = p!q$  and  $\lambda(f) = q?p$ . For each  $p$ , any events  $e, e'$  in  $E_p$ , satisfies that  $e \prec_p e'$  iff  $\mu(e) \prec_q \mu(e')$ , where  $\mu(e)$  is a  $q$ -event.
- Write  $<$  for the transitive closure of  $(\bigcup_{p \in \mathcal{P}} \prec_p) \cup \mu$ . The time constraint labelling function  $\delta$  associates an interval in  $\mathcal{I}(\mathbb{N})$  to each pair of events  $(e, f) \in E \times E$  with  $e < f$ .

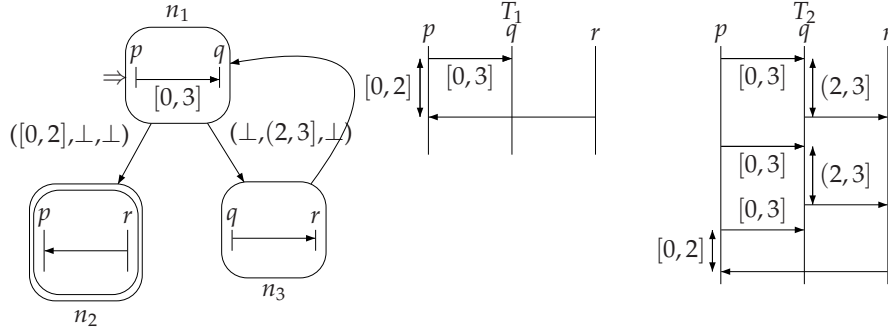
Abusing notation, we write a TC-MSC more conveniently as  $(E, <, \lambda, \mu, \delta)$ , with the understanding that  $<$  is in fact defined from  $\prec_p$  for  $p$  ranging over  $\mathcal{P}$ , and  $\mu$ . In what follows, we fix a TC-MSC  $T = (E, <, \lambda, \mu, \delta)$ .

A linearization of  $T$  is a sequence  $\sigma = a_1 \dots a_\ell$  over  $\Sigma^*$ , 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, 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 of each pair of causally ordered event falls within the interval dictated by  $\delta$ . To avoid confusion, we shall term occurrence times as *dates*:

**DEFINITION 2.** Let  $T$  be as fixed above. 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, one uses the formalism of TC-MSC graphs.

**DEFINITION 3.** Let  $\mathcal{T}$  be a finite nonempty set of TC-MSCs. A TC-MSC graph over  $\mathcal{T}$  is a tuple  $G = (N, \longrightarrow, n_{\text{ini}}, N_{\text{fin}}, \Lambda, \Delta)$  where  $N$  is a finite set of nodes,  $\longrightarrow \subseteq N \times N$  a transition relation,  $n_{\text{ini}} \in N$  the initial node,  $N_{\text{fin}} \subseteq N$  the subset of final nodes, and  $\Lambda : N \rightarrow \mathcal{T}$  labels each node with a TC-MSC from  $\mathcal{T}$ . Further, the mapping  $\Delta$  associates each transition  $(n, n')$  in  $\longrightarrow$  with a  $\mathcal{P}$ -indexed family of intervals in  $\mathcal{I}(\mathbb{N})$ .


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

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 whose nodes consist of  $n_1, n_2, n_3$ . The initial node  $n_1$  is indicated by an incoming arrow. There is only one final node,  $n_3$ . In node  $n_1$ , the relative delay between the sending event of  $p$  and the receiving 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 shall omit all time constraints of  $\perp$  inside a basic scenario.

We fix a TC-MSC graph  $G = (N, \longrightarrow, n_{ini}, N_{fin}, \Lambda, \Delta)$ . We write  $n \longrightarrow n'$  for  $(n, n') \in \longrightarrow$ . We shall 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  $G$ . We first define this mechanism of *concatenating* adjacent TC-MSCs in  $G$ . 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  $n'$ .

**DEFINITION 4.** Let  $G$  be as fixed above and  $(n, n')$  a transition of  $G$ . Let  $\Lambda(n) = (E, <, \lambda, \mu, \delta)$  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  $E'$ ;  $\lambda''$  agrees with  $\lambda$  on events in  $E$ , and with  $\lambda'$  on events in  $E'$ . Secondly, for each  $p$ ,  $<''_p$  is  $<_p \cup <'_p \cup E_p \times E'_p$ ;  $\mu''$  is the union of  $\mu$  and  $\mu'$ . 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 as in [1, 10, 2], in the above definition, if either  $E_p = \emptyset$  or  $E'_p = \emptyset$ , then  $\Delta_p(n, n')$  is disregarded in  $\Lambda(n) \circ \Lambda(n')$ . That is, we can assume without loss of generality that  $\Delta_p = \perp$  for each such  $p$  on such a transition  $(n, n')$ . Observe that now  $\circ$  is associative.

A *path* of  $G$  is a sequence of nodes  $\rho = n_0 \dots n_\ell$  of  $G$  such that each  $n_0 = n_{ini}$  and  $n_i \longrightarrow n_{i+1}$  for  $i = 0, \dots, \ell - 1$ . We emphasize that a path always starts with the initial node. Since  $\circ$  is associative, we can unambiguously define the TC-MSC induced by  $\rho$ , denoted  $T^\rho$ , to be  $\Lambda(n_0) \circ \dots \circ \Lambda(n_\ell)$ . A path is *final* if its last node is in  $N_{fin}$ .

The *TC-MS language* of  $G$  is the set of TC-MSCs induced by final paths of  $G$ . For a TC-MS  $T$ , let  $L(T)$  denote its set of timed executions. For the TC-MS graph  $G$ , the *timed execution language* of  $G$ , denoted  $L(G)$ , is the union of  $L(T^\rho)$  ranging over final paths  $\rho$  of  $G$ . We say that a TC-MS  $T$  (resp. a path  $\rho$ ) is *consistent* iff  $L(T) \neq \emptyset$  (resp.  $L(T^\rho) \neq \emptyset$ ).

We tackle the *emptiness problem of TC-MS graphs*, which can be stated as: given a TC-MS graph  $G$ , determine whether  $L(G)$  is empty. The emptiness of  $L(G)$  implies that for any TC-MS  $T^\rho$  induced by a final path  $\rho$  of  $G$ , no assignment of dates to events in  $T^\rho$  can satisfy all the time constraints in  $T^\rho$ . Thus, such a  $G$  with  $L(G) = \emptyset$  should be considered ill-specified, and thus it should be checked for. However, it is known [10] that:

**PROPOSITION 5.** [10] *The emptiness problem of TC-MS graphs is undecidable.*

In [1, 2], decidability is obtained for locally-synchronized TC-MS 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* for obtaining decidability of the emptiness problem of TC-MS graphs. This is a partial order reduction technique (since not all timed executions will be considered), which can handle TC-MS graphs with *non-regular* timed execution languages. We begin with the following definition:

**DEFINITION 6.** *Let  $G$  be a TC-MS graph. A subset  $R$  of  $L(G)$  is called a set of representatives for  $G$  if for each consistent final path  $\rho$  of  $G$ ,  $R \cap L(T^\rho) \neq \emptyset$ .*

It immediately follows that:

**PROPOSITION 7.** *If  $R$  is a set of representatives for  $G$ , then  $L(G) = \emptyset$  iff  $R = \emptyset$ .*

Indeed, many timed executions of a TC-MS graph  $G$  are equivalent, in the sense that they are timed executions of the TC-MS 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 of TC-MS graphs can be decided. For example, consider  $G_2$  in Figure 2. The language  $L(G_2)$  is not regular. However, the set  $\{\sigma_0, \sigma_0\sigma_1, \sigma_0\sigma_1\sigma_2, \dots\}$ , 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  $G_2$ .

Thus, there are three elements in the technique of regular set of representatives:

1. Choose a subset  $R$  of  $L(G)$ .
2. Show that the chosen subset  $R$  is a set of representatives for  $G$ .
3. Prove that  $R$  is regular.

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

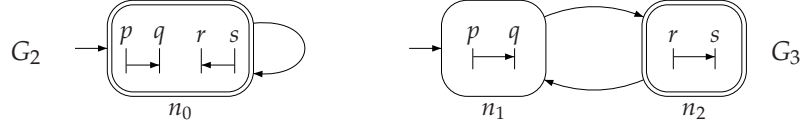


Figure 2: Two TC-MSC graphs  $G_2, G_3$ . Specification  $G_2$  is scenario-connected and  $G_3$  is not.

**DEFINITION 8.** Let  $K$  be an integer. We say that  $w$  is  $K$ -drift-bounded if for each  $0 \leq u \leq \ell$ , and  $i, j \in \{1, \dots, h\}$ , if  $e_i, e_j$  are in  $\Lambda(n_u)$ , then  $|d_i - d_j| \leq K$ .

In other words,  $w$  is  $K$ -drift-bounded if the difference between the first and the last date associated with an event of any basic scenario is bounded by  $K$ . Interpreting the scenario in each node of a TC-MSC graph as one phase or one transaction of a distributed protocol, it is realistic to believe that 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  $\rho$  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 with the above definitions,  $G$  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, for the TC-MSC graph  $G_2$  in Figure 2,  $L_K(G_2)$  is not regular, for any  $K$ , because of timed executions with an unbounded number of events per unit of time. Formally, for an integer  $K'$ , we say that  $w$  has at most  $K'$  events per unit of time if for any  $i, j \in \{1, \dots, h\}$ ,  $d_j - d_i \leq 1$  implies  $j - i < K'$ .

This phenomenon of having unboundedly many events within one unit of time is known as Zenoness and occurs commonly in timed specifications. It turns out that by imposing the next simple syntactical condition, one can prevent a TC-MSC graph from enforcing Zenoness (this is one consequence of Theorem 12 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 it can be  $[0, 1]$ ,  $[3, 3]$ ,  $[2, \infty)$ ...). We say that  $G$  is *positively constrained* if every transition of  $G$  is positively constrained. We can now present our regular set of representatives, namely the set  $L_{K, K'}(G)$  of  $(K, K')$ -well-behaved timed executions, defined as follows:

**DEFINITION 9.** For integers  $K, K'$ , 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.

To get the representativity of  $L_{K, K'}(G)$ , we need a restriction on the TC-MSC graph:

**DEFINITION 10.** We say that 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 of 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 11.** *Let  $K, K'$  be integers. If  $G$  is scenario-connected, then  $L_{K,K'}(G)$  is regular.*

Concerning the representativity, we need  $G$  to be well-formed.

**THEOREM 12.** *Let  $K$  be an integer. If  $G$  is  $K$ -well-formed and scenario-connected, then  $L_{K,K'}(G)$  is a set of representatives of  $G$ , with  $K' = 4(|\mathcal{P}| + 1)^2 \cdot K \cdot M$ , where  $M$  is the maximal number of events in one node of  $G$ .*

Theorems 11,12 imply, with effort to lift the scenario-connected assumption:

**THEOREM 13.** *Given a  $K$ -well-formed TC-MSG graph  $G$  for some integer  $K$ , it is decidable to determine whether  $L(G) = \emptyset$ .*

An immediate question is if given a TC-MSG graph  $G$  and an integer  $K$ , one can effectively determine whether  $G$  is  $K$ -well-formed. In fact, it turns out that this question is decidable. However, this problem is not in the scope of this paper, and its proof, involving vastly different techniques, will be dealt with in a subsequent paper.

## 5 Proofs of Representativity, Regularity and Decidability

### 5.1 Regularity

We prove Theorem 11 by constructing a timed automaton ([3])  $\mathcal{A}$  which recognizes  $L_{K,K'}(G)$ . In general, to recognize  $L(G)$ , a timed automaton  $\mathcal{A}$  needs to generate every linearization of the TC-MSG induced by every path of  $G$ . In linearizing events from the TC-MSG induced by a path  $\rho = n_0 \dots n_z$ , the discrete state of  $\mathcal{A}$  keeps track of nodes which have not yet been (fully) executed, and for each such node memorizes which events have yet to be executed [15]. To ensure that time constraints are respected when linearizing  $\rho$ ,  $\mathcal{A}$  needs to draw clocks from a fixed pool to keep track of the relative delay between an executed event  $e$  and an event  $f$  not yet executed, when  $e, f$  are in the same node or when  $e$  is the last  $p$ -event of  $n_i$  and  $f$  is the first  $p$ -event of  $n_{i+1}$  [1]. In general, this requires memorizing an arbitrarily large number of events, and it requires an infinite pool of clocks.

However, to recognize  $L_{K,K'}(G)$ , it suffices to remember a finite number of nodes and events, which follows from the following crucial property. Notice that bounding the number of nodes and events also bounds the set of clocks needed [1]. If  $\rho = n_0 \dots n_z$  is the current path (that is, at least one event of  $n_z$  has been executed), then the not yet (fully) executed nodes are in the suffix  $n_y \dots n_z$  of  $\rho$ , where  $z - y$  is bounded by a fixed constant  $C$  depending only on  $|\mathcal{P}|, K, K'$ . This property is proved formally in Lemma 14. Once this is established, the detailed construction of  $\mathcal{A}$ , which is sketched above, follows precisely [1] (see also [2]). The main difference is that in [1, 2], the bound  $C$  is obtained by restricting the concurrency.

Before stating the lemma, let us fix some notations. Let  $G, K, K'$  be as given in theorem 11. Let  $C_{K,K'} = 2 \cdot |\mathcal{P}| \cdot (|\mathcal{P}| + 1) \cdot K \cdot K'$  and  $\rho = n_0 \dots n_z$  be a path of  $G$ . Let  $w = (a_1, d_1) \dots (a_\ell, d_\ell)$  be a  $(K, K')$ -well-behaved timed execution of  $\rho$ . We denote by  $e_1, \dots, e_\ell$  the enumeration of events of  $T^\rho$  associated with  $a_1, \dots, a_\ell$ . For all  $i \leq \ell$ , we define  $d(e_i) = d_i$ .

**LEMMA 14.** *Let  $n_x$  be a node of the path  $\rho = n_0 \dots n_z$  given above, and such that  $z - x \geq C_{K,K'}$ . Then  $d(e) < d(f)$  for all events  $e$  of  $n_x$  and  $f$  of  $n_z$ .*

PROOF. Let  $Events_x$  be the collection of events in the suffix  $n_x \dots n_z$ . Assume that  $z - x$  is sufficiently large. From the fact that  $w$  has at most  $K'$  events per unit of time, we first prove that there exist two events  $g, g'$  in  $Events_x$  such that  $d(g') > d(g) + 2C_1$ , for some large  $C_1$ . Note that each node of  $\rho$  contains at least one event. As  $z - x$  is sufficiently large, there are  $m \geq 2K' \times C_1$  events  $f_1 <_p \dots <_p f_m$  in  $Events_x$  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  $d(f_m) > d(f_1) + 2C_1$ .

Consider the event  $g$  built as above, and let  $n_s$  be the state it is in. We will prove that  $d(g) \geq d(e) - |\mathcal{P}| \cdot K$  (actually this is true for all events  $g$  in  $Events_x$ ). Since  $G$  is scenario-connected, it follows that for each node  $n_t$  of  $\rho$ , there exists a process  $p_t$ , an event  $f_t$  in node  $n_t$  and an event  $g_t$  in node  $n_{t+1}$  such that both  $f_t, g_t$  are on process  $p_t$ . There exists a subsequence of  $(f_t, g_t)_{0 \leq t \leq z}$  consisting of  $|\mathcal{P}|$  pairs  $(f'_i, g'_i)_{i \leq |\mathcal{P}|}$  s.t.  $f_1$  and  $e$  are in the same node,  $f'_{i+1}$  and  $g'_i$  are in the same node for each  $i < |\mathcal{P}|$  and  $g$  and  $g'_{|\mathcal{P}|}$  are in the same node. To obtain this, we only ensure that  $f'_i$  and  $g'_j$  are on different processes for all  $i \neq j$ .

First, we have  $f'_i < g'_i$  for each  $i \leq |\mathcal{P}|$ , as they are on the same process. Hence  $d(f'_i) \leq d(g'_i)$ . Also,  $|d(e) - d(f'_1)| \leq K$  since  $e, f'_1$  are in the same node. Similarly,  $|d(f'_{i+1}) - d(g'_i)| \leq K$  for each  $i \leq |\mathcal{P}|$ , and  $|d(g'_u) - d(g)| \leq K$ . Thus,  $d(g) \geq d(e) - (|\mathcal{P}| + 1) \cdot K$ . The same analysis gives  $d(f) \geq d(g') - (|\mathcal{P}| + 1) \cdot K$ . Taking  $C_1 = (|\mathcal{P}| + 1) \cdot K$ , and recalling that  $d(g') > d(g) + 2C_1$ , we get  $d(f) > d(e)$ .  $\blacksquare$

## 5.2 Representativity

Secondly, we prove theorem 12. Let  $\rho = n_0 \dots n_z$  be a consistent path of  $G$ . As  $G$  is  $K$  drift bounded, there exists a  $K$ -drift-bounded timed execution  $w = (a_1, d_1) \dots (a_\ell, d_\ell)$  of  $T^\rho$ . We construct another timed execution  $w' = (a_1, d'_1) \dots (a_\ell, d'_\ell)$  from  $w$  by suitably modifying the dates, such that  $w'$  is still an execution of  $T^\rho$ , is  $K$ -drift-bounded, and has at most  $K'$  events per unit of time, for a suitable choice of  $K'$ . The key idea in the construction of  $w'$  is to inductively postpone (when needed/possible) the dates of all events of  $n_x \dots n_z$ . 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  $1/2$ . If it is already the case, then we do not postpone.

As before, let  $e_1 \dots e_\ell$  be an enumeration of events in  $T^\rho$  corresponding to  $a_1 \dots a_\ell$ , 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  $Q \neq \emptyset$  be the collection of those processes  $p$  such that both  $n_0$  and  $n_1$  have  $p$ -events. 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  $n_1 \dots n_z$ , 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  $n_2 \dots n_z$ , 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  $n_2, \dots, n_z$ . The timed execution  $w'$  is obtained after considering all the nodes  $n_0, \dots, n_z$ . 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, for a sufficiently large  $K'$ . Let  $M$  be the maximum number of events in any node of  $G$ . It suffices to show that every pair of events  $e, f$  from two nodes  $n, n'$  sufficiently apart in the path (and  $n$  appearing first) satisfies  $d'(f) > d'(e) + 1$ , as then events in the same unit of time belong to nodes sufficiently close, and there are at most  $(M$  times the number of nodes) such events. The proof follows on the exact same lines as the regularity proof above, with the same constant  $C_1 = (|\mathcal{P}| + 1)K$ . We show that we can find two events  $g, g'$  on nodes between  $n, n'$  with  $d'(g') > d'(g) + 2C_1 + 1$ . The regularity proof showed that fact using the bounded number of events per unit of time of  $w$ . Here, this is what we want to prove, so we rely instead on the construction of  $w'$  above. If there are more than  $|\mathcal{P}| \cdot (4C_1 + 2)$  nodes between  $n$  and  $n'$ , then there is one process  $p$  which is in  $Q$  for more than  $4C_1 + 2$  transitions. By construction of  $w'$ , on each of these transitions, time elapses by  $1/2$  on  $p$ , and the result follows. Now, we can apply the results from the regularity proof, to get  $d'(f) > d'(g') - C_1$  and  $d'(g) > d'(e) - C_1$ , which gives  $d'(f) > d'(e) + 1$ . That is,  $K' = |\mathcal{P}| \cdot M \cdot (4C_1 + 2)$  is sufficient.

### 5.3 Decidability

From theorem 11 and 12, one readily concludes:

**COROLLARY 15.** *Let  $G$  be a TC-MSG graph as fixed earlier, and  $K$  an integer. If  $G$  is scenario-connected, and  $K$ -well-formed, then one can effectively determine whether  $L(G)$  is empty.*

We now lift the scenario-connected restriction, proving Theorem 13. Suppose  $G$  is not scenario-connected. Let  $NSC$  denote the set of transitions of  $G$  that are not scenario-connected. Proposition 16 states the crucial observation that for any path  $\rho = n_0 \cdots n_\ell$  with  $(n_i, n_{i+1})$  in  $NSC$ , the dates of events in  $n_{i+1} \dots n_\ell$  are *not* constrained in any way by the dates of events in  $n_0 \cdots n_i$ . This fact was also used in [10] along the same lines.

**PROPOSITION 16.** *Suppose  $\rho = n_0 \cdots n_\ell$  is a path of  $G$ .*

- *If  $(n_i, n_{i+1})$  is in  $NSC$  where  $0 \leq i < \ell$ , then  $\rho$  is consistent iff both path  $n_0 \cdots n_i$  and path  $n_{i+1} \cdots n_\ell$  are consistent.*
- *Suppose  $\rho$  is consistent. If  $(n_i, n_{i+1}), (n_j, n_{j+1})$  are both in  $NSC$  where  $0 \leq i < j < \ell$ , and  $n_i = n_j$ , then the path  $n_0 \dots n_i n_{j+1} \dots n_\ell$  is also consistent.*

We now decompose  $G$  into a finite collection  $\mathcal{H}$  of TC-MSG 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 by Corollary 15. 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-MSG 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 following Corollary 15. From the first item of Proposition 16,  $L(G)$  is *not empty* iff there exist a sequence  $H_{n_0, n_1}, H_{n_2, n_3}, \dots, H_{n_{2\ell}, n_{2\ell+1}}$  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 chose  $n_0, n_2 \dots, n_{2\ell}$  to be distinct according to the second item of Proposition 16. In particular,  $\ell$  is at most the number of nodes of  $G$ , and an algorithm follows.

## 6 Discussion and Related Work

In this section, we discuss the role of the restrictions we imposed.

### 6.1 Practical Motivation of Drift-boundedness

Interpreting a node of a TC-MSG graph as specifying communications in one phase or one transaction of a distributed protocol, it is realistic to expect that *in an implementation of the specification*, the scenario labeling the node is performed in a bounded time, that is the difference between the date of the first and the last event of the scenario is bounded. One may thus be tempted to impose that *all* timed executions of the language of a TC-MSG graph are  $K$ -drift-bounded, for some uniform constant  $K$ . However, TC-MSG graphs are meant for high-level abstract description of system behaviour at an early design stage. The designer should implement flexibility in the TC-MSG graphs to adapt to different implementation hardware, and this sort of strong restriction is not acceptable. This is the reason why our  $K$ -drift-bounded restriction is existential in nature: we only demand that each path of a TC-MSG graph contains at least one  $K$ -drift-bounded timed execution, which can then be chosen in the implementation process. Notice that being  $K$ -drift-bounded implies to be also  $\hat{K}$ -drift-bounded for  $\hat{K} \geq K$ , which can then adapt to many different hardware. Conversely, a TC-MSG graph which is not  $K$ -drift-bounded for any  $K$  have executions that *require* unbounded delay to finish scenarios, which is then not implementable.

**Non regular examples:** every TC-MSG graph used as example in this paper is  $K$ -drift-bounded for some small  $K$  ( $K \leq 3$ ), and its timed execution language is not regular. For instance, consider  $G_1$  from Figure 1. No assumption is made on the speed of  $r$  (the constraint for  $r$  is  $\perp$  in the loop), which may delay message reception for an unbounded amount of time. Hence, at a given moment, there might be an arbitrary number of messages transiting from  $q$  to  $r$ . Note that even if process  $r$  is allowed to drift from  $q$ , the specification still allows behaviors in which they send and consume messages at the same pace. Hence unbounded drifting can occur, but can also be avoided.

**Why it is necessary but not sufficient:** First, we remark that the undecidability proof from [10] uses a family of TC-MSG graphs  $(G_i)_{i \in \mathbb{N}}$  which is not  $K$ -drift-bounded for any  $K$ . Second, drift-boundedness is not sufficient to obtain a regular set of representatives. For instance, the set  $L_K(G_2)$  of representatives for  $G_2$  (see Figure 2) is not regular for any  $K \geq 0$ . Indeed, consider the set of timed executions where every event is executed at time 0. This set is not regular as the number of messages from  $p$  to  $q$  and from  $r$  to  $s$  should be the same. Also, this set is included in  $L_K(G_2)$ . Thus, the untimed projection of  $L_K(G_2)$  and of this set are equal, and  $L_K(G_2)$  is not regular.

### 6.2 Practical motivation of positively constrained transitions

Consider a modified version of the TC-MSG graph  $G_2$  of Figure 2, denoted  $G'_2$ , where constraints  $[0, 0]$  have been added to all messages and on all processes in the unique transition. Clearly, this enforces all the events to be done exactly at date 0: for all  $k$ , all the timed executions of the TC-MSG associated with the path  $\rho = (n_0)^k$  of  $G'_2$  have more than  $k$  events per unit of time. Thus, this specification is not implementable as it forces an unbounded num-

ber of events to be done in 1 unit of time. Hence, it makes sense to disallow such constraints  $[0, 0]$  by requiring the system to be scenario-connected.

**Related Work:** The usual notion of *strongly non-Zeno* timed automaton [5] requires that there exists a bound  $K'$  such that all timed executions have at most  $K'$  events per unit of time. This imposes a given pace at which events can be done, which is very restrictive for a high-level specification. In comparison, positively constrained does not impose a pace and allows these kinds of executions in general, but implies that there exists at least one well-behaved execution for each path. For example, taking the positively constrained TC-MSC graph  $G_2$  of Figure 2, it is possible for all the events to happen at date 0.

**Why it is not sufficient:** Notice that we can slightly change the TC-MSC graph used in the undecidability proof of [10] to make it positively constrained, hence positively constrained alone does not suffice to get decidability, and a restriction such as drift boundedness is required. In fact, in this paper, we have shown that being both  $K$ -drift-bounded and positively constrained is sufficient to get decidability.

### 6.3 Scenario-connected TC-MSC graphs

The motivation behind the scenario-connected restriction is simple. Theorem 11,12 do not hold if we remove this restriction. It suffices to consider the TC-MSC graph  $G_3$  of Figure 2 which is not scenario-connected. The timed executions  $(p!q, 1)(q?p, 2) \cdots (p!q, t + 1)(q?p, t + 2)(s!r, t + 3)(r?s, t + 4) \cdots (s!r, 2t + 3)(r?s, 2t + 4)$  is in  $L_{1,1}(G_3)$  for all date  $t$ , and thus  $L_{1,1}(G_3)$  is not a regular set. While we can tweak the definition of  $L_{K,K'}(G)$  to make it a regular set of representatives for  $G$ , this set is no longer easily defined. On the other hand, the scenario-connected restriction can be more elegantly lifted directly on Theorem 13.

### 6.4 Regular Set of Representatives and existential bounds on communications

The method of regular set of representatives has been used in [11, 8] with the fixed set of *existentially bounded executions* as representatives. Indeed, if a language has a regular set of representatives, then it must be existentially  $B$  bounded for some  $B$ , that is every execution is equivalent with (is a linearization of the same MSC as) a  $B$  bounded execution (an execution where the number of messages sent and not yet received is at most  $B$ ). In particular, if  $G$  is well-formed and scenario-connected,  $L_{K,K'}(G)$  is existentially-bounded. However, the set of existentially  $B$  bounded linearizations is not regular in general, even for untimed MSC-graphs. In [11, 8], another restriction on communication is used, namely *globally cooperative* MSC-graphs [11], which require (as regularity) an exponential blow up in the size of the MSC-graph. We have shown the versatility of the technique by proposing a new set of representatives which refines the set of existentially  $B$  bounded executions, and which does not need globally cooperative MSC-graphs. We believe that this method could be used in other context than (Timed Constrained) MSC-graphs.

## 7 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 decidable for regular TC-MSC graphs. The subclass considered in this paper contains non-regular specifications. It is characterized in terms of bounds on the time a basic scenario takes, and disallows the constraint  $[0, 0]$  on transitions. We believe that these two requirements do not impair implementability, and meet what designers have in mind when designing a TC-MSC: event execution takes time, and the specification is split in phases. Several related problems remain to be tackled. Given a TC-MSC graph  $G$  and an integer  $K$ , determine whether  $G$  is  $K$ -drift-bounded. It turns out that this problem is decidable, but its proof uses very different techniques which will be dealt with in a forthcoming paper. Also, while drift boundedness is needed to obtain decidability, we doubt that positively constrained transitions are necessary.

## References

- [1] S. Akshay, M. Mukund, and K. Narayan Kumar. Checking coverage for infinite collections of timed scenarios. In *CONCUR'07*, pages 181–196, 2007.
- [2] S. Akshay, P. Gastin, K. Narayan Kumar, and M. Mukund. Model checking time-constrained scenario-based specifications. In *FSTTCS'10*, 2010.
- [3] R. Alur and D. L. Dill. A theory of timed automata. *TCS*, 126(2):183–235, 1994.
- [4] R. Alur and M. Yannakakis. Model checking of message sequence charts. In *CONCUR'99*, volume 1664 of *LNCS*, pages 114–129, 1999.
- [5] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *IFAC Symposium on System Structure and Control*, pages 469–474, 1998.
- [6] P. Bouyer, S. Haddad, and P.-A. Reynier. Timed unfoldings for networks of timed automata. In *ATVA'06*, volume 4218 of *LNCS*, 2006.
- [7] F. Cassez, Th. Chatain, and C. Jard. Symbolic unfoldings for networks of timed automata. In *ATVA'06*, volume 4218 of *LNCS*, pages 307–321, 2006.
- [8] Ph. Darondeau, B. Genest, and L. Hélouët. Products of message sequence charts. In *FOSSACS'08*, volume 4962 of *LNCS*, pages 459–474, 2008.
- [9] C. Dima and R. Lanotte. Distributed time-asynchronous automata. In *ICTAC'07*, pages 185–200, 2007.
- [10] P. Gastin, K. Narayan Kumar, and M. Mukund. Reachability and boundedness in time-constrained MSC graphs. *Perspectives in Concurrency Theory*, 2008.
- [11] 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.
- [12] 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.
- [13] ITU-TS Recommendation Z.120: Message Sequence Chart 1999 (MSC99), 1999.
- [14] 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.
- [15] A. Muscholl and D. Peled. Message sequence graphs and decision problems on mazurkiewicz traces. In *MFCS'99*, volume 1672 of *LNCS*, pages 81–91, 1999.