Reachability and Boundedness in Time-Constrained MSC Graphs*

Paul Gastin[1], **Madhavan Mukund**[2] **and K Narayan Kumar**[2]
[1]*LSV, ENS Cachan, France*
Paul.Gastin@lsv.ens-cachan.fr
[2]*Chennai Mathematical Institute, Siruseri, India*
{madhavan,kumar}@cmi.ac.in

Abstract

Channel boundedness is a necessary condition for a message-passing system to exhibit regular, finite-state behaviour at the global level. For Message Sequence Graphs (MSGs), the most basic form of High-level Message Sequence Charts (HMSCs), channel boundedness can be characterized in terms of structural conditions on the underlying graph. We consider MSGs enriched with timing constraints between events. These constraints restrict the global behaviour and can impose channel boundedness even when it is not guaranteed by the graph structure of the MSG. We show that we can use MSGs with timing constraints to simulate computations of a two-counter machine. As a consequence, even the more fundamental problem of reachability, which is trivial for untimed MSGs, becomes undecidable when we add timing constraints. Different forms of channel boundedness also then turn out to be undecidable, using reductions from the reachability problem.

Keywords: Communicating systems, message sequence charts, timed specifications

1 Introduction

In a distributed system, several agents interact to generate a global behaviour. This interaction is usually specified in terms of scenarios, using message sequence charts (MSCs) [11]. A message sequence graph (MSG) is a finite directed graph with nodes labelled by MSCs. MSGs

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

are the most basic versions of High-level Message Sequence Charts (HMSCs) [12] and are a convenient mechanism for generating possibly infinite collections of MSCs.

Communicating finite-state machines (CFMs) are a natural implementation model for message-passing systems. In recent years, there has been a considerable body of work on the analysis of message-passing systems specified in terms of MSCs and communicating finite-state machines [4, 5, 8, 9, 15, 16]. A fruitful approach is to synthesize CFMs from MSC specifications and then use standard automata-theoretic techniques for formal verification.

One essential requirement for effective synthesis of CFMs from MSCs is channel boundedness. An MSC specification is said to be universally bounded if there is a uniform upper bound on the size of all channels along any execution consistent with the specification. A specification is existentially bounded if every computation can be scheduled in at least one way so that a uniform channel bound is maintained. Algorithms to synthesize CFMs from MSC specifications were originally obtained for universally bounded specifications [9] and later extended to the existentially bounded case [8].

MSG specifications are always existentially bounded. We also have a precise characterization of universal boundedness for MSGs [14]. Interestingly, the channel boundedness problem is known to be undecidable for CFMs [6], so the limited expressiveness of MSGs with respect to CFMs is responsible for making the problem decidable in the setting of MSGs.

In this paper, we consider the analysis of message-passing systems equipped with timing constraints. The basic MSC notation does not have any provision for describing explicit real-time constraints. On the other hand, timing is an important issue in practical specifications—for instance, how long should a server wait before deciding to drop an idle connection with a client?

Time-constrained MSCs (TC-MSCs) are an extension of the MSC notation in which we can specify timing constraints between pairs of events. If we label the nodes in an MSG with TC-MSCs, we obtain a time-constrained MSG (TC-MSG). We can regard each node in a TC-MSG as one phase of a protocol. To allow us to describe timing constraints in making the transition from one phase to another, we also permit time constraints along the edges between nodes in a TC-MSG.

On the automaton front, each component in a CFM can be replaced by a timed automaton [3] yielding a natural timed extension of the CFM model. Some progress has been made in extending the analysis of MSC specifications vis-a-vis CFMs to the timed setting [2, 7].

Our focus in this paper is the boundedness problem for TC-MSGs. Since the boundedness problem is already undecidable for untimed CFMs, it is clear that it is also undecidable for timed CFMs. Somewhat surprisingly, it turns out that channel boundedness is undecidable for timed CFMs even if there is no communication loop by which a sender gets feedback from the recipient of a message [10]. However, as in the untimed case, TC-MSGs are less expressive than timed CFMs, so these undecidability results cannot be transported directly to the TC-MSG setting.

Of course, if the underlying untimed MSG is universally bounded, so will any TC-MSG derived from it by adding timing constraints. However, it is also possible that timing constraints enforce universal boundedness even if the underlying untimed MSG does not satisfy the criterion described in [14].

On the other hand, even though untimed MSG specifications are always existentially bounded, it is not difficult to construct TC-MSGs in which the timing constraints do not guarantee existential boundedness. This is because timing constraints may prevent us from choosing the schedule required in the untimed case to guarantee a uniform bound.

Our main results are negative. We show that various variants of the boundedness problem are undecidable for TC-MSGs, even when we impose severe restrictions on the manner in which timing constraints can be used. The main technique that we use to demonstrate undecidability is a simulation of two counter machines [13] using TC-MSGs.

Our simulation makes crucial use of timing contraints across the edges of a TC-MSG. We believe that the boundedness problem is decidable for TC-MSG specifications without edge constraints. We have a sufficient condition for boundedness in this case, based on an analysis of a time-constrained producer-consumer system. However, decidability of boundedness in this case remains open.

The paper is organized as follows. We begin with some preliminaries about (timed) MSCs and MSGs. Section 3 formally describes the various versions of the boundedness problem that we look at in this paper. To show that boundedness is undecidable, in general, for TC-MSGs, we first establish that reachability is undecidable, in Section 4. We then show how to reduce reachability to boundedness. In Section 6, we strengthen our undecidability results to the setting where constraints can only be described using open intervals. In the next section, we show that we can obtain undecidability even with bounded channels. Finally, in Section 8 we show that we can restrict all edge constraints to refer to a single process and still establish undecidability. In Section 9 we obtain partial results concerning the decidability

of boundedness for TC-MSGs without edge constraints. The paper concludes with a brief discussion.

2 Preliminaries

2.1 Message sequence charts

Let $\mathscr{P}$ be a finite set of processes that communicate using a finite set of message types $\mathscr{M}$ over reliable FIFO channels. For $p \in \mathscr{P}$, let $Act_p = \{p!q(m), p?q(m) \mid p \neq q \in \mathscr{P}, m \in \mathscr{M}\}$ be the set of communication actions for p. The actions $p!q(m)$ and $p?q(m)$ are read as *p sends m to q* and *p receives m from q*, respectively. Let $Act = \bigcup_{p \in \mathscr{P}} Act_p$.

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 $X \subseteq E$, $\downarrow X = \cup_{e \in X} \downarrow e$. For $p \in \mathscr{P}$ and $a \in Act$, we set $E_p = \{e \mid \lambda(e) \in Act_p\}$ and $E_a = \{e \mid \lambda(e) = a\}$, respectively.

Let $Ch = \{(p,q) \mid p \neq q\}$ denote the set of *channels*. For each $(p,q) \in Ch$, we define a relation $<_{pq}$ as follows, to capture the fact that channels are FIFO with respect to each message.

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

Finally, for each $p \in \mathscr{P}$, 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 $\mathscr{P}$) 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 $m \in \mathscr{M}$, $|E_{p!q(m)}| = |E_{q?p(m)}|$.*
3. *If $e <_{pq} e'$, then $|\downarrow e \cap \left(\bigcup_{m \in \mathscr{M}} E_{p!q(m)}\right)| = |\downarrow e' \cap \left(\bigcup_{m \in \mathscr{M}} E_{q?p(m)}\right)|$.*
4. *The partial order $\leq$ is the reflexive, transitive closure of the relation $\bigcup_{p,q \in \mathscr{P}} <_{pq}$.*

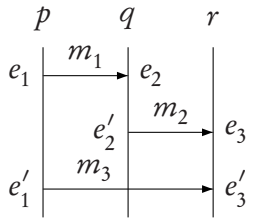

Figure 1: *An MSC*

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.

For an MSC $M = (E, \leq, \lambda)$, we let $\mathrm{lin}(M) = \{\lambda(\pi) \mid \pi \text{ is a linearization of } (E, \leq)\}$. 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. An MSC language $\mathscr{L}$ can also be seen as a word language L over Act corresponding to the linearizations of the MSCs in $\mathscr{L}$. For an MSC language $\mathscr{L}$, we set $\mathrm{lin}(\mathscr{L}) = \bigcup\{\mathrm{lin}(M) \mid M \in \mathscr{L}\}$.

Definition 2 *An MSC language $\mathscr{L}$ is said to be a* regular MSC language *if the word language* $\mathrm{lin}(\mathscr{L})$ *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 $\mathscr{L}$ is universally B-bounded if every $M \in \mathscr{L}$ is universally B-bounded. Finally, $\mathscr{L}$ is universally bounded if it is universally B-bounded for some B.

We then have the following result [9].

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

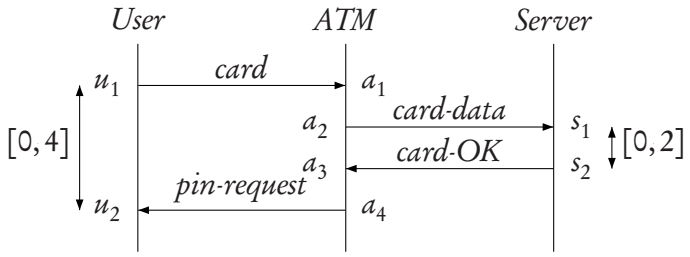

Figure 2: *A TC-MSC describing interaction with an ATM.*

A weaker notion of channel boundedness is existential boundedness. An MSC M is said to be existentially B-bounded if *some* $w \in \mathrm{lin}(M)$ is B-bounded. Existential boundedness corresponds to choosing a good schedule for the events to ensure the channel bound B. An MSC language $\mathscr{L}$ is existentially B-bounded if every $M \in \mathscr{L}$ is existentially B-bounded. Finally, $\mathscr{L}$ is existentially bounded if it is existentially B-bounded for some B.

2.2 Time-constrained MSCs

A time-constrained MSC (denoted TC-MSC) is an MSC annotated with time intervals. For simplicity, we assume that the interval bounds are natural numbers. For $a, b \in \mathbb{N}$, we allow intervals that are open (a, b), closed $[a, b]$, half-open $(a, b]$, $[a, b)$, or unbounded $[a, \infty)$, (a, ∞). As usual, by (a, b), we mean $\{x \in \mathbb{R}_{\geq 0} \mid a < x < b\}$ and so on. Let $\mathscr{I}$ denote the set of all such intervals.

Definition 4 *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 $p \in \mathscr{P}$ or $e_1 <_{pq} e_2$ for some channel $(p, q) \in Ch$ and $I \in \mathscr{I}$.*

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. Fig. 2 shows a TC-MSC describing the interaction between a user, an ATM and a server. For instance, the constraint $[0, 2]$ on (s_1, s_2) specifies that the server is expected to respond to an authentication request within 2 time units.

Definition 5 *A* time-constrained MSC (TC-MSC) *is a pair $\mathscr{T} = (M, \mathscr{C})$ where $M = (E, \leq, \lambda)$ is an MSC and $\mathscr{C} \subseteq (E \times E) \times \mathscr{I}$ is a set of interval constraints such that each pair (e_1, e_2) is mapped to at most one interval.*

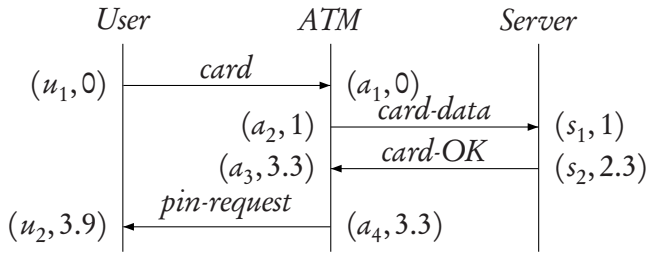

Figure 3: *A timed MSC describing interaction with an ATM.*

2.3 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 6 *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 *realizes* 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 $I \in \mathscr{I}$. We write $r \models I$ to denote that r lies in the interval specified by I.

Definition 7 *Let* $M = (E, \leq, \lambda)$ *be an MSC,* $\mathscr{T} = (M, \mathscr{CC})$ *a TC-MSC and* $M_\tau = (M, \tau)$ *a timed MSC.* M_τ *is said to* realize $\mathscr{T}$ *if for each* $\langle (e_1, e_2), I \rangle \in \mathscr{CC}$, $\tau(e_2) - \tau(e_1) \models I$.

Fig. 3 shows a timed MSC that realizes the TC-MSC in Fig. 2.

We say that a TC-MSC is *realizable* if it is realized by at least one timed MSC. Realizability amounts to checking if the constraints in a TC-MSC are feasible. This can be checked by constructing a graph corresponding to the events with weighted, directed edges in which lower bounds are represented by negative weights and upper bounds by positive weights. We can then show that the constraints in the original TC-MSC are feasible if and only if this graph has no negative-weight cycles—see, for instance, [4].

2.4 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.

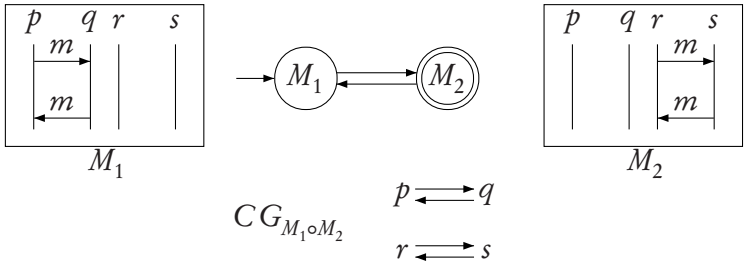

Figure 4: *A message sequence graph*

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

$$\leq = \left(\leq^1 \cup \leq^2 \cup \bigcup_{p \in \mathscr{P}} E^1_p \times E^2_p \right)^* .$$

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.

Let $\pi = q_0 \rightarrow q_1 \rightarrow \cdots \rightarrow q_n$ be a path through G. The MSC generated by π is $\Phi(\pi) = \Phi(q_0) \circ \Phi(q_1) \circ \cdots \circ \Phi(q_n)$. A path $\pi = q_0 q_1 \ldots q_n$ is a *run* if $q_0 \in Q_{in}$ and $q_n \in Q_F$. The language of MSCs accepted by G is $L(G) = \{\Phi(\pi) \mid \pi \text{ is a run through } G\}$.

An example of an MSG is depicted in Fig. 4. The initial state is marked $\rightarrow$ and the final state has a double circle. The language $\mathscr{L}$ defined by this MSG is *not* regular: $\mathscr{L}$ 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.

In general, it is undecidable whether an MSG describes a regular MSC language [9]. However, in this paper, our main focus is not regularity but channel-boundedness—is it the case that the MSC language $\mathscr{L}$ defined by an MSG G is universally B-bounded?

It is easy to see that $\mathscr{L}$ is always existentially B-bounded. Since each MSC $M \in \mathscr{L}$ is generated by a path $\pi = q_0 \rightarrow q_1 \rightarrow \cdots \rightarrow q_n$, we can decompose $M = \Phi(\pi)$ as $\Phi(q_0) \circ \Phi(q_1) \circ \cdots \circ \Phi(q_n)$. Each individual $\Phi(q_i)$ is existentially B_i bounded for some bound B_i. By scheduling events so that $\Phi(q_i)$ is completed before we start $\Phi(q_{i+1})$, we observe that M is existentially B bounded for $B = \max_{i \in \{0,1,\ldots,n\}} B_i$. Thus,

overall $\mathscr{L}$ must be existentially B_G bounded, where $B_G = \max_{q \in Q} B_q$ and B_q is the existential bound associated with $\Phi(q)$.

A necessary and sufficient condition for the MSC language of an MSG to be universally bounded is that the MSG be *locally strongly connected* [14]. To formalize this, we define the notion of a communication graph.

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

- $\mathscr{P}$ 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 *locally strongly connected* if every connected component of CG_M is strongly connected. An MSG G is said to be *locally strongly connected* if for each simple loop π in G, $\Phi(\pi)$ is locally strongly connected.

Notice that the MSC language defined by the MSG in Figure 4 is universally 1-bounded, though the language is not regular, and that the communication graph of the one simple loop in this MSG is in fact locally strongly connected.

2.5 Time-constrained MSGs

To describe infinite families of TC-MSCs, we label the nodes of an MSG with TC-MSCs instead of normal MSCs. We also permit process-wise timing constraints along the edges of the MSG. A constraint for process p along an edge $q \longrightarrow q'$ specifies a constraint between the final p-event of $\Phi(q)$ and the initial p-event of $\Phi(q')$, provided p actively participates in both these nodes. If p does not participate in either of these nodes, the constraint is ignored.

Definition 8 *A* time-constrained MSG (TC-MSG) *is a structure* $G = (Q, \rightarrow, Q_{in}, Q_F, \Phi, EdgeC)$, *where*

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

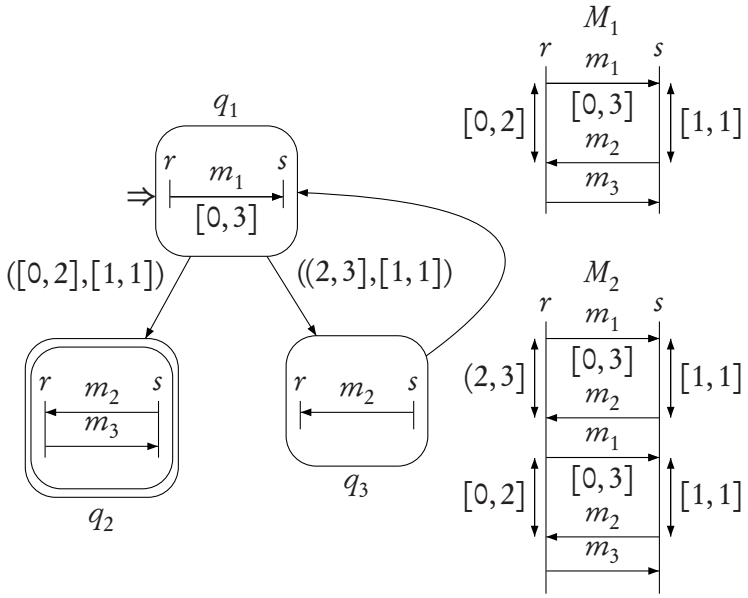

Figure 5: *A TC-MSG and some TC-MSCs that it generates*

For a path $\pi = q_0 q_1 \ldots q_n$ through G, we define $\Phi(\pi)$, the TC-MSC generated by π as follows. We begin with the TC-MSC $\Phi(q_0) \circ \Phi(q_1) \circ \cdots \circ \Phi(q_n)$. For each edge $q_i \longrightarrow q_{i+1}$, $0 \leq i < n$, if $(q_i, q_{i+1}, p, I) \in \mathit{EdgeC}$ we add a constraint I between the last p-event in $\Phi(q_i)$ and the first p-event in $\Phi(q_{i+1})$, provided p participates in both $\Phi(q_i)$ and $\Phi(q_{i+1})$. Fig. 5 shows a TC-MSG and some of the TC-MSCs that it generates.

3 Decision Problems for Time-Constrained MSGs

3.1 Channel-boundedness

The focus of this paper is to address the problem of channel-boundedness for time-constrained MSGs—that is, given a TC-MSG G, determine if it is universally or existentially bounded. Recall that the underlying untimed MSG is always existentially bounded and, if it is locally strongly connected, it is also universally bounded.

The situation for TC-MSGs is more complicated. It is possible that the timing constraints do not allow a TC-MSG to be existentially bounded. On the other hand, timing constraints may convert an MSG that is unbounded into one that is univerally bounded. We illustrate both scenarios with a simple TC-MSG modelling a producer-consumer system where one process keeps sending messages to the other, as shown in Figure 6.

In the untimed setting, this system is not universally bounded because in any MSC where k messages are sent, we can find a prefix in which all k messages are sent by P before the first message is received by C.

Proposition 9 *Consider a producer-consumer system with timing constraints as shown in Figure 6. The channel is universally bounded if and only if U is finite and either $l_P > 0$ or $l_C > 0$.*

Proof Suppose that U is finite and $l_P > 0$ and P sends a message every l_P time units starting at time 0, with each message delayed by U units, the maximum possible. We then have messages sent at times $0, l_P, 2l_P, \ldots$, which are received at times $U, U + l_P, U + 2l_P, \ldots$ respectively. In this run, $\lceil \frac{U}{l_P} \rceil$ messages are sent from time 0 to time U. After this, with each new message inserted into the channel one old message is received, so the channel never grows beyond this bound.

On the other hand suppose that U is finite and $l_C > 0$ and there are B messages in the channel at time t. All these messages must be received by C before time $t+U$. However, at most $\lceil \frac{U}{l_C} \rceil$ messages can be received by C within the interval $[t, t+U]$, so $B \leq \lceil \frac{U}{l_C} \rceil$. Since t was arbitrary, the channel is universally bounded.

Conversely, if $U = \infty$ or $l_P = l_C = 0$, we can show that the channel is unbounded. Suppose $U = \infty$ and we propose a bound B. We can delay the receipt of the first message sent by P till $B+1$ messages have been inserted into the channel. On the other hand, if U is finite but $l_P = l_C = 0$, P can send $B+1$ (in fact, any number of messages we want) within U time units and all these messages can be received by C since $l_C = 0$.

Notice also that if $U = \infty$ and $l_C > u_P$, the language of this TC-MSG is not even existentially bounded. This is in sharp contrast to untimed MSGs, which are always existentially bounded.

Variants of the problem

Our basic problem is to check whether a TC-MSG is universally and/or existentially bounded. We can identify several ways to restrict the class of TC-MSGs under consideration.

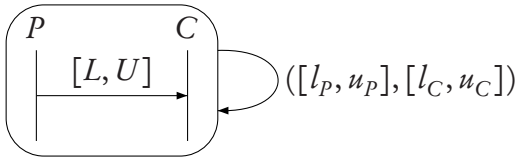

Figure 6: *Producer-Consumer with timing constraints*

Restrictions on constraints The first restriction is to do with edge constraints. In general, an edge in a TC-MSGs can have an independent constraint for each process that participates in both nodes connected by the edge. We consider two special cases:

- There are no edge constraints at all.
- Only one designated process p is permitted to have edge constraints (p is a fixed process for the entire TC-MSG).

We can also vary the type of constraints we consider. In general, as we have seen, the intervals we use in constraints can be closed, open or half open. In particular, we can have point intervals of the form $[a,a]$ which specify an exact delay. The special cases we can consider are:

- Both open and closed intervals are permitted, including point intervals.
- Only open intervals are permitted.

Restrictions on final states Normally, a TC-MSG is equipped with final states and we are only interested in paths from the initial state to one of the final states. We can drop the assumption that we have final states and consider all paths starting from an initial state.

Type of boundedness As we saw with the Producer-Consumer example, both universal and existential boundedness are nontrivial problems for TC-MSGs. The general question asks whether there exists a bound B such that the TC-MSG is existentially or universally B-bounded. We can also ask a weaker question: given a fixed bound B, is the TC-MSG existentially or universally bounded with respect to this fixed bound?

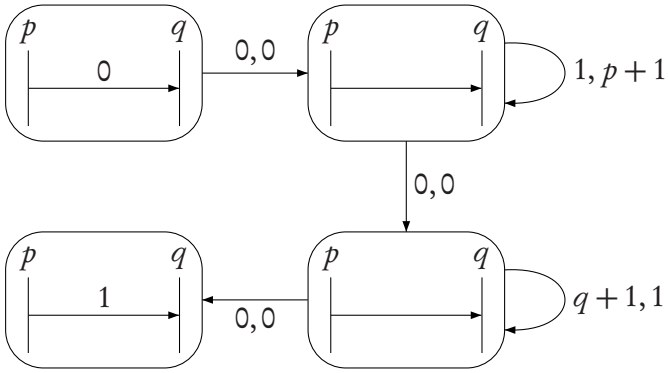

Figure 7: *Reachability in TC-MSGs is difficult*

3.2 Reachability in TC-MSGs

Let $G = (Q, \rightarrow, Q_{in}, Q_F, \Phi, EdgeC)$ be a TC-MSG. A state q is said to be *reachable* in G if there is a path $q_0 q_1 \dots q_n$ such that $q_0 \in Q_{in}$, $q_n = q$ and $\Phi(q_0) \circ \Phi(q_1) \circ \cdots \circ \Phi(q_n)$ is a realizable TC-MSC.

It turns out that even reachability is not trivial for TC-MSGs. Consider the example in Figure 7. (In this example and elsewhere, we denote point intervals $[a, a]$ by a single integer a.) To reach the last node we cannot take the shortest path and we need to iterate the first loop k times and the second loop ℓ times so that $kp - \ell q = 1$.

The negative results that we show for boundedness will, in fact, be derived from negative results for reachability—that is, we will reduce reachability to boundedness. When addressing the reachability problem, we will again consider restricted versions corresponding to the special cases on constraints, as discussed above in the context of boundedness.

4 Reachability With Edge Constraints is Undecidable

We first show that reachability is undecidable with unrestricted edge constraints. For this, we show that we can simulate the behaviour of a 2-counter machine [13]. As the name suggests, a 2-counter machine has two counters c_1 and c_2, each of which can hold a non-negative number. A program is a sequence of labelled instructions $\ell : I$, where I is one of the following:

- $c_1{+}{+}$ or $c_2{+}{+}$ which increments the value of the counter.

- `if` $c_1 \stackrel{?}{=} 0$ `goto` ℓ' `else` $c_1{-}{-}$ which transfers control to the instruction labelled ℓ' if counter c_1 is zero, and otherwise it decrements c_1 and continues with the next instruction labelled $\ell + 1$. Indeed, we also have a similar instruction for c_2.

Observe, that 2-counter machines are deterministic. Thus, a given machine will either reach its final instruction and implicitly halt after a finite number of steps or perform an infinite computation.

To simulate a 2-counter machine, we construct a TC-MSG in which each node represents one labelled instruction $\ell : I$ of the 2-counter machine. We encode each counter by a pair of processes—the counter value is represented by the difference in time between the local clock values of the pair of processes associated with the counter.

Figure 8 shows a simple simulation for one counter c. Let t_p and t_q denote the time stamp of the last event on processes p and q, respectively. The value of counter c is encoded by $t_q - t_p$. We maintain, as an invariant, that $t_p \leq t_q$. Recall that a constraint of the form b denotes a point interval $[b, b]$. Notice that this simulation does not use any time constraints on messages, but that it does use point intervals.

The initial node synchronizes p and q, thereby setting $c = t_q - t_p$ to 0. Between two nodes, we always use edge constraints $(1, 1)$ enforcing that the time difference between the last event on process p (resp. q) in the previous node and the first event on process p (resp. q) in the next node is always 1. Therefore, the node *Freeze* preserves the value of the counter. The node labelled $c{+}{+}$ delays q, thereby incrementing c. Symmetrically, the node labelled $c{-}{-}$ delays p, which decrements c. Since the last message in node $c{-}{-}$ goes from p to q, we have $c = t_q - t_p \geq 0$ at the end. So this is realizable only if counter c was positive before entering node $c{-}{-}$. The node $c \stackrel{?}{=} 0$ checks if c is 0 by sending a message back from q to p. Let t_p and t_q be the time stamps of the last events on p and q in the previous node. The message in node $c \stackrel{?}{=} 0$ is sent at time $t_q + 1$ and received at time $t_p + 1 \geq t_q + 1$. Since our invariant demands that $t_p \leq t_q$, this is realizable precisely when $t_p = t_q$, which means $c = 0$. Note that the invariant is preserved.

Having shown how to encode counter values, it is a simple matter to construct a TC-MSG that simulates a given 2-counter machine. We use two pairs of processes (p_1, q_1) and (p_2, q_2) to encode the counters c_1 and c_2, respectively. By definition of the counter machine, exactly one counter is active in each instruction, for which we use the

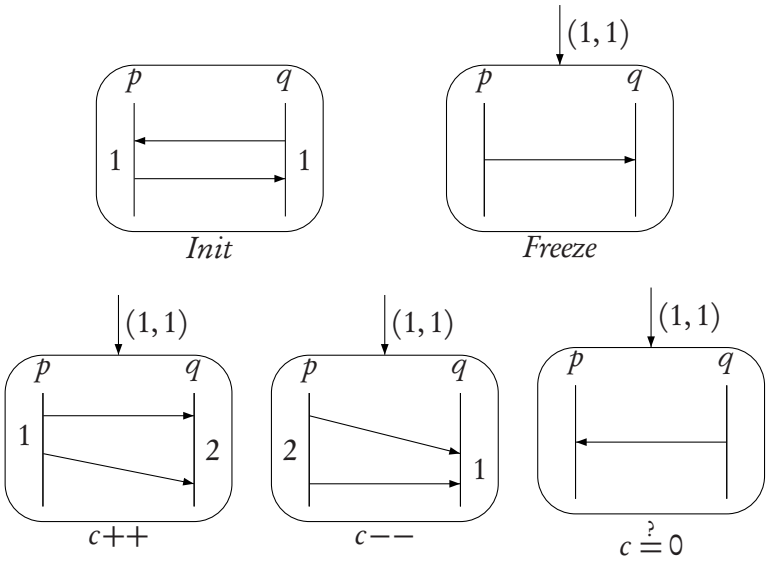

Figure 8: *Simulating a counter machine*

appropriate encoding described below. For the inactive counter, we preserve the value using the *Freeze* construction.

For each instruction $\ell : c{+}{+}$ ($c \in \{c_1, c_2\}$) of the 2-counter machine, we construct a state ℓ in the TC-MSG corresponding to $c{+}{+}$ and freezing the other counter. We connect state ℓ to the state(s) corresponding to instruction $\ell+1$ in the TC-MSG to capture the implicit control flow in the counter machine.

For each instruction ℓ : `if` $c \stackrel{?}{=} 0$ `goto` ℓ' `else` $c{-}{-}$ we create two states corresponding to $\ell_{c=0}$ and ℓ_{c--} which are labelled with the corresponding MSCs from Figure 8. Again, we freeze the inactive counter. We connect state $\ell_{c=0}$ to the state(s) corresponding to instruction ℓ' and state ℓ_{c--} to the state(s) corresponding to instruction $\ell+1$.

Let $\ell_f : I_f$ be the final (halting) instruction of the 2-counter machine. Then, checking whether the corresponding node in the TC-MSG is reachable is equivalent to checking whether the given 2-counter machine halts. Since this is an undecidable problem for 2-counter machines, we have shown the following.

Theorem 10 *The reachability problem for TC-MSGs with arbitrary edge constraints is undecidable, even without constraints on message delays.*

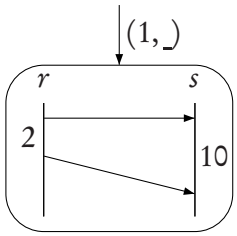

Figure 9: *Reducing reachability to boundedness*

5 Reducing Reachability to Boundedness

We add two new processes r and s to each node in our TC-MSG simulation of a 2-counter machine with two messages from r to s as shown in Figure 9. Events along r are tightly constrained by edge constraints. On the other hand, there are no edge constraints for s, events along s can be delayed arbitrarily.

There are two cases to consider.

- If the counter machine halts then the simulation is finite and channels are bounded by some B—we can calculate the bound from the length of the computation.
- If the counter machine does not halt then the simulation is infinite and r and s form a timed producer-consumer system in which the channel (r, s) is both existentially and universally unbounded, as analyzed in Section 3.1.

This reduction gives us the following result.

Theorem 11 *Both existential boundedness and universal boundedness are undecidable for TC-MSGs with arbitrary edge constraints, even without constraints on message delays.*

We note, in passing, that boundedness does not imply reachability. For instance, we can strengthen the notion of locally strongly connected TC-MSGs to obtain the class of *locally synchronized* TC-MSGs that have both bounded channels and regular behaviours [5, 9, 15]. However, reachability is still a nontrivial problem. In fact, it turns out that reachabililty is decidable for locally synchronized TC-MSGs, but this requires a somewhat sophisticated argument [2].

6 Simulating Counters With Open Intervals

We next consider the restriction where constraints can only be open intervals. Reachability remains undecidable even in this setting.

As before, we will model a counter by the difference in time across two processes p and q. However, the counter value is denoted not by $t_q - t_p$ but by $t_q - t_p - 1$. Thus, the counter is 0 when $t_q - t_p = 1$.

Our simulation of counters will use only open intervals. Instead of simulating p and q with timestamps that exactly capture the value of the counter, we use two pairs of processes (p_l, q_l) and (p_u, q_u) that serve as the *lower* and *upper* approximations of the value denoted by (p, q). We maintain as an invariant that $0 \leq t_{q_l} - t_{p_l} < t_q - t_p < t_{q_u} - t_{p_u}$.

The simulation is described in Figure 10. In the pictures, we show p and q, the processes with point intervals whose value we are trying to track, but this is only for reference. The actual simulation uses only the pairs (p_l, q_l) and (p_u, q_u)

The node *Init* sets up the invariant corresponding to $t_q - t_p = 1$, i.e., the counter value is 0.

In the exact simulation using p and q, each edge carries a constraint 1. Correponding to this, we compose nodes using edge constraints for the lower and upper approximations as shown in *Composition*. For the pair of nodes n and n' connected by such an edge, we use t to denote the times associated with the last events in n and t' to denote the times associated with the first events in n'. Then, we have:

$$\begin{array}{lcll} t'_{q_l} - t'_{p_l} & < & t_{q_l} - t_{p_l} & \text{(by edge constraints)}, \\ t_{q_l} - t_{p_l} & < & t_q - t_p & \text{(by assumption on } n\text{)}, \\ t_q - t_p & = & t'_q - t'_p & \text{(exact delay of 1 on } p \text{ and } q\text{)}, \\ t_q - t_p & < & t_{q_u} - t_{p_u} & \text{(by assumption on } n\text{)}, \\ t_{q_u} - t_{p_u} & < & t'_{q_u} - t'_{p_u} & \text{(by edge constraints)}. \end{array}$$

From this, it follows that in n', we still have $0 \leq t'_{q_l} - t'_{p_l} < t'_q - t'_p < t'_{q_u} - t'_{p_u}$ as required.

The node $c{+}{+}$ increments the counter. Once again using t' for the times of the second message and t for the times of the first one, we have

$$0 \leq t'_{q_l} - t'_{p_l} < t_{q_l} - t_{p_l} + 1 < t_q - t_p + 1 = t'_q - t'_p < t_{q_u} - t_{p_u} + 1 < t'_{q_u} - t'_{p_u},$$

so the lower and upper approximations correctly track c after the increment.

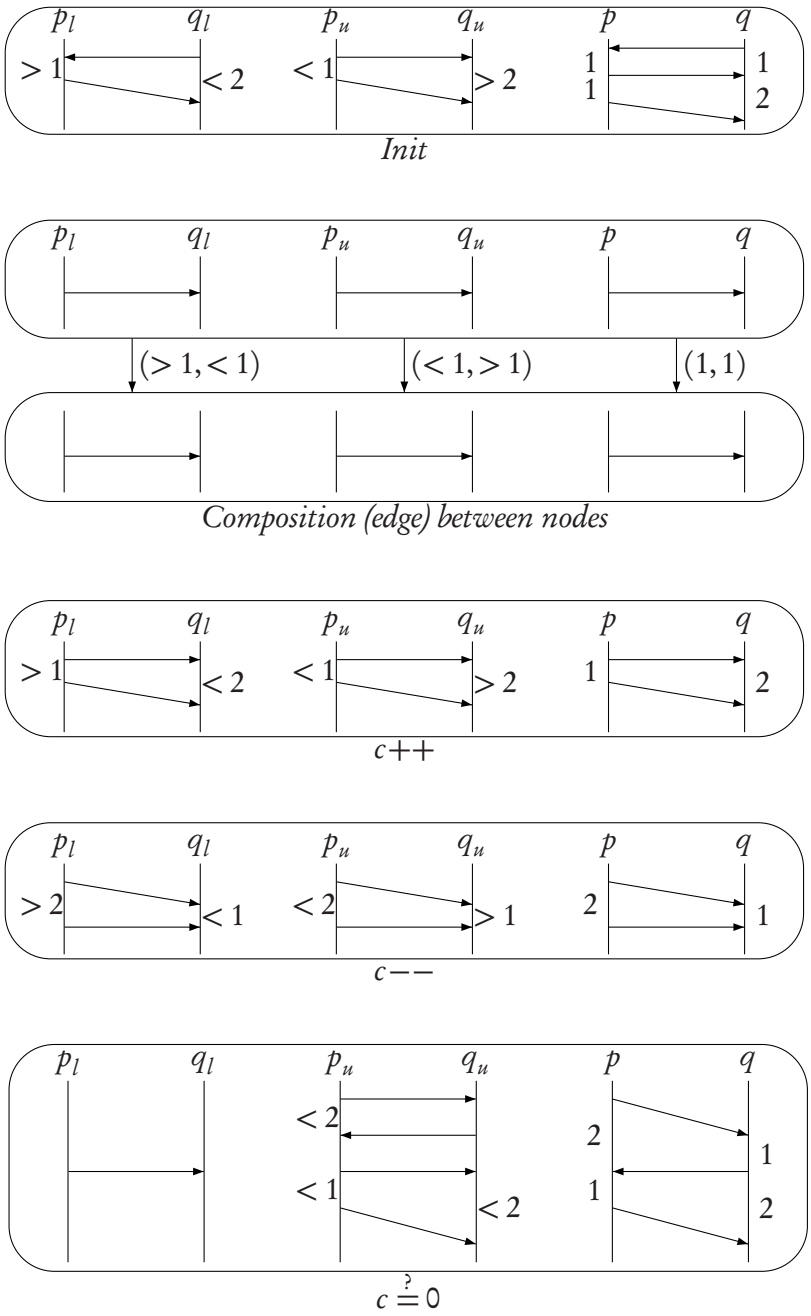

Figure 10: *Open interval simulation*

The node $c{-}{-}$ decrements the counter. Once again using t' for the times of the second message and t for the times of the first one, we have

$$0 \leq t'_{q_l} - t'_{p_l} < t_{q_l} - t_{p_l} - 1 < t_q - t_p - 1 = t'_q - t'_p < t_{q_u} - t_{p_u} - 1 < t'_{q_u} - t'_{p_u},$$

so the lower and approximations correctly track c after the increment.

As before, we *freeze* a value by sending a single message on both (p_l, q_l) and (p_u, q_u).

For $c \stackrel{?}{=} 0$, initially $1 \leq t_q - t_p < t_{q_u} - t_{p_u}$. Once again using t' for the times of the second message and t for the times of the first one, we have $t_{q_u} \leq t'_{q_u} \leq t'_{p_u} < t_{p_u} + 2$. Hence, $t_{q_u} - t_{p_u} < 2$. From this, we deduce that $1 \leq t_q - t_p < 2$, so $t_q - t_p = 1$ which means that $c = 0$.

Conversely, if $c = 0$, there is a timed MSC that realizes the path used so far with the property that $1 = t_q - t_p < t_{q_u} - t_{p_u} < 1 + \frac{1}{2}$. This means that the first two messages between p_u and q_u in the figure are realizable. The next two messages between p_u and q_u then "reset" the counter so that, at the end, $1 < t_{q_u} - t_{p_u}$.

We can now use two sets of lower and upper approximations to track two counters and set up a TC-MSG that simulates a 2-counter machine as in Section 4. It is a simple matter to add a suitably modified version of the MSC shown in Figure 9 to each node, so that we have the following results.

Theorem 12 *Reachability, existential boundedness and universal boundedness are all undecidable for TC-MSGs even if we restrict all constraints to open intervals and without constraints on message delays.*

7 Counter Simulation With Bounded Channels

We can use a more sophisticated construction to simulate a counter in which all channels are 1-bounded. The main ingredients are shown in Figure 11. The counter value c is encoded as the difference $t_q - t_p$. This quantity is manipulated by sending messages on the channels (p, p') and (q, q'), with timing constraints on the send and receive events along processes (p, p') and (q, q').

The main complication is that the construction for $c{-}{-}$ does not prevent the value in c from going below 0. Note however that the TC-MSC for $c \stackrel{?}{=} 0$ can be realized only when $t_q - t_p = 0$, i.e., it accurately checks that $c = 0$ without assuming that the counter is non-negative. We use this fact to implement a decrementation which "terminates"

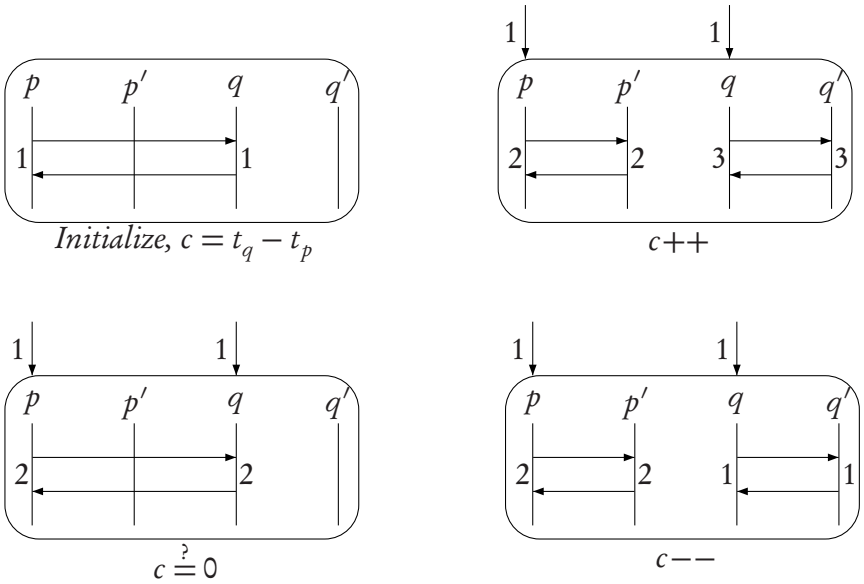

Figure 11: *Counter simulation with bounded channels*

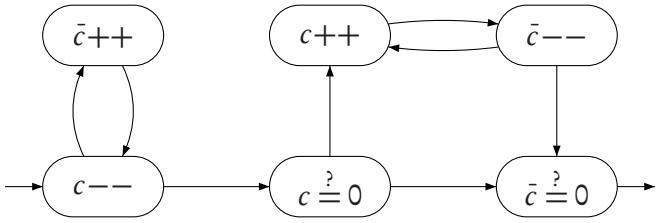

Figure 12: *Decrementation without going below* 0

only if the counter stay above 0. We copy the contents of c to another counter $\bar{c}$ by repeatedly decrementing c and incrementing $\bar{c}$, allowing the loop to terminate when $c \stackrel{?}{=} 0$. In this case, we transfer the contents back from $\bar{c}$ to c using the same trick. The precise construction is depicted in Figure 12. If $c < 0$ after the original decrement, we have a livelock in which we cycle through the first loop an infinite number of times, thus effectively blocking the simulation. So the simulation (may) terminates only if the counter is non-negative after the initial decrement.

For *Freeze*, we exchange a pair of messages between p, p' and between q, q' with the same delay along each process. So the node is similar to that of c++ where we replace 3 by 2.

With these ingredients in place, we can once again simulate a 2-counter machine with a TC-MSG using six pairs of processes,

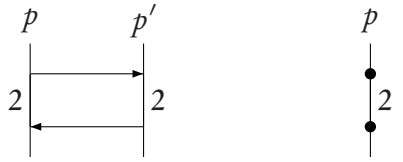

Figure 13: *Messages can be eliminated*

$\{(p_1, p_1'), (q_1, q_1'), (p_2, p_2'), (q_2, q_2'), (p, p'), (q, q')\}$, to encode the three counters c_1, c_2 and $\bar{c}$. The main difference with our earlier construction is that even when the run of the 2-counter machine is infinite, the corresponding TC-MSC traced out by our TC-MSG is 1-bounded.

The fact that this simulation of a 2-counter machine has 1-bounded channels allows us to sharpen our undecidability result for boundedness. In the earlier reduction from reachability to boundedness, we had included the MSC in Figure 9 within each node of the 2-counter simulation. Here, instead, we create a single separate node containing this TC-MSC, with a self-loop. We then add an edge from the node in our TC-MSG corresponding to the final instruction $\ell_f : I_f$ of the 2-counter machine to this new node.

Clearly, the new node with the TC-MSC from Figure 9 is reachable if and only if the 2-counter machine that we are simulating terminates at $\ell_f : I_f$. Once we enter this new node, we start generating a TC-MSC that is not existentially or universally bounded for *any* choice of B. On the other hand, if the 2-counter machine computation never terminates, we always remain within the 1-bounded portion of the simulation. Thus, if we can decide whether this TC-MSG is B-bounded, either existentially or universally, for any $B \geq 1$, we can also decide whether the 2-counter machine halts. Hence, we have the following result.

Theorem 13 *Checking whether a TC-MSG with arbitrary edge constraints is existentially or universally bounded with respect to a fixed bound B is undecidable for every $B \geq 1$.*

In fact, the construction can be further simplified if we permit internal events along processes. We can then replace each exchange of a pair of messages along the channel (p, p') by two internal events on p with the same interval constraint, as shown in Figure 13. But we still need to exchange messages between p and q to initialize the counter or to check $c \stackrel{?}{=} 0$. Thus, we can divide the number of processes by 2 in this simulation.

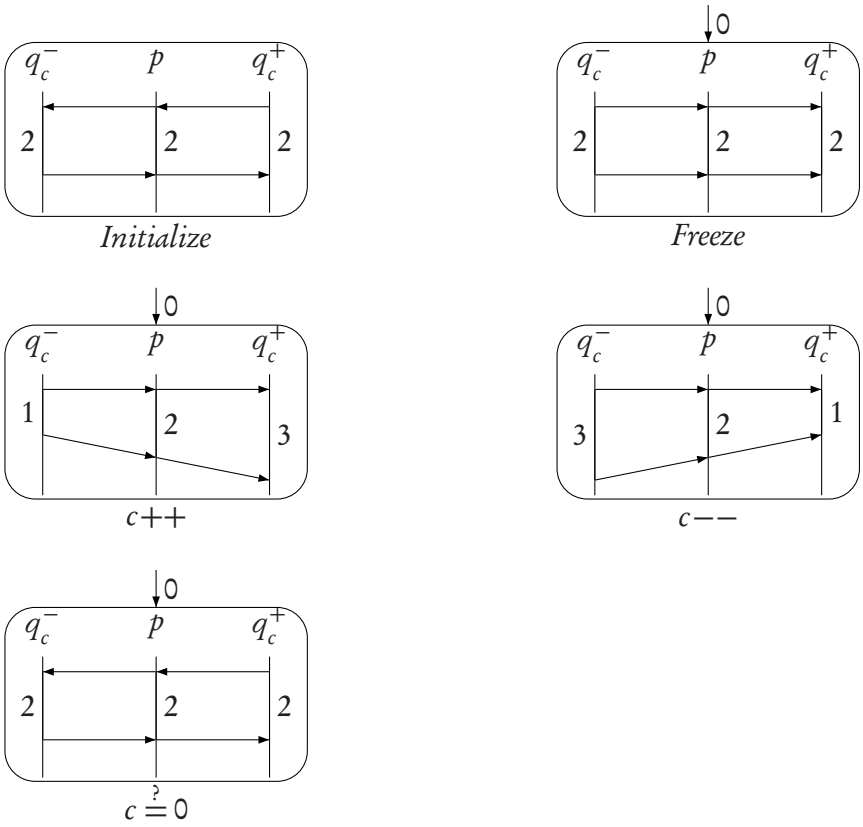

Figure 14: *Simulation with constraints along a single process*

8 Edge Constraints Along a Single Process

The last case we consider is when edge constraints are only permitted for a fixed process. This is a natural restriction — for instance this process could be a controller that coordinates between the different phases of a protocol. Let us denote this special process p. To simulate a counter c, we add two processes q_c^+ and q_c^-. Let t_c^+, t_c^- and t denote the time of the last event along processes q_c^+, q_c^- and p, respectively. We then maintain the value of c in terms of an upper approximation $c^+ = t_c^+ - t$ and a lower approximation $c^- = t - t_c^-$, so that we always have $c^+ \geq c \geq c^- \geq 0$.

Figure 14 shows the basic constructs needed for the simulation. As usual, the node labelled *Init* ensures that $t_c^+ = t = t_c^-$, so initially we have $c^+ = c^- = 0$. The node labelled *Freeze* preserves the current value of c. The nodes labelled $c{+}{+}$ and $c{-}{-}$ correspond to in-

crementing and decrementing the value of c, respectively. The node labelled $c \stackrel{?}{=} 0$ check if the value of c is 0.

Any sequence $\sigma = s_1 s_2 \dots s_n$ of states where each s_j is one among $c{+}{+}$, $c{-}{-}$, *Freeze* and $c \stackrel{?}{=} 0$ yields a TC-MSC M_σ. We say that σ describes a *valid* computation if when executing the corresponding sequence of instructions the counter c never goes below 0 and whenever a zero-test is performed then the value of c is indeed 0. Note that, if σ describes a valid computation then M_σ is realizable by a T-MSC such that after each instruction we have $c^+ = c = c^- \geq 0$.

Conversely, to justify that c^+ and c^- track the value of c accurately, we show that any T-MSC that realizes M_σ maintain the invariant $c^+ \geq c \geq c^- \geq 0$. Let us denote $|\sigma|_a$ the number of occurrences of the letter a in the sequence σ so that $c = |\sigma|_{c++} - |\sigma|_{c--}$. We verify that the invariant holds with the following calculation.

$$\begin{aligned}
c^+ &= t_c^+ - t \\
&\geq 3|\sigma|_{c++} + 2|\sigma|_{c\stackrel{?}{=}0} + 2|\sigma|_{Freeze} + |\sigma|_{c--} - \\
&\quad 2(|\sigma|_{c++} + |\sigma|_{c\stackrel{?}{=}0} + |\sigma|_{Freeze} + |\sigma|_{c--}) \\
&= |\sigma|_{c++} - |\sigma|_{c--} = c
\end{aligned}$$

$$\begin{aligned}
c^- &= t - t_c^- \\
&\leq 2(|\sigma|_{c++} + |\sigma|_{c\stackrel{?}{=}0} + |\sigma|_{Freeze} + |\sigma|_{c--}) - \\
&\quad -(3|\sigma|_{c--} + 2|\sigma|_{c\stackrel{?}{=}0} + 2|\sigma|_{Freeze} + |\sigma|_{c++}) \\
&= |\sigma|_{c++} - |\sigma|_{c--} = c
\end{aligned}$$

Note that we always have $c^- = t - t_c^- \geq 0$ since the last message of each node always goes from q_c^- to p. Hence, if M_σ is realizable then $c \geq c^- \geq 0$ and the counter never goes below 0 during the computation.

Moreover, the realization of the TC-MSC in $c \stackrel{?}{=} 0$ implies $t_c^+ = t = t_c^-$ at the end, i.e., $c^+ = c^- = 0$. Using the invariant, we deduce that the counter must be 0 whenever we use state $c \stackrel{?}{=} 0$ in the sequence σ.

The simulation of a 2-counter machine with a TC-MSG follows along the usual lines. In each MSC of our simulation, we add a copy of the TC-MSC from Figure 9. Since edge-constraints are permitted only on p, we use p to play the role of r and introduce s as a fresh process. Then, from our argument, the channel (p, s) will be bounded if and only if the 2-counter machine that we are simulating has a halting computation. This yields the following result.

Theorem 14 *Reachability, existential boundedness and universal boundedness are all undecidable for TC-MSGs even if all edge constraints in the TC-MSG are restricted to a fixed process and without constraints on message delays.*

9 Decidability Without Edge Constraints

If we have no edge constraints in a TC-MSG, reachability is decidable and the status of boundedness is still open.

9.1 Reachability

Since there are no edge constraints, for a path $\pi = q_0 q_1 \dots q_n$, we can always choose to execute the events in $\Phi(\pi) = \Phi(q_0) \circ \Phi(q_1) \circ \cdots \circ \Phi(q_n)$ one node at a time, as in an untimed MSG. The only difficulty that can arise is that a TC-MSC labelling a node is not realizable—that is, the constraints within it are not feasible. We can check this easily, as explained in Section 2.3, and delete nodes that are labelled by unrealizable TC-MSCs. Reachabilty then amounts to finding a path in the reduced graph after eliminating infeasible nodes.

9.2 Boundedness

When there are no constraints on edges, one strategy is to extend the producer-consumer analysis in Section 3 and check universal and existential boundedness. Recall that an MSG in which every node is labelled by a nonempty MSC is universally bounded if for every simple loop, the communication graph of the MSC described by the loop is locally strongly connected. Given this, it suffices to concentrate on simple loops where the communication graph of the underlying MC is not locally strongly connected.

Fix a simple loop π whose underlying MSC is $M = \Phi(\pi)$ such that CG_M is not locally strongly connected. Let (p, q) be a pair of processes such that there is an edge from p to q in CG_M, but no path back from q to p. If we iterate M, p and q play the role of P and C from the producer-consumer system described in Section 3. This system can be completely analyzed if we know the constraints on messages from p to q as well as between successive send events on p and receive events on q. The only difficulty is that the constraints along p and q may arise due to the transitive closure of dependencies through other pairs of events.

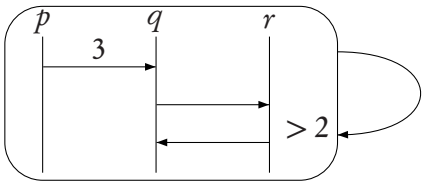

Figure 15: *Transitive dependency*

For example, in the system shown in Figure 15, the channel (p,q) is apparently unbounded since there is no explicit lower bound on consecutive sends by p or consecutive receives by q, though there is an upper bound on message delays along (p,q). However, the lower bound between the two events along r implicitly enforces a lower bound between the two corresponding events along q and hence imposes a lower bound between the receipt of consecutive messages by q along the channel (p,q). Thus, this loop satisifies the condition specified in Proposition 9 for the channel (p,q) to be universally bounded.

Unfolding the loop π several times may "reveal" timing constraints between two occurrences of the same send event $p!q(m)$ in the loop, or two occurrences of the same receive event $q?p(m)$. These timing constraints may be induced by constraints in the SCC of p or q respectively. Doing so, we may obtain for some message from p to q induced constraints $[l_p, u_p]$ and $[l_q, u_q]$ for successive send and receive events, respectively. Considering in addition the bounds $[L,U]$ carried by this message, we may apply Proposition 9 to prove that this loop is universally bounded if U is finite and $l_p > 0$ or $l_q > 0$.

Unfortunately, we do not obtain a necessary and sufficient condition following this approach. Consider the system in Figure 16. There is a single loop whose communication graph is not strongly connected. If we focus on the channel (p,r), we can regard this as a producer consumer system. In this system, we cannot derive an upper bound for the message from p to r sent in the third node. Nevertheless, a careful analysis shows that the channel (p,r) is indeed bounded by $\frac{2+U_1+U_2}{\ell}$.

Thus, we have a sufficient condition to ensure that a TC-MSG without edge constraints is bounded, but not a necessary one.

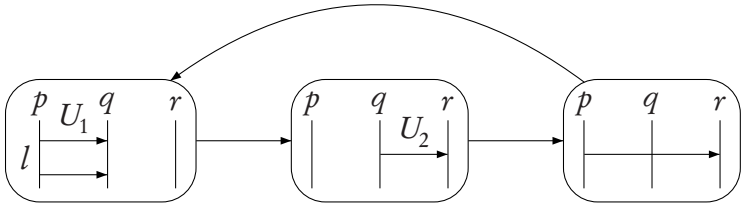

Figure 16: *Boundedness without edge constraints*

10 Discussion

We have shown that reachability and boundedness are, in general, undecidable for TC-MSGs. This is in contrast to the positive results for both questions in the untimed setting, confirming the suspicion that adding real-time constraints to specifications of distributed systems is tricky. As we have shown, even the simplest form of timing constraints across edges of a TC-MSG induce undecidability. Without edge constraints, reachability becomes decidable for TC-MSGs. We conjecture that boundedness is also decidable for this class of TC-MSGs.

The main reason why timed distributed systems are difficult to analyze is that global time acts as a covert channel to convey information across processes. This observation has also been made in the context of timed CFMs [10]. One way to get around this problem is to allow each component's clock to drift with respect to other components. The difficulty is to describe a model that permits this without making the notion of time across components meaningless. A model of timed CFMs with independently evolving clocks has been introduced in [1]. It remains to be seen if similar ideas can be incorporated into the TC-MSG framework.

References

[1] S. Akshay, B. Bollig, P. Gastin, M. Mukund and K. Narayan Kumar: Distributed Timed Automata with Independently Evolving Clocks. *Proc. CONCUR 2008*, Springer LNCS **5201** (2008) 82–97.

[2] S. Akshay, M. Mukund and K. Narayan Kumar: Checking Coverage for Infinite Collections of Timed Scenarios *Proc. CONCUR 2007*, Sringer LNCS **4703** (2007) 181–196.

[3] R. Alur and D. Dill: A Theory of Timed Automata. *Theor. Comput. Sci.*, **126** (1994) 183–225.

[4] R. Alur, G. Holzmann and D. Peled: An analyzer for message sequence charts. *Software Concepts and Tools*, **17(2)** (1996) 70–77.

[5] R. Alur and M. Yannakakis: Model checking of message sequence charts. *Proc. CONCUR'99*, Springer Lecture Notes in Computer Science **1664**, (1999) 114–129

[6] D. Brand and P. Zafiropulo: On communicating finite-state machines. *J. ACM*, **30(2)** (1983) 323–342.
[7] P. Chandrasekaran and M. Mukund: Matching Scenarios with Timing Constraints. *Proc. FORMATS 2006*, Springer LNCS 4202 (2006) 98–112.
[8] B. Genest, D. Kuske and A. Muscholl: A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inf. Comput.* **204(6)** (2006) 920–956.
[9] J.G. Henriksen, M. Mukund, K. Narayan Kumar, M. Sohoni and P.S. Thiagarajan: A Theory of Regular MSC Languages. *Inf. Comput.*, **202(1)** (2005) 1–38.
[10] P. Krcál and W. Yi: Communicating Timed Automata: The More Synchronous, the More Difficult to Verify. *Proc. CAV 2006*, Springer LNCS 4144 (2006) 249–262.
[11] ITU-T Recommendation Z.120: *Message Sequence Chart (MSC)*. ITU, Geneva (1999).
[12] S. Mauw and M.A. Reniers: High-level message sequence charts. *Proc. SDL'97*, Elsevier (1997) 291–306.
[13] M. Minsky: *Computation: finite and infinite machines*. Prentice-Hall (1967).
[14] R. Morin: On Regular Message Sequence Chart Languages and Relationships to Mazurkiewicz Trace Theory. *FoSSaCS 2001*, Springer LNCS **2030** (2001) 332–346.
[15] 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
[16] A. Muscholl, D. Peled, and Z. Su: Deciding properties for message sequence charts. *Proc. FOSSACS'98*, LNCS **1378**, Springer-Verlag (1998) 226–242.