Automata, Languages and Programming, 27th International Colloquium
Proceedings: Ugo Montanari, José D. P. Rolim, Emo Welzl(eds.)
Springer Lecture Notes in Computer Science **1853** (2000), 675–686.

On Message Sequence Graphs and Finitely Generated Regular MSC Languages

Jesper G. Henriksen[1], Madhavan Mukund[2],
K. Narayan Kumar[2], and P.S. Thiagarajan[2]

[1] BRICS, University of Aarhus, Denmark.
E-mail: gulmann@brics.dk
[2] Chennai Mathematical Institute, Chennai, India.
E-mail: {madhavan,kumar,pst}@smi.ernet.in

Abstract. Message Sequence Charts (MSCs) are an attractive visual formalism widely used to capture system requirements during the early design stages in domains such as telecommunication software. A standard method to describe multiple communication scenarios is to use message sequence graphs (MSGs). A message sequence graph allows the protocol designer to write a finite specification which combines MSCs using basic operations such as branching choice, composition and iteration. The MSC languages described by MSGs are not necessarily regular in the sense of [HM+99]. We characterize here the class of regular MSC languages that are MSG-definable in terms of a notion called finitely generated MSC languages. We show that a regular MSC language is MSG-definable if and only if it is finitely generated. In fact we show that the subclass of "bounded" MSGs defined in [AY99] exactly capture the class of finitely generated regular MSC languages.

1 Introduction

Message sequence charts (MSCs) are an appealing visual formalism often used to capture system requirements in the early design stages. They are particularly suited for describing scenarios for distributed telecommunication software [RGG96,ITU97]. They also appear in the literature as timing sequence diagrams, message flow diagrams and object interaction diagrams and are used in a number of software engineering methodologies [BJR97,HG97,RGG96]. In its basic form, an MSC depicts a single partially-ordered execution of a distributed system which just describes the exchange of messages between the processes of the system. A collection of MSCs is used to capture the scenarios that a designer might want the system to exhibit (or avoid).

Message Sequence Graphs (MSGs) are a nice mechanism for defining collections of MSCs. An MSG is a finite directed graph with a designated initial vertex and terminal vertex in which each node is labelled by an MSC and the edges represent a natural concatenation operation on MSCs. The collection of MSCs defined by an MSG consists of all those MSCs obtained by tracing a path in the MSG from the initial vertex to the terminal vertex and concatenating the MSCs that are encountered along the path. It is easy to see that this way of defining a collection of MSCs extends smoothly to the case where there are multiple terminal nodes. Throughout what follows we shall assume this extended notion of an MSG (that is, with multiple terminal nodes). For ease of presentation, we shall also not deal with the so called hierarchical MSGs [AY99].

Intuitively, not every MSG-definable collection of MSCs can be realized as a finite-state device. To formalize this idea we have introduced a notion of a *regular* collection of MSCs and studied its basic properties [HM+99]. Our notion of regularity is independent of the notion of MSGs.

Our main goal in this paper is to pin down the regular MSC languages that can be defined using MSGs. We introduce the notion of an MSC language being *finitely generated*. From our results, which we detail below, it follows that a regular MSC language is MSG-definable if and only if it is finitely generated. In fact we establish the following results.

As already mentioned, not every MSG defines a regular MSC language. Alur and Yannakakis have identified a syntactic property called *boundedness* and shown that the set of all linearizations of the MSCs defined by a bounded MSG is a regular string language over an appropriate alphabet of events. It then follows easily that, in the present setting, every bounded MSG defines a finitely generated regular MSC language. One of our main results here is that the converse is also true, namely, every finitely generated regular MSC language can be defined by a bounded MSG. Since every MSG (bounded or otherwise) defines only a finitely generated MSC language, it follows that a regular MSC language is finitely generated if and only if it is MSG-definable and, in fact, if and only if it is bounded MSG-definable.

Since the class of regular MSC languages strictly includes the class of finitely generated regular MSC languages, one could ask when a regular MSC language is finitely generated. We show that this question is decidable. Finally, one can also ask whether a given MSG defines a regular MSC language (and is hence "equivalent" to a bounded MSG). We show that this decision problem is undecidable.

Turning briefly to related literature, a number of studies are available which are concerned with individual MSCs in terms of their semantics and properties [AHP96,LL95]. A variety of algorithms have been developed for MSGs in the literature—for instance, pattern matching [LP97,Mus99,MPS98], detection of process divergence and non-local choice [BL97], and confluence and race conditions [MP99]. A systematic account of the various model-checking problems associated with MSGs and their complexities can be found in [AY99]. Finally, many of our proof techniques make use of results from the theory of Mazurkiewicz traces [DR95].

In the next section we introduce MSCs and regular MSC languages. We then introduce message sequence graphs in Section 3. In Section 4 we define finitely generated MSC languages and provide an effective procedure to decide whether a regular MSC language is finitely generated. Our main result that the class of finitely generated regular MSC languages coincides with the class of bounded

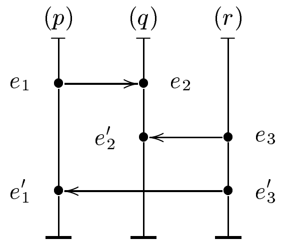

Fig. 1. An example MSC over $\{p, q, r\}$.

MSG-definable languages is then established. Finally, we sketch why the problem of determining if an MSG defines a regular MSC language is undecidable. Due to lack of space, we give here only the main technical constructions and sketches of proofs. All the details are available in [HM+99].

2 Regular MSC Languages

We fix a finite set of processes (or agents) $\mathcal{P}$ and let p, q, r range over $\mathcal{P}$. For each $p \in \mathcal{P}$ we define $\Sigma_p = \{p!q \mid p \neq q\} \cup \{p?q \mid p \neq q\}$ to be the set of communication actions in which p participates. The action $p!q$ is to be read as *p sends to q* and the action $p?q$ is to be read as *p receives from q*. At our level of abstraction, we shall not be concerned with the actual messages that are sent and received. We will also not deal with the internal actions of the agents. We set $\Sigma = \bigcup_{p \in \mathcal{P}} \Sigma_p$ and let a, b range over Σ. We also denote the set of *channels* by $Ch = \{(p,q) \mid p \neq q\}$ and let c, d range over Ch.

A Σ-labelled poset is a structure $M = (E, \leq, \lambda)$ where $(E, \leq)$ is a poset and $\lambda : E \to \Sigma$ is a labelling function. For $e \in E$ we define $\downarrow e = \{e' \mid e' \leq e\}$. For $p \in \mathcal{P}$ and $a \in \Sigma$, we set $E_p = \{e \mid \lambda(e) \in \Sigma_p\}$ and $E_a = \{e \mid \lambda(e) = a\}$, respectively. For each $c \in Ch$, we define the *communication relation* $R_c = \{(e, e') \mid \lambda(e) = p!q, \lambda(e') = q?p$ and $|\downarrow e \cap E_{p!q}| = |\downarrow e' \cap E_{q?p}|\}$. Finally, for each $p \in \mathcal{P}$, we define the *p-causality relation* $R_p = (E_p \times E_p) \cap \leq$.

An MSC (over $\mathcal{P}$) is a *finite* Σ-labelled poset $M = (E, \leq, \lambda)$ which satisfies the following conditions:

(i) Each R_p is a linear order.
(ii) If $p \neq q$ then $|E_{p!q}| = |E_{q?p}|$.
(iii) $\leq = (R_{\mathcal{P}} \cup R_{Ch})^*$ where $R_{\mathcal{P}} = \bigcup_{p \in \mathcal{P}} R_p$ and $R_{Ch} = \bigcup_{c \in Ch} R_c$.

In diagrams, the events of an MSC are presented in *visual order*. The events of each process are arranged in a vertical line and the members of the relation R_{Ch} are displayed as horizontal or downward-sloping directed edges. We illustrate the idea with an example, shown in Figure 1.

Here $\mathcal{P} = \{p, q, r\}$. For $x \in \mathcal{P}$, the events in E_x are arranged along the line labelled (x) with earlier (relative to $\leq$) events appearing above the later events. The R_{Ch}-edges across agents are depicted by horizontal edges—for instance $e_3\ R_{(r,q)}\ e'_2$. The labelling function λ is easy to extract from the diagram—for example, $\lambda(e'_3) = r!p$ and $\lambda(e_2) = q?p$.

We define regular MSC languages in terms of their linearizations. For the MSC $M = (E, \leq, \lambda)$, let $Lin(M) = \{\lambda(\pi) \mid \pi \text{ is a linearization of } (E, \leq)\}$. By abuse of notation, we have used λ to also denote the natural extension of λ to E^*. The string $p!q\ r!q\ q?p\ q?r\ r!p\ p?r$ is a linearization of the MSC in Figure 1.

We say that $\sigma \in \Sigma^*$ is *proper* if for every prefix τ of σ and every pair (p, q) of processes, $|\tau|_{p!q} \geq |\tau|_{q?p}$. We say that σ is *complete* if σ is proper and $|\sigma|_{p!q} = |\sigma|_{q?p}$ for every pair of processes (p, q). Clearly, any linearization of an MSC is a complete. Conversely, every complete sequence is the linearization of some MSC.

Henceforth, we identify an MSC with its isomorphism class. We let $\mathcal{M}_\mathcal{P}$ be the set of MSCs over $\mathcal{P}$. An MSC language $\mathcal{L} \subseteq \mathcal{M}_\mathcal{P}$ is said to *regular* if $\bigcup\{Lin(M) \mid M \in \mathcal{L}\}$ is a regular subset of Σ^*. We note that the entire set $\mathcal{M}_\mathcal{P}$ is not regular by this definition.

We define $L \subseteq \Sigma^*$ to be a *regular string MSC language* if there exists a regular MSC language $\mathcal{L} \subseteq \mathcal{M}_\mathcal{P}$ such that $L = \bigcup\{Lin(M) \mid M \in \mathcal{L}\}$. As shown in [HM+99], regular MSC languages and regular string MSC languages represent each other. Hence, abusing terminology, we will write "regular MSC language" to mean "regular string MSC language". From the context, it should be clear whether we are working with MSCs from $\mathcal{M}_\mathcal{P}$ or complete words over Σ^*.

Given a regular subset $L \subseteq \Sigma^*$, we can decide whether L is a regular MSC language. We say that a state s in a finite-state automaton is *live* if there is a path from s to a final state. Let $\mathcal{A} = (S, \Sigma, s_{in}, \delta, F)$ be the minimal DFA representing L. Then it is not difficult to see that L is a regular MSC language if and only if we can associate with each live state $s \in S$, a (unique) channel-capacity function $\mathcal{K}_s : Ch \to \mathbb{N}$ which satisfies the following conditions.

(i) If $s \in \{s_{in}\} \cup F$ then $\mathcal{K}_s(c) = 0$ for every $c \in Ch$.
(ii) If s, s' are live states and $\delta(s, p!q) = s'$ then $\mathcal{K}_{s'}((p, q)) = \mathcal{K}_s((p, q))+1$ and $\mathcal{K}_{s'}(c) = \mathcal{K}_s(c)$ for every $c \neq (p, q)$.
(iii) If s, s' are live states and $\delta(s, q?p) = s'$ then $\mathcal{K}_s((p, q)) > 0$, $\mathcal{K}_{s'}((p, q)) = \mathcal{K}_s((p, q))-1$ and $\mathcal{K}_{s'}(c) = \mathcal{K}_s(c)$ for every $c \neq (p, q)$.
(iv) Suppose $\delta(s, a) = s_1$ and $\delta(s_1, b) = s_2$ with $a \in \Sigma_p$ and $b \in \Sigma_q$, $p \neq q$. If $(a, b) \notin Com$ or $\mathcal{K}_s((p, q)) > 0$, there exists s'_1 such that $\delta(s, b) = s'_1$ and $\delta(s'_1, a) = s_2$. (Here and elsewhere $Com = \{(p!q, q?p) \mid p \neq q\}$.)

In the minimal DFA $\mathcal{A}$ representing a regular MSC language, if s is a live state and $a, b \in \Sigma$ then we say that a and b are *independent at s* if $(a, b) \in Com$ implies $\mathcal{K}_s((p, q)) > 0$ where $\mathcal{K}$ is the unique channel-capacity function associated with $\mathcal{A}$ and $a = p!q$ and $b = q?p$.

We conclude this section by introducing the notion of B-bounded MSC languages. Let $B \in \mathbb{N}$ be a natural number. We say that a word σ in Σ^* is B-bounded if for each prefix τ of σ and for each channel $(p, q) \in Ch$, $|\tau|_{p!q} - |\tau|_{q?p} \leq B$. We say that $L \subseteq \Sigma^*$ is B-bounded if every word $\sigma \in L$ is B-bounded. It is not difficult to show:

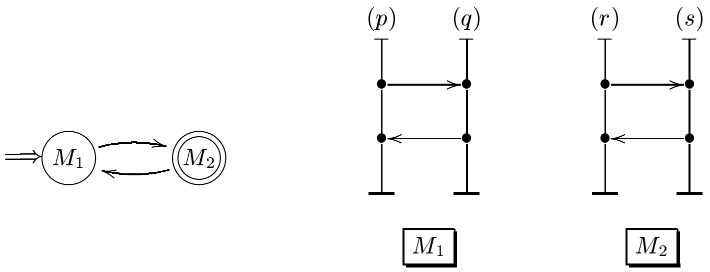

Fig. 2. An example MSG.

Proposition 2.1. *Let L be a regular MSC language. There is a bound $B \in \mathbb{N}$ such that L is B-bounded.*

3 Message Sequence Graphs

An MSG allows a protocol designer to write in a standard way [ITU97], a finite specification which combines MSCs using operations such as branching choice, composition and iteration. Each node is labelled by an MSC and the edges represent the natural operation of MSC concatenation.

To bring out this concatenation operation, we let $M_1 = (E_1, \leq_1, \lambda_1)$ and $M_2 = (E_2, \leq_2, \lambda_2)$ be a pair for MSCs such that E_1 and E_2 are disjoint. For $i \in \{1,2\}$, let R_c^i and $\{R_p^i\}_{p\in\mathcal{P}}$ denote the underlying communication and process causality relations in M_i. The *(asynchronous) concatenation* of M_1 and M_2, denoted $M_1 \circ M_2$, is the MSC $(E, \leq, \lambda)$ where $E = E_1 \cup E_2$, $\lambda(e) = \lambda_i(e)$ if $e \in E_i$, $i \in \{1,2\}$, and $\leq\, = (R_{\mathcal{P}} \cup R_{Ch})^*$, where $R_p = R_p^1 \cup R_p^2 \cup \{(e_1, e_2) \mid e_1 \in E_1, e_2 \in E_2, \lambda(e_1) \in \Sigma_p, \lambda(e_2) \in \Sigma_p\}$ for $p \in \mathcal{P}$, and $R_c = R_c^1 \cup R_c^2$ for $c \in Com$.

A *Message Sequence Graph (MSG)* is a structure $\mathcal{G} = (Q, \rightarrow, Q_{in}, F, \Phi)$, 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.
- $F \subseteq Q$ is a set of final states.
- $\Phi : Q \rightarrow \mathcal{M}_{\mathcal{P}}$ is a (state-)labelling function.

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

An example of an MSG is depicted in Figure 2. It's not hard to see that the MSC language $\mathcal{L}$ defined is *not* regular in the sense defined in Section 2. To see this, we note that $\mathcal{L}$ projected to $\{p!q, r!s\}$ is not a regular string language.

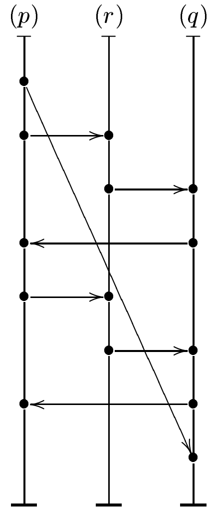

Fig. 3. The atomic MSC M_2 of the non-finitely-generated language $\mathcal{L}_{inf}$.

4 Bounded MSGs and Finitely Generated MSC Languages

Let $\mathcal{X}_1, \mathcal{X}_2 \subseteq \mathcal{M}_\mathcal{P}$ be two sets of MSCs. As usual, $\mathcal{X}_1 \circ \mathcal{X}_2$ denotes the pointwise concatenation of $\mathcal{X}_1$ and $\mathcal{X}_2$ given by $\{M \mid \exists M_1 \in \mathcal{X}_1, M_2 \in \mathcal{X}_2 : M = M_1 \circ M_2\}$. For $\mathcal{X} \subseteq \mathcal{M}_\mathcal{P}$, we define $\mathcal{X}^0 = \{\varepsilon\}$, where ε denotes the empty MSC, and for $i \geq 0$, $\mathcal{X}^{i+1} = \mathcal{X} \circ \mathcal{X}^i$. The *asynchronous iteration of* $\mathcal{X}$ is then defined by $\mathcal{X}^{\circledast} = \bigcup_{i \geq 0} \mathcal{X}^i$.

Let $\mathcal{L} \subseteq \mathcal{M}_\mathcal{P}$. We say that $\mathcal{L}$ is *finitely generated* if there is a finite set of MSCs $\mathcal{X} \subseteq \mathcal{M}_\mathcal{P}$ such that $\mathcal{L} \subseteq \mathcal{X}^{\circledast}$.

We first observe that not every regular MSC language is finitely generated. Let $\mathcal{P} = \{p, q, r\}$. Consider the regular expression $p!q\ \sigma^*\ q?p$, where σ is the sequence $p!r\ r?p\ r!q\ q?r\ q!p\ p?q$. This expression describes an infinite set of MSCs $\mathcal{L}_{inf} = \{M_i\}_{i=0}^{\omega}$. Figure 3 shows the MSC M_2. None of the MSCs in this language can be expressed as the concatenation of two or more non-trivial MSCs. Hence, this language is *not* finitely generated.

Our interest in finitely generated languages stems from the fact that MSGs can define only finitely generated MSC languages. We now wish to decide whether a regular MSC language is finitely generated. To do this we shall make use of the notion of atoms.

Let $M, M' \in \mathcal{M}_\mathcal{P}$ be nonempty MSCs. Then M' *is a component of* M in case there exist $M_1, M_2 \in \mathcal{M}$ such that $M = M_1 \circ M' \circ M_2$. We say that M *is an atom* if the only component of M is M itself. Thus, an atom is a nonempty message sequence chart that cannot be decomposed into non-trivial subcomponents. For an MSC M we let $Comp(M)$ denote the set $\{M' \mid M'$ is a component of $M\}$. We let $Atoms(M)$ denote the set $\{M' \mid M'$ is an atom and M' is a component of $M\}$.

For an MSC language $\mathcal{L} \subseteq \mathcal{M}_{\mathcal{P}}$, $Comp(\mathcal{L}) = \bigcup\{Comp(M) \mid M \in \mathcal{L}\}$ and $Atoms(\mathcal{L}) = \bigcup\{Atoms(M) \mid M \in \mathcal{L}\}$. It is clear that the question of deciding whether L is finitely generated is equivalent to checking whether $Atoms(\mathcal{L})$ is finite.

Theorem 4.1. *Let $\mathcal{L}$ be a regular MSC language. It is decidable whether $\mathcal{L}$ is finitely generated.*

Proof Sketch: Let $\mathcal{A} = (S, \Sigma, s_{in}, \delta, F)$ be the minimum DFA for $\mathcal{L}$. From $\mathcal{A}$, we construct a finite family of finite-state automata which together accept the linearizations of the MSCs in $Atoms(\mathcal{L})$. It will then follow that $\mathcal{L}$ is finitely generated if and only if each of these automata accepts a finite language. We sketch the details below.

We know that for each live state $s \in S$, we can assign a capacity function $\mathcal{K}_s : Ch \to \mathbb{N}$ which counts the number of messages present in each channel when the state s is reached. We say that s is a *zero-capacity state* if $\mathcal{K}_s(c) = 0$ for each channel c. The following claims are easy to prove.

Claim 1: Let M be an MSC in $Comp(\mathcal{L})$ (in particular, in $Atoms(\mathcal{L})$) and w be a linearization of M. Then, there are zero-capacity live states s, s' in $\mathcal{A}$ such that $s \xrightarrow{w} s'$.

Claim 2: Let M be an MSC in $Comp(\mathcal{L})$. M is an atom if and only if for each linearization w of M and each pair (s, s') of zero-capacity live states in $\mathcal{A}$, if $s \xrightarrow{w} s'$ then no intermediate state visited in this run has zero-capacity.

Let us say that two complete words w and w' are equivalent, written $w \sim w'$, if they are linearizations of the same MSC. Suppose $s \xrightarrow{w} s'$ and $w \sim w'$. Then it is easy to see that $s \xrightarrow{w'} s'$ as well.

With each pair (s, s') of live zero-capacity states we associate a language $L_{At}(s, s')$. A word w belongs to $L_{At}(s, s')$ if and only if w is complete, $s \xrightarrow{w} s'$ and for each $w' \sim w$ the run $s \xrightarrow{w'} s'$ has no zero-capacity intermediate states. From the two claims proved above, each of these languages consists of all the linearizations of some subset of $Atoms(\mathcal{L})$ and the linearizations of each element of $Atoms(\mathcal{L})$ is contained in some $L_{At}(s, s')$. Thus, it suffices to check for the finiteness of each of these languages.

Let $L_{s,s'}$ be the language of strings accepted by $\mathcal{A}$ when we set the initial state to be s and the set of final states to be $\{s'\}$. Clearly $L_{At}(s, s') \subseteq L_{s,s'}$. We now show how to construct an automaton for for $L_{At}(s, s')$.

We begin with $\mathcal{A}$ and prune the automaton as follows:

- Remove all incoming edges at s and all outgoing edges at s'.
- If $t \notin \{s, s'\}$ and $\mathcal{K}_t = \overline{0}$, remove t and all its incoming and outgoing edges.
- Recursively remove all states that become unreachable as a result of the preceding operation.

Let $\mathcal{B}$ be the resulting automaton. B accepts any complete word w on which the run from s to s' does not visit an intermediate zero-capacity state. Clearly,

$L_{At}(s, s') \subseteq L(\mathcal{B})$. However, $L(\mathcal{B})$ may also contain linearizations of non-atomic MSCs that happen to have no complete prefix. For all such words, we know from Claim 2 that there is at least one equivalent linearization on which the run passes through a zero-capacity state and which would hence be eliminated from $L(\mathcal{B})$. Thus, $L_{At}(s, s')$ is the $\sim$-closed subset of $L(\mathcal{B})$ and we need to prune $\mathcal{B}$ further to obtain the automaton for $L_{At}(s, s')$.

Recall that the original DFA $\mathcal{A}$ was structurally closed with respect to the independence relation on communication actions in the following sense. Suppose $\delta(s_1, a) = s_2$ and $\delta(s_2, b) = s_3$ with a, b independent at s_1. Then, there exists s_2' such that $\delta(s_1, b) = s_2'$ and $\delta(s_2', a) = s_3$.

To identify the closed subset of $L(\mathcal{B})$, we look for local violations of this "diamond" property and carefully prune transitions. We first blow up the state space into triples of the form (s_1, s_2, s_3) such that there exist a and a' with $\delta(s_1, a) = s_2$ and $\delta(s_2, a') = s_3$. Let S' denote this set of triples. We obtain a nondeterministic transition relation $\delta' = \{((s_1, s_2, s_3), a, (t_1, t_2, t_3)) \mid s_2 = t_1, s_3 = t_2, \delta(s_2, a) = s_3\}$. Set $S_{in} = \{(s_1, s_2, s_3) \in S' \mid s_2 = s_{in}\}$ and $F' = \{(s_1, s_f, s_2) \in S' \mid s_f \in F\}$. Let $\mathcal{B}' = (S', \Sigma, \delta', S_{in}, F')$.

Consider any state s_1 in $\mathcal{B}$ such that a and b are independent at s_1, $\delta(s_1, a) = s_2$, $\delta(s_2, b) = s_3$ but there is no s_2' such that $\delta(s_1, b) = s_2'$ and $\delta(s_2', a) = s_3$. For each such s_1, we remove all transitions of the form $((t, s_0, s_1), a, (s_0, s_1, t'))$ and $((t, s_2, s_3), b, (s_2, s_3, t'))$ from $\mathcal{B}'$. We then recursively remove all states which become unreachable after this pruning.

Eventually, we arrive at an automaton $\mathcal{C}$ such that $L(\mathcal{C}) = L_{At}(s, s')$. Since $\mathcal{C}$ is a finite-state automaton, we can easily check whether $L(\mathcal{C})$ is finite. This process is repeated for each pair of live zero-capacity states. ◻

Alur and Yannakakis [AY99] introduced the notion of boundedness for MSGs. They also showed that the set of all linearizations of the MSCs defined by a bounded MSG is a regular string language. In the present setting this boils down to boundedness of an MSG being a sufficient condition for its MSC language to be regular. To state their condition, we first have to define the notion of connectedness.

Let $M = (E, \leq, \lambda)$ be an MSC. We let CG_M, *the communication graph of* M, be the directed graph $(\mathcal{P}, \mapsto)$ defined as follows: $(p, q) \in \mapsto$ if and only if there exists an $e \in E$ with $\lambda(e) = p!q$. M is *connected* if CG_M consists of one non-trivial strongly connected component and isolated vertices. For an MSC language $\mathcal{L} \subseteq \mathcal{M}_{\mathcal{P}}$ we say that $\mathcal{L}$ *is connected* in case every $M \in \mathcal{L}$ is connected.

Let $\mathcal{G} = (Q, \rightarrow, Q_{in}, F, \Phi)$ be an MSG. A *loop* in $\mathcal{G}$ is a sequence of edges that starts and ends at the same node. We say that $\mathcal{G}$ is *bounded* if for every loop $\pi = q \rightarrow q_1 \rightarrow \cdots \rightarrow q$, the MSC $M(\pi)$ is connected. An MSC language $\mathcal{L}$ is a *bounded MSG-language* if there exists a bounded MSG $\mathcal{G}$ with $\mathcal{L} = \mathcal{L}(\mathcal{G})$.

It is easy to check whether a given MSG is bounded. Clearly, the MSG of Figure 2 is *not* bounded. One of the main results concerning bounded MSGs shown in [AY99] at once implies:

Lemma 4.2. *Every bounded MSG-language is a regular MSC language.*

Our main interest in this section is to prove the converse of Lemma 4.2.

Lemma 4.3. *Let $\mathcal{L}$ be a finitely generated regular MSC language. Then, $\mathcal{L}$ is a bounded MSG-language.*

Proof Sketch: Suppose $\mathcal{L}$ is a regular MSC language accepted by the minimal DFA $\mathcal{A} = (S, \Sigma, s_{in}, \delta, F)$. Let $Atoms(\mathcal{L}) = \{a_1, a_2, \ldots, a_m\}$. For each atom a_i, fix a linearization $u_i \in Lin(a_i)$. Define an auxiliary DFA $\mathcal{B} = (S^{\overline{0}}, Atoms(\mathcal{L}), s_{in}, \widehat{\delta}, \widehat{F})$ as follows:

- $S^{\overline{0}}$ is the set of states of $\mathcal{A}$ which have zero-capacity functions.
- $\widehat{F} = F$
- $\widehat{\delta}(s, a_i) = s'$ iff $\delta(s, u_i) = s'$ in $\mathcal{A}$. (Note that $u, u' \in Lin(a_i)$ implies $\delta(s, u) = \delta(s, u')$, so s' is fixed independent of the choice of $u_i \in Lin(a_i)$.)

Thus, $\mathcal{B}$ accepts the (regular) language of atoms corresponding to $\mathcal{L}(\mathcal{A})$. We can define a natural independence relation I_A on atoms as follows: atoms a_i and a_j are independent if and only if the set of active processes in a_i is disjoint from the set of active processes in a_j. (The process p is *active* in the MSC $(E, \leq, \lambda)$ if E_p is non-empty.)

It follows that $L(\mathcal{B})$ is a regular Mazurkiewicz trace language over the trace alphabet $(Atoms(\mathcal{L}), I_A)$. As usual, for $w \in Atoms(\mathcal{L})^*$, we let $[w]$ denote the equivalence class of w with respect to I_A.

We now fix a strict linear order $\prec$ on $Atoms(\mathcal{L})$. This induces a (lexicographic) total order on words over $Atoms(\mathcal{L})$. Let $\textsc{Lex} \subseteq Atoms(\mathcal{L})^*$ be given by: $w \in \textsc{Lex}$ iff w is the lexicographically least element in $[w]$.

For a trace language L over $(Atoms(\mathcal{L}), I_A)$, let $lex(L)$ denote the set $L \cap \textsc{Lex}$.

Remark 4.4 ([DR95], Sect. 6.3.1)

(i) If L is a regular trace language over $(Atoms(\mathcal{L}), I_A)$, then $lex(L)$ is a regular language over $Atoms(\mathcal{L})$. Moreover, $L = \{[w] \mid w \in lex(L))$.
(ii) If $w_1 w w_2 \in \textsc{Lex}$, then $w \in \textsc{Lex}$.
(iii) If w is not *a connected*[1] *trace, then $ww \notin \textsc{Lex}$.*

From (i) we know that $lex(L(\mathcal{B}))$ is a regular language over $Atoms(\mathcal{L})$. Let $\mathcal{C} = (S', Atoms(\mathcal{L}), s'_{in}, \delta', F')$ be the DFA over $Atoms(\mathcal{L})$ obtained by eliminating the (unique) dead state, if any, from the minimal DFA for $lex(L(\mathcal{B}))$. It is easy to see that an MSC M belongs to $\mathcal{L}$ if and only if it can be decomposed into a sequence of atoms accepted by $\mathcal{C}$. Using this fact, we can derive an MSG $\mathcal{G}$ from $\mathcal{C}$ such that $\mathcal{L}(\mathcal{G}) = \mathcal{L}$. We define $\mathcal{G} = (Q, \rightarrow, Q_{in}, F, \Phi)$ as follows:

- $Q = S' \times (Atoms(\mathcal{L}) \cup \{\varepsilon\})$.
- $Q_{in} = \{(s'_{in}, \varepsilon)\}$.

[1] A trace is said to be *connected* if, when viewed as a labelled partial order, its Hasse diagram consists of a single connected component. See [DR95] for a more formal definition.

– $(s, b) \rightarrow (s', b')$ iff $\delta'(s, b') = s'$.
– $F' = F \times \mathit{Atoms}(\mathcal{L})$.
– $\Phi(s, b) = b$.

Clearly $\mathcal{G}$ is an MSG and the MSC language that it defines is $\mathcal{L}$. We need to show that $\mathcal{G}$ is bounded. To this end, let $\pi = (s, b) \rightarrow (s_1, b_1) \rightarrow \cdots \rightarrow (s_n, b_n) \rightarrow (s, b)$ be a loop in $\mathcal{G}$. We need to establish that the MSC $M(\pi) = b_1 \circ \cdots \circ b_n \circ b$ defined by this loop is connected. Let $w = b_1 b_2 \ldots b_n b$.

Consider the corresponding loop $s \xrightarrow{b_1} s_1 \xrightarrow{b_2} \cdots \xrightarrow{b_n} s_n \xrightarrow{b} s$ in $\mathcal{C}$. Since every state in $\mathcal{C}$ is live, there must be words w_1, w_2 over $\mathit{Atoms}(\mathcal{L})$ such that $w_1 w^k w_2 \in \mathit{lex}(L(\mathcal{B}))$ for every $k \geq 0$.

From (ii) of Remark 4.4, $w^k \in \textsc{Lex}$. This means, by (iii) of Remark 4.4, that w describes a connected trace over $(\mathit{Atoms}(\mathcal{L}), I_A)$. From this, it is not difficult to see that the underlying undirected graph of the communication graph $CG_{M(\pi)} = (\mathcal{P}, \mapsto)$ consists of a single connected component $C \subseteq \mathcal{P}$ and isolated processes. We have to argue that the component C is, in fact, *strongly* connected. We show that if C is not strongly connected, then the regular MSC language $\mathcal{L}$ is not B-bounded for any $B \in \mathbb{N}$, thus contradicting Proposition 2.1.

Suppose that the underlying graph of C is connected but C not strongly connected. Then, there exist two processes $p, q \in C$ such that $p \mapsto q$, but there is no path from q back to p in $CG_{M(\pi)}$. For $k \geq 0$, let $M(\pi)^k = (E, \leq, \lambda)$ be the MSC corresponding to the k-fold iteration $\underbrace{M(\pi) \circ M(\pi) \circ \cdots \circ M(\pi)}_{k \text{ times}}$. Since $p \mapsto q$ in $CG_{M(\pi)}$, it follows that there are events labelled $p!q$ and $q?p$ in $M(\pi)$. Moreover, since there is no path from q back to p in $CG_{M(\pi)}$, we can conclude that in $M(\pi)^k$, for each event e with $\lambda(e) = p!q$, there is no event labelled $q?p$ in $\downarrow e$. This means that $M(\pi)^k$ admits a linearization v'_k with a prefix τ'_k which includes all the events labelled $p!q$ and excludes all the events labelled $q?p$, so that $|\tau|_{p!q} - |\tau|_{q?p} \geq k$.

By Proposition 2.1, since $\mathcal{L}$ is a regular MSC language, there is be a bound $B \in \mathbb{N}$ such that every word in $\mathcal{L}$ is B-bounded—that is, for each $v \in \mathcal{L}$, for each prefix τ of v and for each channel $(p, q) \in \mathit{Ch}$, $|\tau|_{p!q} - |\tau|_{q?p} \leq B$. Recall that $w_1 w^k w_2 \in \mathit{lex}(L(\mathcal{B}))$ for every $k \geq 0$. Fix linearizations v_1 and v_2 of the atom sequences w_1 and w_2, respectively. Then, for every $k \geq 0$, $u_k = v_1 v'_k v_2 \in \mathcal{L}$ where v'_k is the linearization of $M(\pi)^k$ defined earlier. Setting k to be $B{+}1$, we find that u_k admits a prefix $\tau_k = v_1 \tau'_k$ such that $|\tau_k|_{p!q} - |\tau_k|_{q?p} \geq B{+}1$, which contradicts the B-boundedness of $\mathcal{L}$.

Hence, it must be the case that C is a strongly connected component, which establishes that the MSG $\mathcal{G}$ we have constructed is bounded. $\square$

Putting together Lemmas 4.2 and 4.3, we obtain the following characterization of MSG-definable regular MSC languages.

Theorem 4.5. *Let $\mathcal{L}$ be a regular MSC language. Then the following statements are equivalent:*

(i) $\mathcal{L}$ is finitely generated.

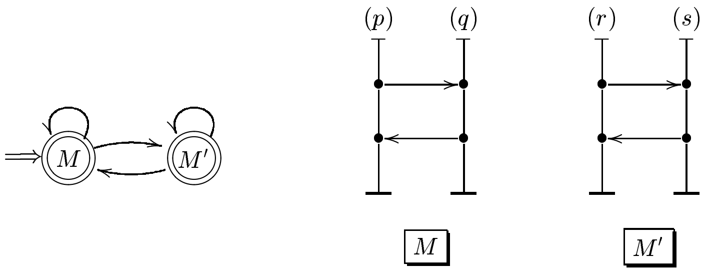

Fig. 4. An non-bounded MSG whose language is regular.

(ii) $\mathcal{L}$ *is a bounded MSG-language.*
(iii) $\mathcal{L}$ *is MSG-definable.*

It is easy to see that boundedness is not a necessary condition for regularity. Consider the MSG in Figure 4, which is not bounded. It accepts the regular MSC language $M \circ (M + M')^{\circledast}$.

Thus, it would be useful to provide a characterization of the class of MSGs representing regular MSC languages. Unfortunately, the following result shows that there is no (recursive) characterization of this class.

Theorem 4.6. *The problem of deciding whether a given MSG represents a regular MSC language is undecidable.*

Proof Sketch: It is known that the problem of determining whether the trace-closure of a regular language $L \subseteq A^*$ with respect to a trace alphabet (A, I) is also regular is undecidable [Sak92]. We reduce this problem to the problem of checking whether the MSC language defined by an MSG is regular.

Let (A, I) be a trace alphabet. We fix a set of processes $\mathcal{P}$ and the associated communication alphabet Σ and encode each letter a by an MSC M_a over $\mathcal{P}$ such that the communication graph CG_{M_a} is strongly connected. Moreover, if $(a, b) \in I$, then the sets of active processes of M_a and M_b are disjoint. The encoding ensures that we can construct a finite-state automaton to parse any word over Σ and determine whether it arises as the linearization of an MSC of the form $M_{a_1} \circ M_{a_2} \circ \cdots \circ M_{a_k}$. If so, the parser can uniquely reconstruct the corresponding word $a_1 a_2 \ldots a_k$ over A.

Let $\mathcal{A}$ be the minimal DFA corresponding to a regular language L over A. We construct an MSG $\mathcal{G}$ from $\mathcal{A}$ as described in the proof of Lemma 4.3. Given the properties of our encoding, we can then establish that the MSC language $L(\mathcal{G})$ is regular if and only if the trace-closure of L is regular, thus completing the reduction. $\square$

References

[AHP96] Alur, R., Holzmann, G.J., and Peled, D.: An analyzer for message sequence charts. *Software Concepts and Tools*, **17**(2) (1996) 70–77.

[AY99] Alur, R., and Yannakakis, M.: Model checking of message sequence charts. *Proc. CONCUR'99*, LNCS **1664**, Springer Verlag (1999) 114–129.

[BL97] Ben-Abdallah, H., and Leue, S.: Syntactic detection of process divergence and non-local choice in message sequence charts. *Proc. TACAS'97*, LNCS **1217**, Springer-Verlag (1997) 259–274.

[BJR97] Booch, G., Jacobson, I., and Rumbaugh, J.: *Unified Modeling Language User Guide*. Addison Wesley (1997).

[DR95] Diekert, V., and Rozenberg, G. (Eds.): *The book of traces*. World Scientific (1995).

[HG97] Harel, D., and Gery, E.: Executable object modeling with statecharts. *IEEE Computer*, July 1997 (1997) 31–42.

[HM+99] Henriksen, J.G., Mukund, M., Narayan Kumar, K., and Thiagarajan, P.S.: Towards a theory of regular MSC languages. BRICS Report RS-99-52, Department of Computer Science, Aarhus University, Denmark (1999).

[ITU97] ITU-TS Recommendation Z.120: *Message Sequence Chart (MSC)*. ITU-TS, Geneva (1997)

[LL95] Ladkin, P.B., and Leue, S.: Interpreting message flow graphs. *Formal Aspects of Computing* **7**(5) (1975) 473–509.

[LP97] Levin, V., and Peled, D.: Verification of message sequence charts via template matching. *Proc. TAPSOFT'97*, LNCS **1214**, Springer-Verlag (1997) 652–666.

[Mus99] Muscholl, A.: Matching specifications for message sequence charts. *Proc. FOSSACS'99*, LNCS **1578**, Springer-Verlag (1999) 273–287.

[MP99] Muscholl, A., and Peled, D.: Message sequence graphs and decision problems on Mazurkiewicz traces. *Proc. MFCS'99*, LNCS **1672**, Springer-Verlag (1999) 81–91.

[MPS98] Muscholl, A., Peled, D., and Su, Z.: Deciding properties for message sequence charts. *Proc. FOSSACS'98*, LNCS **1378**, Springer-Verlag (1998) 226–242.

[RGG96] Rudolph, E., Graubmann, P., and Grabowski, J.: Tutorial on message sequence charts. In *Computer Networks and ISDN Systems—SDL and MSC*, Volume 28 (1996).

[Sak92] Sakarovitch, J.: The "last" decision problem for rational trace languages. *Proc. LATIN'92*, LNCS **583**, Springer-Verlag (1992) 460–473.