Mathematical Foundation of Computer Science (MFCS) 2000,
Proceedings: Mogens Nielsen, Branislav Rovan (eds.)
Springer Lecture Notes in Computer Science **1893** (2000), 405–414.

Regular Collections of Message Sequence Charts
(Extended Abstract)

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

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

Abstract. Message Sequence Charts (MSCs) are an attractive visual formalism used during the early stages of design in domains such as telecommunication software. A popular mechanism for generating a collection of MSCs is a Hierarchical Message Sequence Chart (HMSC). However, not all HMSCs describe collections of MSCs that can be "realized" as a finite-state device. Our main goal is to pin down this notion of realizability. We propose an independent notion of *regularity* for collections of MSCs and explore its basic properties. In particular, we characterize regular collections of MSCs in terms of finite-state distributed automata called bounded message-passing automata, in which a set of sequential processes communicate with each other asynchronously over bounded FIFO channels. We also provide a logical characterization in terms of a natural monadic second-order logic interpreted over MSCs. It turns out that realizable collections of MSCs as specified by HMSCs constitute a strict subclass of the regular collections of MSCs.

1 Introduction

Message sequence charts (MSCs) are an appealing visual formalism often used to capture system requirements in the early stages of design. They are particularly suited for describing scenarios for distributed telecommunication software [12, 19]. They have also been called timing sequence diagrams, message flow diagrams and object interaction diagrams and are used in a number of software engineering methodologies [4, 9, 19]. In its basic form, an MSC depicts the exchange of messages between the processes of a distributed system along a single partially-ordered execution. A collection of MSCs is used to capture the scenarios that a designer might want the system to exhibit (or avoid).

Given the requirements in the form of a collection of MSCs, one can hope to do formal analysis and discover design errors at an early stage. A natural question in this context is to identify when a collection of MSCs is amenable to formal analysis. A related issue is how to represent such collections.

⋆ Supported in part by IFCPAR Project 2102-1.
⋆⋆ Basic Research in Computer Science,
Centre of the Danish National Research Foundation.

A standard way to generate a collection of MSCs is to use a Hierarchical Message Sequence Chart (HMSC) [15]. An HMSC is a finite directed graph in which each node is labelled, in turn, by an HMSC. The labels on the nodes are not permitted to refer to each other. From an HMSC we can derive an equivalent Message Sequence Graph (MSG) [1] by flattening out the hierarchical labelling to obtain a graph where each node is labelled by a simple MSC. An MSG defines a collection of MSCs obtained by concatenating the MSCs labelling each path from an initial vertex to a terminal vertex. Though HMSCs provide more succinct specifications than MSGs, they are only as expressive as MSGs. Thus, one often restricts one's attention to characterizing structural properties of MSGs rather than of HMSCs [2, 17, 18].

In [2], it is shown that *bounded* MSGs define reasonable collections of MSCs—the collection of MSCs generated by a bounded MSG can be represented as a regular string language. Thus, behaviours captured by bounded MSGs can, in principle, be realized as finite-state automata. In general, the collection of MSCs defined by an *arbitrary* MSG is not realizable in this sense. A characterization of the collections of MSCs definable using bounded MSGs is provided in [11].

The main goal of this paper is to pin down this notion of realizability in terms of a notion of *regularity* for collections of MSCs. One consequence of our study is that our definition of regularity provides a general and robust setting for studying collections of MSCs. A second consequence, which follows from the results in [11], is that bounded MSGs define a strict subclass of regular collections of MSCs. A final consequence is that our notion addresses an important issue raised in [6]; namely, how to convert requirements as specified by MSCs into distributed, state-based specifications.

Another motivation for focussing on regularity is that this notion has turned out to be very fruitful in a variety of contexts including finite (and infinite) strings, trees and restricted partial orders known as Mazurkiewicz traces [7, 21]. In all these settings there is a representation of regular collections in terms of finite-state devices. There is also an accompanying monadic second-order logic that usually induces temporal logics using which one can reason about such collections [21]. One can then develop automated model-checking procedures for verifying properties specified in these temporal logics. In this context, the associated finite-state devices representing the regular collections often play a very useful role [22].

We show here that our notion of regular MSC languages fits in nicely with a related notion of a finite-state device, as also a monadic second-order logic. We fix a finite set of processes $\mathcal{P}$ and consider $\mathcal{M}$, the universe of MSCs defined over the set $\mathcal{P}$. An MSC in $\mathcal{M}$ can be viewed as a partial order labelled using a finite alphabet Σ that is canonically fixed by $\mathcal{P}$. We say that $L \subseteq \mathcal{M}$ is regular if the set of all linearizations of all members of L constitutes a regular subset of Σ^*. A crucial point is that the universe $\mathcal{M}$ is itself *not* regular according to our definition, unlike the classical setting of strings (or trees or Mazurkiewicz traces). This fact has a strong bearing on the automata-theoretic and logical formulations in our work.

It turns out that regular MSC languages can be stratified using the concept of *bounds*. An MSC is said to be B-bounded for a natural number B if at every "prefix" of the MSC and for every pair of processes (p, q) there are at most B messages that p has sent to q that have yet to be received by q. An MSC language is B-bounded if every member of the language is B-bounded. Fortunately, for every regular MSC language L we can effectively compute a (minimal) bound B such that L is B-bounded. This leads to our automaton model called B-bounded message-passing automata. The components of such an automaton correspond to the processes in $\mathcal{P}$. These components communicate with each other over (potentially unbounded) FIFO channels. We say that a message-passing automaton is B-bounded if, during its operation, it is never the case that a channel contains more than B messages. We establish a precise correspondence between B-bounded message-passing automata and B-bounded regular MSC languages. In a similar vein, we formulate a natural monadic second-order logic $\mathrm{MSO}(\mathcal{P}, B)$ interpreted over B-bounded MSCs. We then show that B-bounded regular MSC languages are exactly those that are definable in $\mathrm{MSO}(\mathcal{P}, B)$.

In related work, a number of studies are available that are concerned with individual MSCs in terms of their semantics and properties [1, 13]. As pointed out earlier, a nice way to generate a collection of MSCs is to use an MSG. A variety of algorithms have been developed for MSGs in the literature—for instance, pattern matching [14, 17, 18] and detection of process divergence and non-local choice [3]. A systematic account of the various model-checking problems associated with MSGs and their complexities is given in [2].

In this paper, we confine our attention to *finite* MSCs. The issues investigated here have, at present, no counterparts in the infinite setting. We feel, however, that our results will serve as a launching pad for a similar account concerning infinite MSCs. This should then lead to the design of appropriate temporal logics and automata-theoretic solutions (based on message-passing automata) to model-checking problems for these logics.

The paper is organized as follows. In the next section we introduce MSCs and regular MSC languages. In Section 3 we establish our automata-theoretic characterization and, in Section 4, the logical characterization. While doing so, we borrow one basic result and a couple of proof techniques from the theory of Mazurkiewicz traces [7]. However, we need to modify some of these techniques in a non-trivial way (especially in the setting of automata) due to the asymmetric flow of information via messages in the MSC setting, as opposed to the symmetric information flow via handshake communication in the trace setting. Due to lack of space, we provide only proof ideas. Detailed proofs are available in [10].

2 Regular MSC Languages

Through the rest of the paper, 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*.

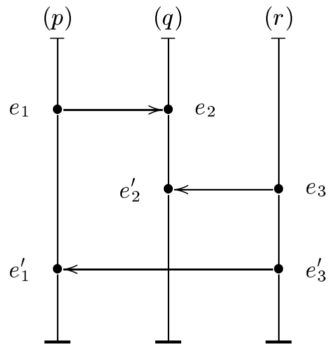

Fig. 1. An example MSC over $\{p, q, r\}$.

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 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 relation $R_p = (E_p \times E_p) \cap \leq$.

An MSC (over $\mathcal{P}$) is a *finite* Σ-labelled poset $M = (E, \leq, \lambda)$ that 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 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 smaller (relative to $\leq$) events appearing above the larger 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 $Lin(M) = \{\lambda(\pi) \mid \pi$ 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.

In the literature [1, 18] one sometimes considers a more generous notion of linearization where two *adjacent* receive actions in a process corresponding to messages from *different* senders are deemed to be causally independent. For instance, $p!q$ $r!q$ $q?r$ $q?p$ $r!p$ $p?r$ would also be a valid linearization of the MSC in Figure 1. All our results go through with suitable modifications even in the presence of this more generous notion of linearization.

Henceforth, we will 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.

To directly characterize the subsets of Σ^* that correspond to regular MSC languages, we proceed as follows. Let $Com = \{(p!q, q?p) \mid (p,q) \in Ch\}$. For $\tau \in \Sigma^*$ and $a \in \Sigma$, let $|\tau|_a$ denote the number of times a appears in τ. We say that $\sigma \in \Sigma^*$ is *proper* if for every prefix τ of σ and every pair $(a,b) \in Com$, $|\tau|_a \geq |\tau|_b$. We say that σ is *complete* if σ is proper and $|\sigma|_a = |\sigma|_b$ for every $(a,b) \in Com$. Next we define a *context-sensitive* independence relation $I \subseteq \Sigma^* \times (\Sigma \times \Sigma)$ as follows: $(\sigma, a, b) \in I$ if σab is proper, $a \in \Sigma_p$ and $b \in \Sigma_q$ for distinct processes p and q, and if $(a,b) \in Com$ then $|\sigma|_a > |\sigma|_b$. Observe that if $(\sigma, a, b) \in I$ then $(\sigma, b, a) \in I$.

Let $\Sigma^\circ = \{\sigma \mid \sigma \in \Sigma^* \text{ and } \sigma \text{ is complete}\}$. We then define $\sim\, \subseteq \Sigma^\circ \times \Sigma^\circ$ to be the least equivalence relation such that if $\sigma = \sigma_1 ab \sigma_2$, $\sigma' = \sigma_1 ba \sigma_2$ and $(\sigma_1, a, b) \in I$ then $\sigma \sim \sigma'$. It is important to note that $\sim$ is defined over Σ° (and not Σ^*). It is easy to verify that for each $M \in \mathcal{M}_\mathcal{P}$, $Lin(M)$ is a subset of Σ° and is in fact a $\sim$-equivalence class over Σ°.

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 $L = \bigcup\{Lin(M) \mid M \in \mathcal{L}\}$. It is easy to see that $L \subseteq \Sigma^*$ is a regular string MSC language if and only if L is a regular subset of Σ^*, every word in L is complete and L is $\sim$-closed (that is, for each $\sigma \in L$, if $\sigma \in L$ and $\sigma \sim \sigma'$ then $\sigma' \in L$). Clearly 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 channel-capacity function $\mathcal{K}_s : Ch \to \mathbb{N}$ that 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 $\delta(s'_1, a) = s_2$.

These conditions can be checked in time linear in the size of δ. 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 complete word 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 $L \subseteq \Sigma^{\circ}$ is B-bounded if every word $\sigma \in L$ is B-bounded. Let L be a regular MSC language and let $\mathcal{A} = (S, \Sigma, s_{in}, \delta, F)$ be its minimal DFA, as described above, with capacity functions $\{\mathcal{K}_s\}_{s \in S}$. Let $B_L = \max_{s \in S, c \in Ch} \mathcal{K}_s(c)$. Then it is easy to see that L is *B_L-bounded* and that B_L can be effectively computed from $\mathcal{A}$. Finally, we shall say that the MSC M is B-bounded if every string in $Lin(M)$ is B-bounded. A collection of MSCs is B-bounded if every member of the collection is B-bounded.

3 An Automata-Theoretic Characterization

Recall that the set of processes $\mathcal{P}$ determines the communication alphabet Σ and that for $p \in \mathcal{P}$, Σ_p denotes the actions in which process p participates.

Definition 3.1. *A* message-passing automaton *over Σ is a structure $\mathcal{A} = (\{\mathcal{A}_p\}_{p \in \mathcal{P}}, \Delta, s_{in}, \mathcal{F})$ where*

- *Δ is a finite alphabet of messages.*
- *Each component $\mathcal{A}_p$ is of the form $(S_p, \rightarrow_p)$ where*
  - *S_p is a finite set of p-local states.*
  - *$\rightarrow_p \subseteq S_p \times \Sigma_p \times \Delta \times S_p$ is the p-local transition relation.*
- *$s_{in} \in \prod_{p \in \mathcal{P}} S_p$ is the global initial state.*
- *$\mathcal{F} \subseteq \prod_{p \in \mathcal{P}} S_p$ is the set of global final states.*

The local transition relation $\rightarrow_p$ specifies how process p sends and receives messages. The transition $(s, p!q, m, s')$ specifies that when p is in the state s, it can send the message m to q by executing the action $p!q$ and move to the state s'. The message m is, as a result, appended to the queue in channel (p,q). Similarly, the transition $(s, p?q, m, s')$ signifies that in the state s, the process p can receive the message m from q by executing the action $p?q$ and move to the state s'. The message m is removed from the head of the queue in channel (q,p).

The set of global states of $\mathcal{A}$ is given by $\prod_{p \in \mathcal{P}} S_p$. For a global state s, we let s_p denote the pth component of s. A *configuration* is a pair (s, χ) where s is a global state and $\chi : Ch \rightarrow \Delta^*$ is the *channel state* that specifies the queue of messages currently residing in each channel c. The *initial configuration* of $\mathcal{A}$ is $(s_{in}, \chi_\varepsilon)$ where $\chi_\varepsilon(c)$ is the empty string ε for every channel c. The set of *final configurations* of $\mathcal{A}$ is $\mathcal{F} \times \{\chi_\varepsilon\}$.

We now define the set of reachable configurations $Conf_{\mathcal{A}}$ and the global transition relation $\Rightarrow \subseteq Conf_{\mathcal{A}} \times \Sigma \times Conf_{\mathcal{A}}$ inductively as follows:

- $(s_{in}, \chi_\varepsilon) \in Conf_{\mathcal{A}}$.
- Suppose $(s, \chi) \in Conf_{\mathcal{A}}$, (s', χ') is a configuration and $(s_p, p!q, m, s'_p) \in \rightarrow_p$ such that the following conditions are satisfied:

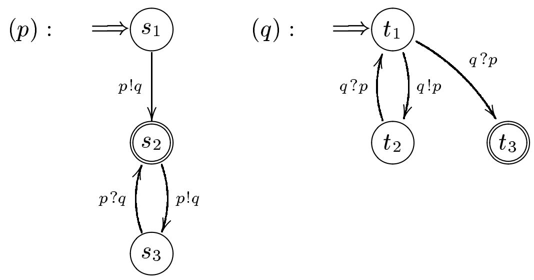

Fig. 2. A 3-bounded message-passing automaton.

- $r \neq p$ implies $s_r = s'_r$ for each $r \in \mathcal{P}$.
- $\chi'((p,q)) = \chi((p,q)) \cdot m$ and for $c \neq (p,q)$, $\chi'(c) = \chi(c)$.

Then $(s,\chi) \overset{p!q}{\Longrightarrow} (s',\chi')$ and $(s',\chi') \in Conf_{\mathcal{A}}$.

– Suppose $(s,\chi) \in Conf_{\mathcal{A}}$, (s',χ') is a configuration and $(s_p, p?q, m, s'_p) \in \rightarrow_p$ such that the following conditions are satisfied:
  - $r \neq p$ implies $s_r = s'_r$ for each $r \in \mathcal{P}$.
  - $\chi((q,p)) = m \cdot \chi'((q,p))$ and for $c \neq (q,p)$, $\chi'(c) = \chi(c)$.

  Then $(s,\chi) \overset{p?q}{\Longrightarrow} (s',\chi')$ and $(s',\chi') \in Conf_{\mathcal{A}}$.

Let $\sigma \in \Sigma^*$. A run of $\mathcal{A}$ over σ is a map $\rho : Pre(\sigma) \to Conf_{\mathcal{A}}$ (where $Pre(\sigma)$ is the set of prefixes of σ) such that $\rho(\varepsilon) = (s_{in}, \chi_\varepsilon)$ and for each $\tau a \in Pre(\sigma)$, $\rho(\tau) \overset{a}{\Longrightarrow} \rho(\tau a)$. The run ρ is *accepting* if $\rho(\sigma)$ is a final configuration. We define $L(\mathcal{A}) = \{\sigma \mid \mathcal{A}$ has an accepting run over $\sigma\}$. It is easy to see that every member of $L(\mathcal{A})$ is complete and $L(\mathcal{A})$ is $\sim$-closed.

Clearly, $L(\mathcal{A})$ need not be regular. Consider, for instance, a message-passing automaton for the canonical producer-consumer system in which the producer p sends an arbitrary number of messages to the consumer q. Since we can reorder all the $p!q$ actions to be performed before all the $q?p$ actions, the queue in channel (p,q) can grow arbitrarily long. Hence, the reachable configurations of this system are not bounded and the corresponding language is not regular.

For $B \in \mathbb{N}$, we say that a configuration (s,χ) of the message-passing automaton $\mathcal{A}$ is *B-bounded* if for every channel $c \in Ch$, it is the case that $|\chi(c)| \leq B$. We say that $\mathcal{A}$ is a B-bounded automaton if every reachable configuration $(s,\chi) \in Conf_{\mathcal{A}}$ is B-bounded. It is not difficult to show that given a message-passing automaton $\mathcal{A}$ and a bound $B \in \mathbb{N}$, one can decide whether $\mathcal{A}$ is B-bounded. Figure 2 shows an example of a 3-bounded message-passing automaton with two components, p and q. In this example, the message alphabet is a singleton, and is hence omitted. The initial state is (s_1, t_1) and there is only one final state, (s_2, t_3). This automaton accepts an infinite set of MSCs, none of which can be expressed as the concatenation of two or more non-trivial MSCs. It follows easily that the MSC language accepted by this automaton cannot be represented by an MSG.

Lemma 3.2. *Let $\mathcal{A}$ be a B-bounded automaton over Σ. Then $L(\mathcal{A})$ is a B-bounded regular MSC language.*

This result follows from the definitions and it constitutes the easy half of the characterization we wish to obtain. The difficult half is :

Lemma 3.3. *Let $L \subseteq \Sigma^*$ be a B-bounded regular MSC language. Then there exists a B-bounded message-passing automaton $\mathcal{A}$ over Σ such that $L(\mathcal{A}) = L$.*

The first step in the proof is to view L as a regular Mazurkiewicz trace language over an appropriate alphabet and apply Zielonka's theorem [23] to obtain an asynchronous automaton recognizing L. The second step, which is the hard part, is to simulate this asynchronous automaton by a B-bounded message passing automaton. This simulation makes crucial use of the technique developed in [16] for locally maintaining the latest information about other processes in a message-passing system.

We say that $\mathcal{A}$ is a bounded message-passing automaton if $\mathcal{A}$ is B-bounded for some $B \in \mathbb{N}$. The main result of this section is an easy consequence of the two previous lemmas.

Theorem 3.4. *Let $L \subseteq \Sigma^*$ be a regular MSC language. Then there exists a bounded message-passing automaton $\mathcal{A}$ over Σ such that $L(\mathcal{A}) = L$.*

4 A Logical Characterization

We formulate a monadic second-order logic that characterizes regular B-bounded MSC languages for each fixed $B \in \mathbb{N}$. Thus our logic will be parameterized by a pair $(\mathcal{P}, B)$. For convenience, we fix $B \in \mathbb{N}$ through the rest of the section. As usual, we shall assume a supply of individual variables $x, y, \ldots$, a supply of set variables $X, Y, \ldots$, and a family of unary predicate symbols $\{Q_a\}_{a \in \Sigma}$. The syntax of the logic is then given by:

$$\mathrm{MSO}(\mathcal{P}, B) ::= Q_a(x) \mid x \in X \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid (\exists x)\varphi \mid (\exists X)\varphi$$

Thus the syntax does not reflect any information about B or the structural features of an MSC. These aspects will be dealt with in the semantics. Let $\mathcal{M}_{\mathcal{P},B}$ be the set of B-bounded MSCs over $\mathcal{P}$. The formulas of our logic are interpreted over the members of $\mathcal{M}_{\mathcal{P},B}$. Let $M = (E, \leq, \lambda)$ be an MSC in $\mathcal{M}_{\mathcal{P},B}$ and $\mathcal{I}$ be an interpretation that assigns to each individual variable a member $\mathcal{I}(x)$ in E and to each set variable X a subset $\mathcal{I}(X)$ of E. Then $M \models_{\mathcal{I}} \varphi$ denotes that M satisfies φ under $\mathcal{I}$. This notion is defined in the expected manner. For instance, $M \models_{\mathcal{I}} Q_a(x)$ if $\lambda(\mathcal{I}(x)) = a$, $M \models_{\mathcal{I}} x \leq y$ if $\mathcal{I}(x) \leq \mathcal{I}(y)$ etc. For convenience, we have used $\leq$ to denote both the predicate symbol in the logic and the corresponding causality relation in the model M.

As usual, φ is a sentence if there are no free occurrences of individual or set variables in φ. With each sentence φ we can associate an MSC language $\mathcal{L}_\varphi = \{M \in \mathcal{M}_{\mathcal{P},B} \mid M \models \varphi\}$. We say that $\mathcal{L} \subseteq \mathcal{M}_{\mathcal{P},B}$ is $\mathrm{MSO}(\mathcal{P}, B)$-definable if there exists a sentence φ such that $\mathcal{L}_\varphi = \mathcal{L}$. We wish to argue that $\mathcal{L} \subseteq \mathcal{M}_{\mathcal{P},B}$ is $\mathrm{MSO}(\mathcal{P}, B)$-definable if and only if it is a B-bounded regular MSC language. It turns out the techniques used for proving a similar result in the theory of traces [8] can be suitably modified to derive our result.

Lemma 4.1. *Let φ be a sentence in $\mathrm{MSO}(\mathcal{P}, B)$. Then $\mathcal{L}_\varphi$ is a B-bounded regular MSC language.*

Proof Sketch: The fact that $\mathcal{L}_\varphi$ is B-bounded follows from the semantics and hence we just need to establish regularity. Consider $\mathrm{MSO}(\Sigma)$, the monadic second-order theory of finite strings in Σ^*. This logic has the same syntax as $\mathrm{MSO}(\mathcal{P}, B)$ except that the ordering relation is interpreted over the positions of a structure in Σ^*. Let $L = \bigcup\{Lin(M) \mid M \in \mathcal{L}_\varphi\}$. We exhibit a sentence $\widehat{\varphi}$ in $\mathrm{MSO}(\Sigma)$ such that $L = \{\sigma \mid \sigma \models \widehat{\varphi}\}$. The main observation is that the bound B ensures that the family of channel-capacity functions $\mathcal{K}$ can be captured by a fixed number of sets, which is used both to assert channel-consistency and to express the partial order of MSCs in terms of the underlying linear order of positions. The required conclusion will then follow from Büchi's theorem [5]. $\square$

Lemma 4.2. *Let $\mathcal{L} \subseteq \mathcal{M}_{\mathcal{P},B}$ be a regular MSC language. Then $\mathcal{L}$ is $\mathrm{MSO}(\mathcal{P}, B)$-definable.*

Proof Sketch: Let $L = \bigcup\{Lin(M) \mid M \in \mathcal{L}\}$. Then L is a regular (string) MSC language over Σ. Hence by Büchi's theorem [5] there exists a sentence φ in $\mathrm{MSO}(\Sigma)$ such that $L = \{\sigma \mid \sigma \models \varphi\}$. An important property of φ is that one linearization of an MSC satisfies φ if and only if all linearizations of the MSC satisfy φ. We then define the sentence $\widehat{\varphi} = \|\varphi\|$ in $\mathrm{MSO}(\mathcal{P}, B)$ inductively such that the language of MSCs defined by $\widehat{\varphi}$ is precisely $\mathcal{L}$. The key idea here is to define a canonical linearization of MSCs along the lines of [20] and show that the underlying linear order is expressible in $\mathrm{MSO}(\mathcal{P}, B)$. We obtain a formula $\widehat{\varphi}$ that says "along the canonical linearization of an MSC, the sentence φ is satisfied".

Since $\mathrm{MSO}(\Sigma)$ is decidable, it follows that $\mathrm{MSO}(\mathcal{P}, B)$ is decidable as well. To conclude, we can summarize the main results of this paper as follows.

Theorem 4.3. *Let $L \subseteq \Sigma^*$, where Σ is the communication alphabet associated with a set $\mathcal{P}$ of processes. Then, the following are equivalent.*

(i) *L is a regular MSC language.*
(ii) *L is a B-bounded regular MSC language, for some $B \in \mathbb{N}$.*
(iii) *There exists a bounded message-passing automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L$.*
(iv) *L is $\mathrm{MSO}(\mathcal{P}, B)$-definable, for some $B \in \mathbb{N}$.*

References

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

2. Alur, R., and Yannakakis, M.: Model checking of message sequence charts. *Proc. CONCUR'99*, LNCS **1664**, Springer-Verlag (1999) 114–129.
3. 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.
4. Booch, G., Jacobson, I., and Rumbaugh, J.: *Unified Modeling Language User Guide*. Addison-Wesley (1997).
5. Büchi, J. R.: On a decision method in restricted second order arithmetic. *Z. Math. Logik Grundlag. Math* **6** (1960) 66–92.
6. Damm, W., and Harel, D.: LCSs: Breathing life into message sequence charts. *Proc. FMOODS'99*, Kluwer Academic Publishers (1999) 293–312.
7. Diekert, V., and Rozenberg, G. (Eds.): *The book of traces*. World Scientific (1995).
8. Ebinger, W., and Muscholl, A.: Logical definability on infinite traces. *Theoretical Computer Science* **154**(1) (1996) 67–84.
9. Harel, D., and Gery, E.: Executable object modeling with statecharts. *IEEE Computer*, July 1997 (1997) 31–42.
10. Henriksen, J. G., Mukund, M., Narayan Kumar, K., and Thiagarajan, P. S.: Towards a theory of regular MSC languages, Report RS-99-52, BRICS, Department of Computer Science, University of Aarhus, Denmark (1999).
11. Henriksen, J. G., Mukund, M., Narayan Kumar, K., and Thiagarajan, P. S.: On message sequence graphs and finitely generated regular MSC languages, *Proc. ICALP'2000*, LNCS **1853**, Springer-Verlag (2000).
12. ITU-TS Recommendation Z.120: *Message Sequence Chart (MSC)*. ITU-TS, Geneva (1997)
13. Ladkin, P. B., and Leue, S.: Interpreting message flow graphs. *Formal Aspects of Computing* **7**(5) (1995) 473–509.
14. Levin, V., and Peled, D.: Verification of message sequence charts via template matching. *Proc. TAPSOFT'97*, LNCS **1214**, Springer-Verlag (1997) 652–666.
15. Mauw, S., and Reniers, M. A.: High-level message sequence charts, *Proc. SDL '97*, Elsevier (1997) 291–306.
16. Mukund, M., Narayan Kumar, K., and Sohoni, M.: Keeping track of the latest gossip in message-passing systems. *Proc. Structures in Concurrency Theory (STRICT)*, Workshops in Computing Series, Springer-Verlag (1995) 249–263.
17. Muscholl, A.: Matching Specifications for Message Sequence Charts. *Proc. FOSSACS'99*, LNCS **1578**, Springer-Verlag (1999) 273–287.
18. Muscholl, A., Peled, D., and Su, Z.: Deciding properties for message sequence charts. *Proc. FOSSACS'98*, LNCS **1378**, Springer-Verlag (1998) 226–242.
19. Rudolph, E., Graubmann, P., and Grabowski, J.: Tutorial on message sequence charts. In *Computer Networks and ISDN Systems—SDL and MSC*, Volume **28** (1996).
20. Thiagarajan, P. S., and Walukiewicz, I: An expressively complete linear time temporal logic for Mazurkiewicz traces. *Proc. IEEE LICS'97* (1997) 183–194.
21. Thomas, W.: Automata on infinite objects. In van Leeuwen, J. (Ed.), *Handbook of Theoretical Computer Science, Volume B*, North-Holland (1990) 133–191.
22. Vardi, M. Y., and Wolper, P.: An automata-theoretic approach to automatic program verification. In *Proc. IEEE LICS'86* (1986) 332–344.
23. Zielonka, W.: Notes on finite asynchronous automata. *R.A.I.R.O.—Inf. Théor. et Appl.*, **21** (1987) 99–135.