Regular Collections of Message Sequence Charts
(Extended Abstract)

Jesper G. Henriksen¹, Madhavan Mukund²,*
K. Narayan Kumar²,* and P.S. Thiagarajan²

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 tele-
communication software. A popular mechanism for generating a collec-
tion of MSCs is a Hierarchical Message Sequence Chart (HMSC). How-
ever, 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 rea-
лизability. We propose an independent notion of regularity for collec-
tions 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 enginee-
rming 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 focusing 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 $\Sigma$ 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 $\Sigma^*$. 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.