– MSCan –
A Tool for Analyzing MSC Specifications

Benedikt Bollig[1], Carsten Kern[2], Markus Schlütter[3], and Volker Stolz[2]

[1] LSV, CNRS UMR 8643 & ENS de Cachan, France
bollig@lsv.ens-cachan.fr
[2] Software Modeling and Verification Group, RWTH Aachen University, Germany
{kern,stolz}@informatik.rwth-aachen.de
[3] Department of Process Control Engineering, RWTH Aachen University, Germany
schluetter@plt.rwth-aachen.de

Abstract. We present the tool MSCan, which supports MSC-based system development. In particular, it automatically checks high-level MSC specifications for implementability.

1 Introduction

Message Sequence Charts (MSCs) constitute a prominent notion for describing protocols in the early stages of system development [8]. An MSC depicts a collection of processes, which, in their visual representation, are drawn as vertical lines and interpreted as time axes. An arrow from one line to a second corresponds to sending and receiving a message. Not only does the MSC standard allow to specify single scenarios; to make MSCs a flexible specification language, it also supports *choice*, *concatenation*, and *iteration*, which give rise to *high-level MSCs*. Consider Fig. 1: the MSCs M_1, M_2, and M_3 are the building blocks of the high-level MSC G, which generates scenarios such as the MSC M.

A high-level MSC specification permits a *global* view of a distributed system, whereas the future implementation thereof will usually be controlled *locally* by rather autonomous processes. Due to this inherent discrepancy, a preliminary high-level MSC specification might not be suitable for an implementation and often requires further refinement and adjustment steps. If, for example, the specification admits some global system behavior where the choice of two alternatives can be triggered by independent processes, inconsistent (local) decisions might lead the system into a deadlock. This phenomenon is known as *non-local choice* [2]. Otherwise, the high-level MSC from Fig. 1 has the local-choice property: the only choice point is entirely under the control of process 2.

A system specification with the local-choice property can always be realized by a deadlock-free distributed implementation. Other requirements ensuring implementability with various characteristics are *local cooperativity*, *global cooperativity*, and *regularity* [5, 7]. Last but not least, high-level MSCs with abovementioned properties come along with decidable model-checking problems for further analyses that, in general, are undecidable [1, 5].

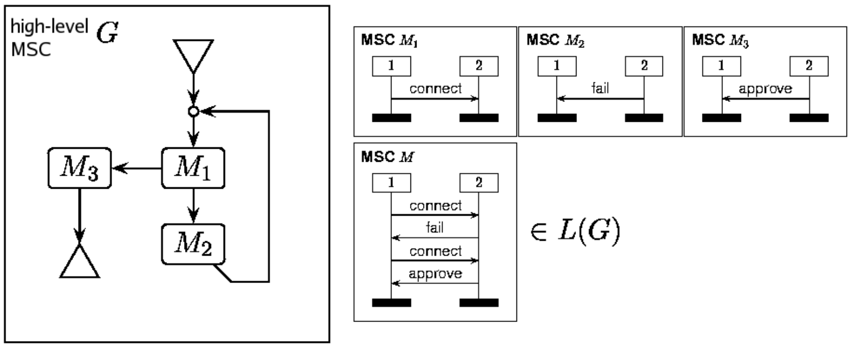

Fig. 1. A (local-choice) high-level MSC

2 The Tool MSCAN

MSCAN supports the system development based on high-level MSCs. It automatically checks a high-level MSC specification for (several variants of) local choice, local and global cooperativity, as well as regularity and many other reasonable requirements to draw conclusions about implementability, consistency, and decidable model-checking problems. Moreover, MSCAN offers numerous features for editing, displaying, and debugging high-level MSCs. It converts an ITU Z.120 textual description of a high-level MSC specification into a graph structure that naturally reflects choice, concatenation, and iteration. Based on the internal graph representation, MSCAN applies graph algorithms to explore the specification and to detect global control structures that do not allow an embedding into a locally controlled implementation.

Note that high-level MSCs, in their basic form, are only capable of specifying *finitely generated* behavior [7]. To overcome those drawbacks and to be able to specify *non* finitely generated behavior such as the alternating-bit protocol, *compositional* high-level MSCs have been introduced by Gunter et al. [6]. We would like to stress that, in all aspects, our tool supports this extension, which enjoys many nice properties and increasing popularity [4].

2.1 Graphical User Interface

To grant the user a maximum degree of comfort, the graphical user interface is partitioned into four main components (cf. Fig. 2). The upper part of the GUI is taken by the menu component of the tool (1). It offers facilities to *create*, *load*, and *save* MSC documents and to change the level of detail and the mode of analysis (e.g., lazy evaluation). Moreover, the menu allows to select a single high-level MSC property as well as grouping several properties together.

Further features that are controlled via the menu component of MSCAN are: (i) processing the MSC document and displaying its graph structure in the graph component of the GUI, (ii) displaying the properties of the currently selected

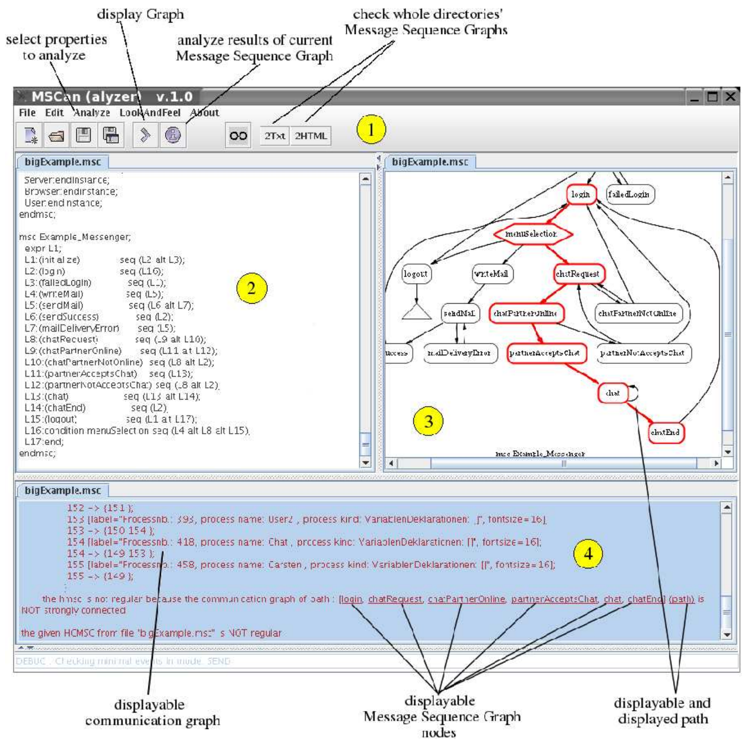

Fig. 2. An MSCAN Session

high-level MSC that have been detected so far, and (iii) checking directories recursively for high-level MSCs, executing all tests currently supported by the tool, and creating a text or HTML output containing the results of the analysis. The left component (2) may be used for editing MSC documents. Herein, the user can specify the system behavior to be analyzed as well as alter faulty specifications to eventually converge to a protocol that exhibits exactly the desired properties. Label (3) is associated with the graph component of MSCAN, in which the high-level MSC under consideration can be displayed. It allows the user to zoom in and out partial behavior as well as clicking onto nodes to depict the associated MSCs. The fourth component (4) is addressed to the analysis output of a test execution providing the user with counter examples, which may be used for debugging and system refinement. It displays test results and calls the user's attention to potential conflicts or inconsistencies in the protocol specification. Additionally, it eases the protocol designer's task of ruling out errors by visualizing high-level MSC components like nodes, edges, paths and all kinds of graphs (e.g., channel and communication graphs). This guides the user and substantially reduces her effort to detect faulty or inconsistent system behavior. For further screenshots and a more elaborate feature description, the reader may visit the web page of MSCAN located at [11].

2.2 General Information

MSCan is written in Java 1.5 using the Java graph visualization package *Grappa* [10] and the parser *MSC2000* [9]. It consists of a console application started by the class MSCExecute of the homonymous package and of a concise, interactive graphical user interface. We developed the tool in a highly modular manner to ease the integration of high-level MSC properties, analysis components, and the graphical user interface. Instructions on how to extend the collection of currently available properties can be found on the web page of the tool. We also offer a web interface and a collection of predefined sample high-level MSCs to test the basic features of the tool online [11].

3 Conclusion and Future Work

To our knowledge, there is no other tool that provides a protocol designer with a likewise great variety of facilities to analyze high-level MSCs. Another project is [3], which, in contrast to our tool, checks exclusively for the non-local choice property. Moreover, it requires a high-level MSC to be in a normal form, demanding additional effort from the protocol designer.

MSCan is currently being enhanced to integrate a subsequent implementation phase to automatically derive implementations from high-level MSCs. As a first step in that direction, we are developing a code generation back-end, which emits out-of-the-box compilable Java code from MSC documents [12].

References

1. R. Alur and M. Yannakakis. Model checking of message sequence charts. In *CONCUR 1999*, volume 1664 of *LNCS*. Springer, 1999.
2. H. Ben-Abdallah and S. Leue. Syntactic detection of process divergence and non-local choice in message sequence charts. In *TACAS 1997*, volume 1217 of *LNCS*. Springer, 1997.
3. H. Ben-Abdallah and S. Leue. Mesa: Support for scenario-based design of concurrent systems. In *TACAS 1998*, volume 1384 of *LNCS*. Springer, 1998.
4. B. Genest. Compositional message sequence charts (CMSCs) are better to implement than MSCs. In *TACAS 2005*, volume 3340 of *LNCS*. Springer, 2005.
5. B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state high-level MSCs: Model-checking and realizability. In *ICALP 2002*, volume 2380 of *LNCS*. Springer, 2002.
6. E. Gunter, A. Muscholl, and D. Peled. Compositional message sequence charts. In *TACAS 2001*, volume 2031 of *LNCS*. Springer, 2001.
7. J. G. Henriksen, M. Mukund, K. Narayan Kumar, and P. S. Thiagarajan. On message sequence graphs and finitely generated regular MSC languages. In *ICALP 2000*, volume 1853 of *LNCS*. Springer, 2000.
8. ITU-TS Recommendation Z.120: Message Sequence Chart 1999 (MSC99), 1999.
9. H. Neukirchen. *MSC2000 Parser*. CS Dept., University of Göttingen.
10. *Grappa (Version 1.2)*. `http://www.research.att.com/~john/Grappa/`.
11. MSCan. `http://www-i2.informatik.rwth-aachen.de/MSCan/`.
12. MSC Execute. `http://www-i2.informatik.rwth-aachen.de/MSCExecute/`.