Deciding Properties for Message Sequence Charts

Anca Muscholl[1], Doron Peled[2] and Zhendong Su[3]

[1] Institut für Informatik, Universität Stuttgart,
Breitwiesenstr. 20-22, 70565 Stuttgart, Germany
[2] Bell Laboratories, Lucent Technologies, 600 Mountain Av., Murray Hill, NJ 07974,
and Carnegie Mellon University, School of Computer Science, Pittsburgh, PA,
15213-3891, USA
[3] EECS Department, University of California, Berkeley, CA 94710-1776, USA

Abstract. Message sequence charts (MSC) are commonly used in designing communication systems. They allow describing the communication skeleton of a system and can be used for finding design errors. First, a specification formalism that is based on MSC graphs, combining finite message sequence charts, is presented. We present then an automatic validation algorithm for systems described using the message sequence charts notation. The validation problem is tightly related to a natural language-theoretic problem over *semi-traces* (a generalization of Mazurkiewicz traces, which represent partially ordered executions). We show that a similar and natural decision problem is undecidable.

1 Introduction

Message sequence charts (MSC) are a notation widely used for the early design of communication protocols. With its graphical representation, it allows to describe the communication skeleton of a protocol by indicating the messages that are sent between its different processes. Using message sequence charts one can document the *features* of a system, and the way its parts interact. Although MSCs often do not contain the full information that is needed for implementing the described protocols, they can be used for various analysis purposes. For example, one can use MSCs to search for missing features or incorrect behaviors. It is possible to detect mistakes in the design, e.g., the existence of *race conditions* [1] or *non-local choice* [2]. Another task that is often done using MSCs is providing 'feature transparence', namely upgrading a communication system in a way that all the previous services are guaranteed to be supported.

In recent years MSCs have gained popularity and interest. An international committee (ITU-Z 120 [7]) has been working on developping standards for MSCs. Some tools for displaying MSCs and performing simple checks were developed [1,8]. We model systems of MSCs, allowing a (possibly infinite) family of (finite or infinite) executions. Each execution consists of a finite or infinite set of *send* and *receive* events, together with a partial (causal) order between them. Such a system is denoted using *MSC graphs*, where individual MSCs are combined to form

a branching and possibly looping structure. Thus, an MSC graph describes a way of combining partially ordered executions of events.

We suggest in this paper a specification formalism for MSC properties based on directed graphs: each node of the graph consists of a template, which includes a set of communication events, and the causal order between them. We study three alternative semantics for the specification by MSC graphs:

- Using the same semantics as for an MSC system. Namely, each maximal sequence corresponds exactly to one execution.
- With gaps, i.e., as a template, where only part of the events (and the order between them) is specified. Moreover, choices in the specification graph correspond to different possible ways to continue the execution.
- Again with gaps, but with choices corresponding to conjunctions. Namely an execution matching the specification must include all the events in every possible path of the specification, respecting the associated causal orders.

The main focus of this paper is on developping an algorithm for deciding whether there are executions of the checked system of MSCs that match the specification. Such an execution is considered as a 'bad' execution and if exists it should be reported as a counter-example for the correctness of the system. For the first semantics we show in Section 5 that the matching problem is undecidable. For the last two problems we provide algorithms and we show them to be NP-complete, see Section 4. In the special case of matching two single MSCs we provide a deterministic polynomial time algorithm, improving the result of [8], see Section 3. The complexity of related problems has been studied for pomset languages [6]. In contrast, in [6] only finite pomset languages are studied (however, over a richer structure).

The matching problem can also be represented as a decision problem for *semi-traces* [4]. A semi-trace is a set of words that is obtained from some word by means of (not necessarily symmetric) rewriting rules. These rules allow commuting pairs of adjacent letters. A semi-trace language is a set of words closed under these given rewriting rules. We provide a natural transformation from MSCs to semi-traces. This allows explaining our decidability result as a decision problem on rational languages of semi-traces. One surprising consequence of this translation is that it applies in the same way to two rather different communication semantics for a natural subclass of MSCs: that of asynchronous fifo communication and that of synchronous (handshake) communication.

Work is in progress to add the proposed validation framework to a toolset that was developed for manipulating MSCs [1]. This paper concludes with several open problems and suggested work.

2 Charts and MSC Graphs

In this section, we introduce message sequence charts (MSC) and MSC graphs, as well as the matching problem.

Definition 1 (MSC). A message sequence chart M is a quintuple $\langle E, <, L, T, \mathcal{P}\rangle$ where E is a set of events, $< \subseteq \ E \times E$ is an acyclic relation, $\mathcal{P}$ is a set of processes, $L : E \to \mathcal{P}$ is a mapping that associates each event with a process, and $T : E \to \{s, r\}$ is a mapping that describes the type of each event (*send* or *receive*).

The order relation $<$ is called the *visual ordering* of events and it is obtained from the syntactical representation of the chart (e.g. represented according to the standard syntax ITU-Z 120). It is called 'visual' since it reflects the graphical representation of MSCs. We distinguish between two types of visual ordering as follows. We let $<_c = \{(e, e') \mid T(e) = s\,, T(e') = r$ and e, e' are the send and receive events of the same message$\}$ denote the message ordering. Furthermore, for $P \in \mathcal{P}$ let $E_P = \{e \mid e \in E \ \wedge \ L(e) = P\}$, i.e., E_P is the set of events that belong to process P. We define the relation $<_P = < \cap (E_P \times E_P)$ that represents the ordering between events of P only. Then the visual order $<$ is the union of these orders, i.e., $< = <_c \cup (\bigcup_{P\in\mathcal{P}} <_P)$.

Thus, for two events e and f, we have $e < f$ if and only if one of the following holds:

- e and f are the send and receive event of the same message. In this case, we call e and f a *message pair*.
- e and f belong to the same process P, with e appearing before f on the process line. This imposes a total order among all events of P, for every process P.

In general, the visual order provides more ordering than intended by the designer. Therefore we associate with every chart a causal structure providing the intended ordering. Causal structures are related to *pomsets* [11], *event structures* [9], and *traces* [5]. A causal structure is obtained from an MSC by means of a given semantics. Formally, the causal structure of an MSC M is a quintuple $\mathrm{tr}(M) = \langle E, \prec, L, T, \mathcal{P}\rangle$, where the only component that differs from the definition of an MSC is the relation $\prec$, called the *precedence order* of events. For two events e and f, we have $e \prec f$ if and only if event e must terminate before event f starts. The transitive closure $\prec^*$ of $\prec$ is called the *causal order*. Events which are not causally ordered can occur independently of each other.

The precedence order of events is defined by a set of semantic rules. As the semantics used throughout the paper, we give below the set of rules for an architecture with fifo queues. This means that every one-directional communication between two processes is done through a fifo channel. For this architecture we have in the visual order for each message pairs $e <_c f$ and $e' <_c f'$ with $e <_P e'$ and $L(f) = L(f') = P'$ also $f <_{P'} f'$. Then, for two events e and f, let $e \prec f$ for the *fifo semantics* if one of the following holds:

1. Two sends from the same process:

$$T(e) = T(f) = s \ \wedge \ e <_P f \ \text{ for some process } P\,.$$

2. A message pair: $T(e) = s \ \wedge \ T(f) = r \ \wedge \ e <_c f$.

3. Messages ordered by the fifo queue:

$$T(e) = T(f) = r \ \wedge \ e <_P f \ \text{ for some process } P \ \wedge$$
$$\exists e',\, f' \,(e' <_c e \ \wedge \ f' <_c f \ \wedge \ e' <_{P'} f' \ \text{ for some process } P')\,.$$

4. A receive precedes a send on the same process line:

$$T(e) = r \ \wedge \ T(f) = s \ \wedge \ e <_P f \ \text{ for some process } P\,.$$

Remark 2. For a causal structure $\mathcal{O} = \langle E, \prec, L, T, \mathcal{P}\rangle$ we use the usual notation $e\downarrow$ for the downward closure of an event $e \in E$ w.r.t. the partial order of $\mathcal{O}$, i.e. $e\downarrow = \{f \in E \mid f \prec^* e\}$. The notion of a minimal element e in $\mathcal{O}$ is also standard, meaning that $e' \prec^* e$ implies $e' = e$. We denote by $\min(\mathcal{O})$ the set of minimal elements of the partial order of $\mathcal{O}$.

Note that the following relation between configurations associated to a message pair holds under the fifo semantics:

Lemma 3. *Let $e <_c f$ be a message pair. Then we have under the fifo semantics:*

$$f\downarrow = e\downarrow \cup \{f_1 \in E \mid T(f_1) = r,\, L(f_1) = L(f) \text{ and}$$
$$e_1 <_c f_1 \text{ for } e_1 \text{ with } e_1 \preceq e,\, T(e_1) = s,\, L(e_1) = L(e)\}\,.$$

2.1 Templates and the Matching Problem

An MSC M *matches* an MSC N (or is *embedded* in N) if the chart N respects the causal order on the events specified by M. (Clearly, matching is defined with respect to a given semantics.) The MSC M is called a *template MSC* and it represents the specification, whereas the MSC N is called a *system* MSC. For matching M against N it suffices to consider the reduced partial order of M. Moreover, a template is viewed as a possibly partially specified execution of the system. The actual executions may contain additional messages, which may induce additional ordering.

Definition 4 (Matching a template with an MSC). Under a given semantics, a template M with the causal structure $\mathrm{tr}(M) = \langle E_M, \prec_M, L_M, T_M, \mathcal{P}_M\rangle$ matches a chart N with the causal structure $\mathrm{tr}(N) = \langle E_N, \prec_N, L_N, T_N, \mathcal{P}_N\rangle$ if and only if $\mathcal{P}_M \subseteq \mathcal{P}_N$ and there exists an injective mapping (called *embedding*) $h : E_M \to E_N$ such that

- for each $e \in E_M$, we have $L_N(h(e)) = L_M(e)$ and $T_N(h(e)) = T_M(e)$ (preserving processes and types), and
- if $e_1 \prec_M e_2$ then $h(e_1) \prec_N h(e_2)$ (preserving the causal order).

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ denote the set of processes. For an event $e \in E$ we are often interested in its 'message type' $\mathrm{msg}(e)$ and we let $\mathrm{msg}(e) = s_{ij}$, if e is a send event from P_i to P_j, and $\mathrm{msg}(e) = r_{ij}$ if e is a receive event of P_j from P_i, respectively. Let $\mathrm{msg}(M) = \{\mathrm{msg}(e) \mid e \in E_M\}$.

Note that under the fifo semantics the injectivity of the embedding is already implied by the two other properties in the definition above. Moreover, under this semantics we have a simpler characterization of embeddings, which takes into account just message types:

Lemma 5. *Let M, N denote two MSCs and let $h : M \to N$ be a mapping. Then h is an embedding from M to N if and only if the following conditions hold for any two events $e, f \in E_M$:*

1. *If (e, f) is a message pair, then $(h(e), h(f))$ is also a message pair between the same processes.*
2. *Let $e \prec_M f$ such that (e, f) is not a message pair (thus, e, f are on the same process). Then $msg(h(e)) = msg(e)$, $msg(h(f)) = msg(f)$ and $h(e) <_N h(f)$ holds in the visual order $<_N$ of N.*

Let $\mathcal{M}$ denote the class of finite message sequence charts. Let $M_i = \langle E_i, <_i, L_i, T_i, \mathcal{P}_i \rangle$ be two MSCs, $i = 1, 2$. The (syntactic) concatenation of M_1 and M_2, denoted $M_1 M_2$, is defined by letting $M_1 M_2 = \langle E_1 \dot{\cup} E_2, <, L, T, \mathcal{P}_1 \cup \mathcal{P}_2 \rangle$ with $L|_{E_i} = L_i, T|_{E_i} = T_i$ and $< = <_1 \cup <_2 \cup \{(e, e') \mid e \in E_1, e' \in E_2, L(e) = L(e')\}$. Here, $E_1 \dot{\cup} E_2$ means the disjoint union of the event sets of M_1 and M_2. The concatenation of an infinite sequence $M_1, M_2, \ldots$ is defined in an analogous way. Message sequence graphs (MSC graphs, sometimes called *high-level MSCs* [7]), are used to compose MSCs to larger systems. Equivalently, one can compose MSCs using rational operations, i.e. union, concatenation and iteration. MSC graphs are finite directed graphs where each node of the graph is associated with a finite MSC [1].

Definition 6 (MSC graph). An MSC graph N is a quadruple $\langle \mathcal{S}, \tau, s_0, c \rangle$ where $\langle \mathcal{S}, \tau, s_0 \rangle$ is a finite, directed graph with states set $\mathcal{S}$, transition relation $\tau \subseteq \mathcal{S} \times \mathcal{S}$ and starting state $s_0 \in \mathcal{S}$. The mapping $c : \mathcal{S} \to \mathcal{M}$ assigns to each node a finite MSC.

Let $\xi = s_1, s_2, \ldots$ be a (possibly infinite) path in N, i.e. $(s_i, s_{i+1}) \in \tau$ for every i. The execution (MSC) defined by ξ is given by $c(\xi) = c(s_1)c(s_2)\ldots$.

In order to distinguish MSC graphs from finite MSCs we denote throughout the paper a finite MSC (not bounded to any MSC graph) as a *single MSC*.

In an MSC graph $N = \langle \mathcal{S}, \tau, s_0, c \rangle$, a path ξ is called *maximal* if it begins with the starting state s_0 and it is not a proper prefix of another path. Notice that a maximal path can be either infinite or finite. Let also $\mathrm{msg}(N) = \cup_{s \in \mathcal{S}} \mathrm{msg}(c(s))$.

Fig. 1 shows an example of an MSC graph where the state in the upper left corner is the starting state. Note that the executions of this system are either finite or infinite. Also note that the events of receiving messages of fail and report are not causally ordered.

Definition 7 (Matching paths). Let ξ_1 and ξ_2 be two finite or infinite paths in some MSC graphs. Then ξ_1 matches ξ_2 if $c(\xi_1)$ matches $c(\xi_2)$.

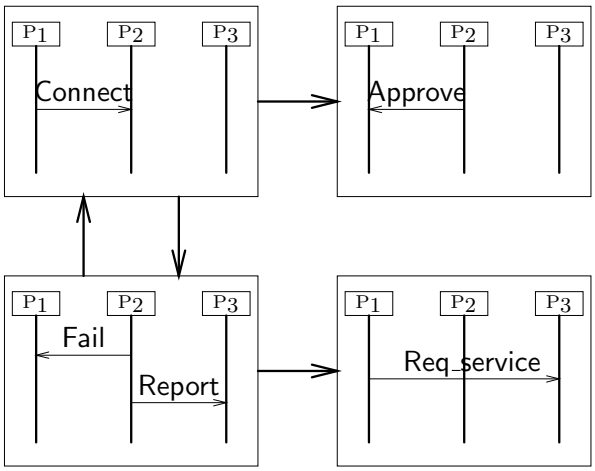

Fig. 1. A system MSC graph.

A *strongly connected component* C of a directed graph $\langle \mathcal{S}, \tau \rangle$ is a subset $C \subseteq \mathcal{S}$ such that for any $u, v \in C$, there is a nonempty path from u to v. A maximal strongly connected component is a strongly connected component which is maximal w.r.t. set inclusion.

3 Matching a Template

In this section, we consider the problem of matching a single template MSC with an MSC graph. As a first result, we show that we can check whether a template can be embedded into a single MSC in polynomial time. (Recall that we assume that the fifo semantics is used.) This algorithm refines the result of [8], where a PSPACE algorithm was exhibited without specifying the semantics. The present matching algorithm is based on the simple observation that it suffices to match a suitable minimal send event and the corresponding receive event with the first occurrence of a message pair of the same type.

Proposition 8. *Let* $M = \langle E_M, <_M, L_M, T_M, \mathcal{P}_M \rangle$, $N = \langle E_N, <_N, L_N, T_N, \mathcal{P}_N \rangle$ *be single MSCs. For each event* $e \in \min(tr(M))$ *which is minimal w.r.t.* $\prec_M^*$ *let* $\mu(e) \in E_N$ *denote the first event in* N *with* $msg(e) = msg(\mu(e))$. *Choose* $e_0 \in \min(tr(M))$ *be such that* $\mu(e_0)$ *is minimal within the set* $\{\mu(e) \mid e \in \min(tr(M))\}$. *Let* f_0, *resp.* f_0', *denote the corresponding receive events of* e_0, *resp.* $\mu(e_0)$. *Let* $M' = M \setminus \{e_0, f_0\}$ *and* $N' = N \setminus \{g' \in E_N \mid g' \prec_N^* f_0'\}$. *Then* M *matches* N *if and only if* e_0 *is well-defined and* M' *matches* N'. *Moreover, if* $h' : M' \to N'$ *is an embedding of* M' *into* N', *then* $h' \cup \{e_0 \mapsto \mu(e_0), f_0 \mapsto f_0'\}$ *is an embedding of* M *into* N.

Proof. Note first that all minimal elements of $\mathrm{tr}(M)$ are send events. Suppose that M matches N via $h : M \to N$, where $h(e_0) \neq \mu(e_0)$ (hence, $h(f_0) \neq f_0'$). Let $e_0' := \mu(e_0)$ and let $\hat{h}$ be given by $\hat{h}(e_0) = e_0'$, $\hat{h}(f_0) = f_0'$ and $\hat{h}(g) = h(g)$ for every $g \notin \{e_0, f_0\}$. Now, if $e_0 \prec_M g$, then $h(e_0) \prec_N h(g)$ and hence also $e_0' \prec_N h(g)$, since $e_0' \prec_N h(e_0)$ and $e_0', h(e_0)$ have the same message type. A similar argument holds for $f_0 \prec_M g$, which shows that $\hat{h}$ is again an embedding from M to N. In order to show that M' matches N' it suffices to show that $\hat{h}(E_M) \cap \{g' \in E_N \mid g' \prec_N^* f_0'\} = \{e_0', f_0'\}$. Assume the contrary, i.e. there exists $g \in E_M$ such that $\hat{h}(g) \prec_N^* f_0'$ and $\hat{h}(g) \notin \{e_0', f_0'\}$. Since every receive event is preceded by its corresponding send event, we may assume that $T_M(g) = s$, i.e. g is a send event. Let $e_1 \in \min(\mathrm{tr}(M))$ be a minimal event with $e_1 \prec_M^* g$, then $\hat{h}(e_1) \prec_N^* \hat{h}(g) \prec_N^* f_0'$. By Lemma 3 we obtain that $\hat{h}(e_1) \prec_N^* e_0'$, since e_1 is a send event. By the definition of μ we have $\mu(e_1) \prec_N^* \hat{h}(e_1)$, hence $\mu(e_1) \prec_N^* e_0'$. Thus, by the choice of e_0 we obtain $\mu(e_1) = e_0'$. Therefore, $e_0' \prec_N^* \hat{h}(g) \prec_N^* f_0'$, which yields $e_0' = \hat{h}(g)$ due to $T_M(g) = s$, contradiction.

Suppose finally that M' matches N' via h' and consider some event g in M'. If $e_0 \prec_M g$, then we also have $e_0' \prec_N h'(g)$, since h' preserves message types and $h'(g) \in E_{N'}$. Similarly, $f_0 \prec_M g$ implies $e_0' \prec_N h'(g)$, which shows that $h' \cup \{e_0 \mapsto e_0', f_0 \mapsto f_0'\}$ is an embedding of M into N.

Remark 9. Proposition 8 yields an embedding algorithm, mapping the events of M in such a way that minimal events are mapped first, to the first event with the same type. This algorithm is of linear complexity if we keep $\min(\mathrm{tr}(M))$, resp. $\{\mu(e) \mid e \in \min(\mathrm{tr}(M))\}$ in two lists. More precisely, note that on each process of M, resp. N, there is at most one event $e \in \min(\mathrm{tr}(M))$, resp. $\mu(e)$. Moreover, we will record for each process of N the event on that process line which is of the form $\mu(e)$ for some $e \in \min(\mathrm{tr}(M))$, if there is one on that process. This additional information is needed in order to update the set of minimal elements of $\{\mu(e) \mid e \in \min(\mathrm{tr}(M))\}$ in constant time. For the complexity of our algorithm note first that $\min(\mathrm{tr}(M'))$ can be updated in constant time, since at most two new minimal events can occur on $L(e_0)$ and $L(f_0)$. Moreover, for $e_1 \in \min(\mathrm{tr}(M')) \setminus \min(\mathrm{tr}(M))$ we can check whether $\mu(e_1)$ is minimal in $\{\mu(e) \mid e \in \min(\mathrm{tr}(M'))\}$ in constant time, using the additional information mentioned above. This suffices, since $\mu(e_1)$ is a send event and every event preceding it in the visual order is a predecessor in the causal order, too. Hence, $\mu(e_1)$ is not minimal within $\{\mu(e) \mid e \in \min(\mathrm{tr}(M'))\}$ if and only if its process contains an event $\mu(e_2)$, $e_2 \in \min(\mathrm{tr}(M'))$, such that $\mu(e_2) <_{N'} \mu(e_1)$.

Note also that the embedding h suggested by Proposition 8 is actually unique. It is not difficult to show that an event $e \in E_M$ is mapped by h to $e' \in E_N$ if and only if e' is the minimal event w.r.t. $\prec_N^*$ such that $e{\downarrow}$ matches $e'{\downarrow}$.

In the remaining of this section we consider the exact complexity of matching a template MSC with an MSC graph.

Definition 10 (Matching a template with an MSC graph). A template MSC M matches an MSC graph N if M matches some maximal path of N.

Matching a template against an MSC graph actually requires only paths of bounded length to be checked:

Proposition 11. *Let N be an MSC graph and let M be a single template MSC such that M matches N. Then there is a path in N that embeds M and has length at most md, where m is the number of messages in M and d is the maximal length of a simple path in N (i.e. of a path where no node appears twice).*

Proposition 11 yields a non-deterministic algorithm for matching a template with an MSC graph which guesses a path in N and verifies that the template matches the graph. The algorithm is polynomial in the size of the template and the number of nodes in the graph. The proposition below shows that matching is also NP-hard.

Proposition 12. *Matching a single template MSC with an MSC graph is NP-complete, even if the graph is acyclic.*

Proof. It suffices to show that matching is NP-hard. For this, we reduce the satisfiability problem for formulas in conjunctive normal form (CNF-SAT) to the MSC matching problem.

Consider a formula $\bigwedge_{j=1}^{k} C_j$ with clauses (disjunctions) C_j over the variables $x_1, \ldots, x_l$. For each clause C_j we take two processes, P_j and R_j. Let $m(j)$ denote a message from P_j to R_j. Note that the events of different messages $m(i), m(j)$, $i \neq j$, are not causally ordered. Then the template M is given as $M = m(1) \cdots m(k)$. The system graph $N = \langle \mathcal{S}, \tau, s_0, c \rangle$ contains for each variable x_i three states denoted as o_i, p_i and n_i, i.e. $\mathcal{S} = \{o_i, p_i, n_i \mid 1 \leq i \leq l\}$. Let $s_0 = o_1$. The edge set is given by $\tau = \{(o_i, p_i), (o_i, n_i), (p_j, o_{j+1}), (n_j, o_{j+1}) \mid 1 \leq i \leq l, 1 \leq j < l\}$. The assignment of MSC to states is as follows: for every i, $c(o_i) = \emptyset$, $c(p_i) = \{m(j) \mid x_i \text{ occurs in } C_j\}$ and $c(n_i) = \{m(j) \mid \bar{x}_i \text{ occurs in } C_j\}$. That is, $c(p_i)$ contains messages associated to all clauses satisfied by $x_i :=$ true, whereas $c(n_i)$ contains messages associated to all clauses satisfied by $x_i :=$ false. Thus, a maximal path in the MSC graph N corresponds exactly to an assignment of the variables. The single MSC M matches a maximal path of N if and only if the assignment given by the path satisfies all clauses.

4 Matching MSC Graphs

In this section, we discuss our extension of the matching algorithm to deal with MSC graphs. Adopting the same convention for matching two single MSCs, we call one of the MSC graphs the *template* (MSC) graph. The other graph is called the *system* (MSC) graph.

The template graph represents a collection of properties (behaviors), each defined by one of its maximal paths. Then for the or-semantics as defined below, the template corresponds to a non-deterministic choice among these behaviors, so an execution of the system needs to contain at least one of the executions

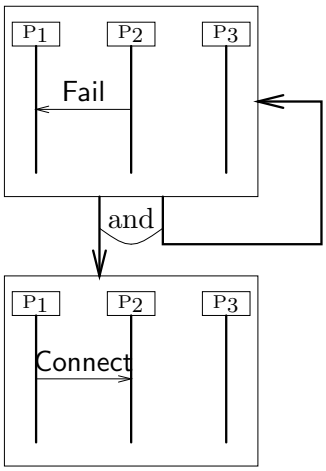

Fig. 2. A template MSC graph.

described by the template. For the and-semantics an execution of the system matches the template if it contains all the executions of the template MSC graph.

Definition 13 (Matching a template graph with a system graph). Let M and N be two MSC graphs.

1. M *or-matches* N if there exists a maximal path ξ' of N and a maximal path ξ of M which matches ξ'.
2. M *and-matches* N if there exists a maximal path ξ' of N such that all maximal paths ξ of M match ξ'.

Consider the and-graph template in Fig. 2. This template matches the system of Fig. 1, since the system may alternate infinitely often between Connect and Fail.

The next lemmas present some fundamental properties of matching paths of MSC graphs. A *subpath* ξ' of a path $\xi = s_0, s_1, \ldots$ in some graph G is a path of G of the form $\xi' = s_{i_0}, s_{i_1}, s_{i_2}, \ldots$ with $i_0 < i_1 < \ldots$. In this case, we denote ξ a *superpath* of ξ'.

Lemma 14. *Let M, N be two MSC graphs and let ξ_1, ξ_2 denote paths in M, N, resp. Let ξ_1 match ξ_2. Then for every subpath ξ_1' of ξ_1 and every superpath ξ_2' of ξ_2, ξ_1' matches ξ_2'.*

Proposition 15. *Let M, N be two MSC graphs and consider an infinite path ξ in M such that every state from ξ occurs infinitely often in ξ. Let C be the strongly connected component of M induced by the states from ξ. Consider also*

an infinite path χ in N and let C' denote the strongly connected component of the states occurring infinitely often in χ. Then the following holds:

1. *ξ matches χ if and only if msg$(C) \subseteq$ msg(C').*
2. *Let K denote a simple cycle within C and suppose that ξ matches χ. Then K^ω matches χ, too (here, K^ω denotes the infinite path $KK\ldots$).*
3. *Let $\hat{K}$ be a cycle containing all states from C'. Then ξ matches χ if and only if ξ matches $\hat{K}^\omega$.*

Proof. Suppose first that ξ matches χ. Then, since embeddings preserve message types, it is easily seen that $\mathrm{msg}(C) \subseteq \mathrm{msg}(C')$. For the converse let $\chi = \chi_0\chi_1\ldots$, with χ_i finite paths such that every χ_i, $i \geq 1$, contains all states from C'. Also, consider a linearization $e_1e_2\ldots$ of $\mathrm{tr}(\xi)$ satisfying the property that for each i, (e_{2i-1}, e_{2i}) is a message pair. We define an embedding h inductively by mapping (e_{2i-1}, e_{2i}) to events from χ_i, $i \geq 1$. More precisely, h maps e_{2i-1} to the first event e' occurring in $c(\chi_i)$ satisfying $\mathrm{msg}(e_{2i-1}) = \mathrm{msg}(e')$. Then, e_{2i} is mapped by h to the corresponding receive event of e'. By Lemma 5 it is easy to check that h preserves the causal order.

The second assertion of the proposition is obtained directly from Lemma 14, whereas the last assertion is a consequence of the first one.

4.1 The Complexity of OR-Matching

The next theorem shows that for or-matching two MSC graphs only finite paths have to be considered for an embedding. More precisely, for the recurrent part of a path only the message types of events are relevant. For a strongly connected component C and a state s we denote below a path from s to some node in C as a path from s to C.

Theorem 16. *Let $M = \langle \mathcal{S}, \tau, s_0, c\rangle$ be a template graph and $N = \langle \mathcal{S}', \tau', s_0', c'\rangle$ be a system graph. Then M or-matches N if and only if either there exists a finite maximal path of M which matches N, or there exist*

- *a simple cycle K in M and a simple path ξ from s_0 to K,*
- *a strongly connected component C' of N and a path χ from s_0' to C',*

such that ξ matches χ and msg$(K) \subseteq$ msg(C').

Proof. Suppose that M or-matches N via an infinite maximal path. Then, by Lemma 14 and Proposition 15(2) we also obtain a path of M of the form $\xi KK\ldots$ which matches N, where K is a simple cycle and ξ is a simple path from s_0 to K. Let ρ denote a path in N such that ξK^ω matches ρ. Moreover, let χ be a minimal prefix of ρ such that ξ matches χ and the corresponding suffix is a strongly connected component of N. Then, by applying Proposition 15(1), we obtain the result.

For the converse we may use again Proposition 15(1) in order to extend the embedding of ξ into χ to an embedding of ξK^ω into a path in N starting with χ.

First note that in Theorem 16 the path χ is in general not simple. But by Proposition 11 its length is bounded by $\text{size}(\xi) \cdot n$, with $\text{size}(\xi)$ denoting the number of messages in ξ, and n denoting the number of states in N. Note also that we can require above that C' is a maximal strongly connected component, due to Lemma 14. Hence, an algorithm based on Theorem 16 would first compute in linear time all maximal strongly connected components of N. Then, for each maximal strongly connected component C' consider the states s of M with $\text{msg}(c(s)) \subseteq \text{msg}(C')$ and the subgraph $M_{C'}$ induced by these states. The algorithm checks whether there is some simple path ξ from s_0 to some strongly connected component of $M_{C'}$ which matches a path χ from s'_0 to C'. (The length of χ is bounded by a polynomial in the size of ξ and the size of N.)

The complexity of the above algorithm basically derives from two problems: one consists of finding all simple paths from the initial node to a given subgraph, and the second one is the problem of matching a single template MSC with an MSC graph. Clearly, Theorem 16 directly yields an NP-algorithm for or-matching. Moreover, by Proposition 12 already the case where the template graph is a single node is NP-hard. Hence, we obtain:

Corollary 17. *The or-matching problem for MSC graphs is NP-complete.*

4.2 The Complexity of AND-Matching

For the and-matching problem we need to deal not only with strongly connected components, but also with states reachable from some strongly connected component. The reason is that some of the events in such states have to be mapped to events belonging to recurrent states in the system graph.

For an MSC graph $M = \langle \mathcal{S}, \tau, s_0, c \rangle$ let $\mathcal{S}_c \subseteq \mathcal{S}$ denote the set of nodes belonging to some strongly connected component of M. For each state $s \in \mathcal{S}$ let us partition the events belonging to the single MSC $c(s)$ associated with s in two sets $c_f(s), c_\omega(s)$ as follows. For each event $e \in c(s)$ let $e \in c_\omega(s)$ if and only if there exist some state $s' \in \mathcal{S}_c$, some event e' in $c(s')$ and a path ξ from s' to s with $e' \prec^*_\xi e$ for the causal order $\prec^*_\xi$ associated to the execution of ξ. We denote by E_ω the set of events $\{e \mid e \in c_\omega(s), s \in \mathcal{S}\}$. The set E_ω can be computed in polynomial time as follows: let $E_\omega := \{e' \mid e' \in c(s'), s' \in \mathcal{S}_c\}$. Then for every $e \notin E_\omega$, $e \in c(s)$, test whether there is some event $e' \in E_\omega$, $e' \in c(s')$, such that s is reachable from s' through a path ξ and $e' \prec_\xi e$ for the execution of that path. Note that $e' \prec_\xi e$ holds if and only if $e' \prec_\chi e$ holds for any other path χ from s' to s. Moreover, by Lemma 3 the condition $e' \prec_\xi e$ can be checked by examining the message types of e, e'. If the test is positive, then let $E_\omega = E_\omega \cup \{e\}$. This step is repeated until no more events can be added. Note also that for every $e \in c_\omega(s)$ and $e' \in c(s)$ with $e \prec^*_{c(s)} e'$, also $e' \in c_\omega(s)$ holds. Moreover, for every message pair $e_1 <_c e_2$ in $c(s)$ we have $e_1 \in c_f(s)$ if and only if $e_2 \in c_f(s)$ (this is easily checked using Lemma 3.) The set $c_f(s)$ together with the visual order inherited from $c(s)$ is thus a single MSC which we also denote by $c_f(s)$ (analogously for $c_\omega(s)$). By the previous remarks we have that the causal order

of $c(s)$ is the same as the causal order of $c_f(s)c_\omega(s)$. Finally, for $s \in \mathcal{S}_c$ we have $c(s) = c_\omega(s)$.

Theorem 18. *Let $M = \langle \mathcal{S}, \tau, s_0, c\rangle$ be a template graph and $N = \langle \mathcal{S}', \tau', s_0', c'\rangle$ be a system graph. Define a mapping $\hat{c} : \mathcal{S} \to \mathcal{M}$ by letting $\hat{c}(s) = c_f(s)$. Let $\hat{M} = \langle \hat{\mathcal{S}}, \hat{\tau}, s_0, \hat{c}\rangle$ denote the MSC graph with states set $\hat{\mathcal{S}} = \{s \in \mathcal{S} \mid \hat{c}(s) \neq \emptyset\} \cup \{s_0\}$ and $(s, s') \in \hat{\tau}$ if and only if $s, s' \in \hat{\mathcal{S}}$ such that $\neg(s = s' = s_0)$ and there is a path $s = s_1, \ldots, s_k = s'$ in M satisfying $\hat{c}(s_i) = \emptyset$ for all $1 < i < k$. Then M and-matches N if and only if there exists a subgraph C' of N and a path χ from s_0' to C' such that*

1. *All paths in $\hat{M}$ match χ.*
2. *If M contains cycles then $msg(E_\omega) \subseteq msg(C')$ and C' is a strongly connected component of N.*

Proof. First, note that the MSC graph $\hat{M}$ is acyclic (since the only possible loop would be a self-loop of s_0, which has been excluded by definition).

Suppose that M and-matches N and consider a path ρ in N such that all maximal paths in M match ρ. If M is acyclic, hence $M = \hat{M}$, then we are done by choosing an appropriate finite prefix χ of ρ. So suppose that $\mathcal{S}_c \neq \emptyset$, then ρ must be infinite. Let C' be the strongly connected component containing exactly the states occurring infinitely often in ρ. Let ξ be a (finite) path from $\hat{M}$. Then it is easy to verify that there exists a path σ in M such that the causal order of the execution of ξ is a prefix of the causal order of the execution of σ. Hence, ξ matches ρ, too. Let χ be a finite prefix of ρ such that all (finite) paths from $\hat{M}$ match χ and the corresponding suffix is a strongly connected component of N. Finally, consider an event e in some $c_\omega(s)$, for some state s. Then there exists for each $n \geq 0$ a path ξ from s_0 to s such that the configuration $e \downarrow$ of the occurrence of e in the last node of ξ contains at least n events. Hence, there is some state s' occurring in ρ infinitely often, such that $\mathrm{msg}(e) = \mathrm{msg}(e')$ holds for some event e' in s'. This concludes one direction of the proof.

Conversely, suppose that M has cycles. Let $\xi = s_0, s_1, \ldots$ be a maximal (finite or infinite) path in M. Note that the causal order associated to the execution $c(\xi)$ of ξ is identical to the causal order of $c_f(\xi)c_\omega(\xi)$, where $c_f(\xi) = c_f(s_0)c_f(s_1)\ldots$ and $c_\omega(\xi) = c_\omega(s_0)c_\omega(s_1)\ldots$. Moreover, $c_f(s_0)c_f(s_1)\ldots$ is a finite MSC since there can be only a finite number of nodes s_i with $c_f(s_i) \neq \emptyset$. Also, $c_f(s_0)c_f(s_1)\ldots$ is the execution of a finite path in $\hat{M}$, thus it matches χ. Since $\mathrm{msg}(E_\omega) \subseteq \mathrm{msg}(C')$ we obtain similarly to Proposition 15 that the MSC $c_\omega(s_0)c_\omega(s_1)\ldots$ matches $\hat{K}^\omega$, for some fixed cycle $\hat{K}$ containing all the states from C'. Thus, ξ matches $\chi\hat{K}^\omega$, which shows the claim.

By the previous theorem we have to consider the problem of and-matching a single MSC against an *acyclic* MSC graph. The next proposition shows that for and-matching an acyclic graph it suffices to look for a mapping which is an embedding for *all* the paths (instead of embedding each path separately).

Proposition 19. *Let M be an acyclic MSC graph and let N be a single MSC. Then M and-matches N if and only if there exists a mapping $g : M \to N$ which is an embedding for* all *paths in M.*

Proof. Suppose that M and-matches N and let g_ξ denote an embedding of a maximal path ξ of M in N. Let Ξ denote the set of all maximal paths of M. Define a mapping $g : M \to N$ by letting $g(e) = \max\{g_\xi(e) \mid \xi \in \Xi, e \text{ occurs on } \xi\}$. Note that for a fixed event e the set $\{g_\xi(e) \mid \xi \in \Xi, e \text{ occurs on } \xi\}$ is totally ordered w.r.t. $\prec_N^*$. This is due to the fifo semantics, since for each e, e' with $\mathrm{msg}(e) = \mathrm{msg}(e')$ we have either $e \preceq e'$ or $e' \preceq e$.

We show that g is an embedding for every path $\xi \in \Xi$. If $e <_c f$ is a message pair in M, then $g(e) <_c g(f)$ holds, due to the fifo semantics. Thus, suppose that $e \prec_\xi f$ and $e \not<_c f$ both hold, where $\prec_\xi$ denotes the causal order associated to the execution of ξ. Let $\chi \in \Xi$ be a path also containing e, f. Note that we have $e \prec_\chi f$ due to M being acyclic. Hence, $g_\chi(e) \prec_N g_\chi(f)$. By the definition of g we finally obtain $g(e) \prec_N g(f)$.

The previous proposition shows that there exists a mapping g for matching all paths in M with N. This yields an NP-algorithm for matching an acyclic MSC graph M with a single MSC (note that after guessing the mapping g we test the embedding property for every pair of events e, f with $e \prec_\xi f$ for some path ξ). We now show that we can even find a canonical mapping deterministically in polynomial time (similar to Proposition 8).

Proposition 20. *Let $M = \langle \mathcal{S}, \tau, s_0, c\rangle$ be an acyclic MSC graph and let $s \in \mathcal{S}$ be a source node, i.e. a node without predecessors. Let $N = \langle E_N, <_N, L_N, T_N, \mathcal{P}_N\rangle$ be a single MSC. Assume that M and-matches N and let $h : c(s) \to N$ be defined by*

$$h(e) = e' \text{ if } e' \text{ is minimal w.r.t. } \prec_N^* \text{ such that } e{\downarrow} \text{ matches } e'{\downarrow}$$

Let $g : M \to N$ be a mapping which is an embedding for all paths from M in N. We define a mapping $g' : M \to N$ by letting $g'|_{c(s)} = h$ and $g'(e) = g(e)$ for every $e \notin c(s)$. Then g' is also a mapping which embeds all paths of M into N.

Proof. It can be easily verified that for every event $e \in c(s)$ and every mapping $g : M \to N$ which is an embedding for all paths in M (in particular for $c(s)$) one has $h(e) \prec_N g(e)$. Therefore, if $e \prec_\xi^* f$ holds for the execution of a nonempty path ξ from s to s' for two events e, f with $e \in s$ and $f \in s'$, then also $h(e) \prec_N g(f)$ holds.

Proposition 20 yields a polynomial-time algorithm for matching an acyclic and-graph with an MSC defined by a path. We first determine for each node s and for each event $e \in c(s)$ the immediate predecessor events of e (w.r.t. the causal order) located in s and in the nodes preceding s. Then we embed a source node s of M and iterate this procedure with $M \setminus \{s\}$. When processing the

current node s events in $c(s)$ are mapped according to the partial order (starting with minimal elements) as suggested by Proposition 8. That is, a suitable event $e \in \min(\mathrm{tr}(M))$ is mapped to the minimal event e' of the same type in N, such that $e' \downarrow$ contains all events to which the immediate predecessor events of e were mapped to.

Together with Theorem 18 we obtain an NP-algorithm for the and-matching problem by first guessing a subgraph C' of the system graph N and a path χ from the starting node of N to some node in C'. Then we verify deterministically that the acyclic MSC graph $\hat{M}$ defined in Theorem 18 and-matches the single MSC corresponding to χ. Note that due to Proposition 19 we can bound the length of χ by a polynomial in the number of messages in $\hat{M}$ and the number of nodes in N. Together with Proposition 12 we obtain:

Corollary 21. *The and-matching problem for MSC graphs is NP-complete.*

5 An Undecidable Problem

The matching problems considered previously were based on the paradigm that templates represent partial specifications of system behaviors. We show below that if we require that templates represent exact behaviors, then the or-matching problem is undecidable.

For the fifo semantics considered in this paper we show first that considering a message pair as a single letter we obtain an isomorphism between the causal orders of a natural subclass of message sequence charts and partial orders of *semi-traces*. Semi-traces are objects known from the algebraic study of concurrency (for a survey on semi-traces see Chapter 12 in [5]).

Formally, assume that $\mathcal{P} = \{P_1, \ldots, P_m\}$ is the set of processes. We associate an alphabet $\Sigma = \{m_{ij} \mid 1 \leq i \neq j \leq m\}$ and a *non-commutation relation* $\mathrm{SD} \subseteq \Sigma \times \Sigma$, $\mathrm{SD} = \{(m_{ij}, m_{ik}) \mid j \neq i \neq k\} \cup \{(m_{ij}, m_{jk}) \mid i \neq j \neq k\}$. The idea underlying SD is to consider in the precedence order the order between sends on the same process and receives ordered by the fifo condition (m_{ij}, m_{ik}), and receives followed by sends on the same process line (m_{ij}, m_{jk}). The complementary relation, $\mathrm{SI} = (\Sigma \times \Sigma) \setminus \mathrm{SD}$, called *semi-commutation relation*, yields a rewriting system $\{ab \to ba \mid (a, b) \in \mathrm{SI}\}$, which will be also denoted by SI. A semi-trace $[w]$ is a set of words, $[w] = \{v \in \Sigma^* \mid w \stackrel{*}{\to}_{\mathrm{SI}} v\}$. The concatenation of two semi-traces $[u], [v]$ is defined as $[u][v] = [uv]$. It is an associative operation and the set of all semi-traces over (Σ, SI) together with the concatenation is a monoid with identity $1 = [\epsilon]$, which is denoted $(\mathbb{M}(\Sigma, \mathrm{SI}), \cdot, 1)$. Note also that the relation SD is reflexive. Moreover, $[w] = [w']$ holds if and only if w can be rewritten into w' by using symmetric rules only.

In the next proposition we show that a naturally arising subclass of MSCs can be identified with semi-traces. We restrict our consideration to MSCs satisfying the condition that in the visual representation no two message lines intersect. We denote this subclass as *ordered* MSCs. Clearly, ordered MSCs satisfy the fifo condition on the visual order. Note also that the syntactic concatenation of

MSCs induces a concatenation operation for the associated causal orders, which is associative.

Proposition 22. *Let $\mathcal{M}_o$ denote the set of ordered MSCs over the set of processes $\mathcal{P} = \{P_1, \ldots, P_m\}$ and let (Σ, SI) be defined as above. Then the monoid of causal orders over $\mathcal{M}$ is isomorphic to $(\mathbb{M}(\Sigma, SI), \cdot, 1)$.*

Proof. Let $M = \langle E, <, L, T, \mathcal{P}\rangle$ and define a homomorphism $h : E_M^* \to \Sigma^*$ by letting $h(e) = m_{ij}$, if e is a send event from P_i to P_j, and $h(e) = \lambda$ if e is a receive event. To M we associate a language t_M over Σ^*:

$$t_M = \{h(z) \mid z \in E_M^* \text{ is a linearization of } \prec_M^*\}$$

Then we can show that t_M is a semi-trace over (Σ, SI). For this, we first define a linearization $z_0 \in E_M^*$ of M inductively by choosing some message pair (e, f) of M satisfying

- e is minimal w.r.t. the visual order $<$ in M
- for every $g \in E_M$: $g < f \;\Leftrightarrow\; g = e$

and letting $z_0 = efz_0'$, where z_0' is defined accordingly for $M' := M \setminus \{e, f\}$. (Note that the existence of e, f as above is due to M being an ordered MSC.) Then we claim that $t_M = [h(z_0)]$, i.e. t_M is the semi-trace associated to $h(z_0)$. We show this by induction on the length of t_M. For lack of space, the details are left to the full version of the paper.

Traces [5] result from in symmetric rewriting rules, i.e. both SI and SD are symmetric relations. For the trace monoid given by the rules $ab = ba$, $cd = dc$ it is known that one cannot decide for given regular languages $L_1, L_2 \subseteq \{a, b, c, d\}^*$ whether $[L_1] \cap [L_2]$ is empty [3], where $[L] = \cup_{u \in L}[u]$ denotes the closure of L under $\overset{*}{\to}_{\mathrm{SI}}$.

Proposition 23. *Let M, N be two MSC graphs. Then it is undecidable whether there exist two maximal paths ξ_1 in M, ξ_2 in N such that the associated MSCs m_1, m_2 have the same causal order under the fifo semantics.*

Proof. We consider four processes, $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$ and we denote by s_a, r_a a message pair from P_1 to P_2, resp. by s_b, r_b a message pair from P_2 to P_1. Dually, s_c, r_c denotes a message pair from P_3 to P_4, whereas s_d, r_d is a message pair from P_4 to P_3. Then we associate to each letter a, b, c, d an MSC as given by the mapping h, with $h(x) = s_x r_x$, for $x \in \{a, b, c, d\}$. Moreover, h induces a homomorphism from $\{a, b, c, d\}^*$ to $\mathcal{M}$.

Note that for any word u over $\{a, b, c, d\}$ the partial order $\mathrm{tr}(h(u))$ consists of two totally ordered sequences, one over events between processes P_1 and P_2, the other over events between P_3 and P_4. Moreover, these total orders are completely independent. Viewed as a mapping from $\mathbb{M}(\Sigma, \mathrm{SI})$ to $\mathrm{tr}(\mathcal{M})$, h is injective. This, together with [3], concludes our proof.

Let us comment our results in the context of semi-trace languages. One cannot decide the emptiness of the intersection of two MSC graphs since given two regular languages $L, K \subseteq \Sigma^\omega$ and a semi-commutation relation SI over Σ, the question whether the intersection $[L] \cap [K]$ is nonempty is undecidable. In contrast, the or-matching problem of Section 4.1 can be expressed as a very particular instance of the above problem. Before going into some details, let us fix notations. For a language $L \subseteq \Sigma^*$, we denote by $L \sqcup\!\sqcup \Sigma^*$ the shuffle of L and Σ^*, i.e. the language $\{u_1 v_1 u_2 v_2 \cdots u_n v_n \mid u_1 u_2 \cdots u_n \in L, v_i \in \Sigma^*\}$. The shuffle $L \sqcup\!\sqcup \Sigma^\omega$ for $L \subseteq \Sigma^* \cup \Sigma^\omega$ is defined analogously.

Formally, the or-matching problem for the semantics with gaps is equivalent to the question whether the intersection $[L \sqcup\!\sqcup \Sigma^\omega] \cap [K]$ is empty or not, for regular languages $L, K \subseteq \Sigma^\omega$. The crucial point now is that $[L \sqcup\!\sqcup \Sigma^\omega]$ has a very particular form. Suppose without loss of generality that $L = UV^\omega$, with $U, V \subseteq \Sigma^*$ regular languages such that every element of V has the same alphabet $A \subseteq \Sigma$. Then $UV^\omega \sqcup\!\sqcup \Sigma^\omega = (U \sqcup\!\sqcup \Sigma^*)\ \mathrm{Inf}(A)$, with $\mathrm{Inf}(A) = \{u \in \Sigma^\omega \mid |u|_a = \infty, \forall a \in A\}$. Moreover, $[UV^\omega \sqcup\!\sqcup \Sigma^\omega] = [(U \sqcup\!\sqcup \Sigma^*)]\ \mathrm{Inf}(A)$. But it is easy to check that $U \sqcup\!\sqcup \Sigma^*$ is a very simple regular language, a finite union of languages of the form $\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \cdots a_k \Sigma^*$ for some letters $a_i \in \Sigma$. (This family of languages corresponds exactly to level '1/2' in the concatenation hierarchy of Straubing-Thérien [10]). Finally, $[\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \cdots a_k \Sigma^*] = \cup_{a_{i_1} \cdots a_{i_k} \in [a_1 \cdots a_k]} \Sigma^* a_{i_1} \Sigma^* \cdots a_{i_k} \Sigma^*$.

6 Conclusion

In this paper we presented specification and verification methods for MSCs, which employ languages of *partially ordered executions*. We were interested in the problem of deciding whether there is an execution of the given MSC system that matches the specification. We considered three alternative semantics and showed that the matching problem under both the or-semantics and the and-semantics is NP-complete. Under a semantics which allows no gaps in the specification the matching problem becomes the intersection of two MSC graphs. We showed that this problem is undecidable. Some open directions for further research include extending the framework by allowing and/or-graphs and negation, expressing the finite occurrence of certain events, and obtaining complementable specification formalisms.

References

1. R. Alur, G. Holzmann, and D. Peled. An analyzer for message sequence charts. *Software Concepts and Tools*, 17(2):70–77, 1996.
2. H. Ben-Abdallah and S. Leue. Syntactic detection of process divergence and non-local choice in message sequence charts. In E. Brinksma, editor, *Proceedings of theTools and Algorithms for the Construction and Analysis of Systems, Third International Workshop, TACAS'97*, number 1217 in Lecture Notes in Computer Science, pages 259–274, Enschede, The Netherlands, 1997. Springer.
3. J. Berstel. *Transductions and context-free languages.* Teubner Studienbücher, Stuttgart, 1979.

4. M. Clerbout and M. Latteux. Partial commutations and faithful rational transductions. *Theoretical Computer Science*, 34:241–254, 1984.
5. V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995.
6. J. Feigenbaum, J. Kahn, and C. Lund. Complexity results for pomset languages. *SIAM Journal Disc. Math.*, 6(3):432–442, 1993.
7. ITU-T Recommendation Z.120, Message Sequence Chart (MSC), March 1993.
8. V. Levin and D. Peled. Verification of message sequence charts via template matching. In *TAPSOFT (FASE)'97, Theory and Practice of Software Development*, volume 1214 of *Lecture Notes in Computer Science*, pages 652–666, Lille, France, 1997. Springer.
9. M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, part 1. *Theoretical Computer Science*, 13:85–108, 1981.
10. J.-E. Pin. Syntactic semigroups. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1, pages 679–738. Springer, Berlin-Heidelberg-New York, 1997.
11. V. R. Pratt. Modelling concurrency with partial orders. *International Journal of Parallel Programming*, 15(1):33–71, 1986.