Matching Specifications for Message Sequence Charts

Anca Muscholl

Institut für Informatik, Universität Stuttgart,
Breitwiesenstr. 20-22, 70565 Stuttgart, Germany

Abstract. Message sequence charts (MSC) are widely used in the early design of communication protocols. They allow describing the communication skeleton of a system. We consider a basic verification task for MSC graphs, the matching problem via MSC templates. We characterize the complexity of checking properties which are expressible by and-or templates, resp. by LTL formulas. Both problems are shown to be PSPACE-complete.

1 Introduction

Message sequence charts (MSC) are an ITU-standardized notation which is widely used for the early design of communication protocols. They describe in a graphical way how message passing is supposed to be performed between concurrent processes. Although MSCs 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 detect mistakes in the design, like *race conditions* [1] or *non-local choice* [2]. Some tools for performing basic verification tasks have been developed, [1, 3].

The problem considered in this paper is to verify certain properties of MSC graphs by means of template (MSC) graphs. Matching a template graph with a system graph means that a specific *set of executions in the template* is required to occur in an execution of the system, such that the causal order of events is preserved. The occurrence of executions is meant as an embedding, i.e. it allows gaps. (Actually, determining the existence of an exact mapping is easily seen to be undecidable, [7]). Two semantics have been already considered for template graphs, or-graphs resp. and-graphs, [7].

Here we focuss on specifications (templates) given as and-or MSC graphs, which can be seen as alternating transition systems with an associated causal order related to Mazurkiewicz traces, [5]. Like alternating automata on ω-words [10], it is more convenient to translate specifications into and-or MSC graphs than just or-graphs. The matching problem for and-or template graphs is shown to be PSPACE-complete. However, our proof shows somewhat surprisingly that and-or templates are not more expressive than or-templates (or and-templates). And-or MSC templates just provide a more succinct representation of specifications. They express properties given by shuffle ideals (of finite sets) and liveness conditions. We also consider the problem of specifying templates by LTL formulas, that is by matching the causal orders of sequences which satisfy a given formula. For this semantics we show that deciding the existence of a matching is PSPACE-complete, too. The paper is organized as follows: Sect. 2 gives basic notions and definitions, in Sect. 3 we consider the and-or matching problem, and Sect. 4 deals with the matching problem for LTL specifications.

2 Preliminaries

Definition 1 (MSC). *A message sequence chart $M = (\mathcal{E}, <, \mathcal{P}, L, T)$ is given by a poset $(\mathcal{E}, <)$ of events, a set $\mathcal{P}$ of processes, a mapping $L : \mathcal{E} \to \mathcal{P}$ that associates each event with a process, and a mapping $T : \mathcal{E} \to \{\mathsf{s}, \mathsf{r}\}$ that describes the type of each event (*send *or* receive*).*

The partial order $<$ is called the *visual order* of events and it is obtained from the syntactical representation of the chart (e.g. represented according to the standard syntax ITU-Z 120). The visual order is induced by an acyclic relation $<_c \cup (\bigcup_{P \in \mathcal{P}} <_P)$ that is explained in the following. First, there is a one-to-one correspondence between send events, $\mathcal{E} \cap T^{-1}(\mathsf{s})$, and receive events, $\mathcal{E} \cap T^{-1}(\mathsf{r})$. Let $\mathcal{M}$ denote the graph of this correspondence, i.e. the set of messages. Then we have $e <_c f$ for every message $(e, f) \in \mathcal{M}$ (*message ordering*). Secondly, for every process $P \in \mathcal{P}$ the set $\mathcal{E} \cap L^{-1}(P)$ is totally ordered by $<_P$ (*process line ordering*).

In general, the visual order provides more ordering than intended by the designer. Therefore every chart has an associated causal structure providing the intended ordering [1]. Causal structures are related to *pomsets* [9], *event structures* [8], and *Mazurkiewicz traces* [4]. A causal structure is obtained from a chart by means of a given semantics, which depends on the system architecture. Formally, the causal structure of a chart $M = (\mathcal{E}, <, \mathcal{P}, L, T)$ is given as $\mathrm{tr}(M) = (\mathcal{E}, \prec, \mathcal{P}, L, T)$, where the only difference between M and $\mathrm{tr}(M)$ is the poset $(\mathcal{E}, \prec)$, with $\prec$ denoting the *causal order*. The partial order $\prec$ is generated by a so-called *precedence relation* $\lessdot$, which depends on the implementation. The meaning is that for any two events e and f, we have $e \lessdot f$ if and only if event e must terminate before event f starts.

As the semantics used throughout the paper, we define the precedence relation 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 first impose the following constraint on the visual order: For any messages $(e, f), (e', f') \in \mathcal{M}$ with $e <_P e'$ and $L(f) = L(f') = P'$ for some $P, P' \in \mathcal{P}$ we require that $f <_{P'} f'$. Let $e, f \in \mathcal{E}$ be two events. Then $e \lessdot f$ for the *fifo semantics* if one of the following holds:

1. A send preceded by some event on the same process:
$$T(f) = \mathsf{s} \ \wedge \ e <_P f \ \text{ for some process } P\,.$$
2. Message ordering: $e <_c f$.

3. Messages ordered by the fifo queue:

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

Slightly abusing the notation, we identify $\mathrm{tr}(M)$ with the set of all linearizations of the causal order of $\mathrm{tr}(M)$. We also write $\mathrm{tr}(\alpha)$ for a sequence $\alpha \in \mathcal{E}^\infty$ and mean by that the set of all linearizations of the causal order associated with α.

Given two charts $M_i = (\mathcal{E}_i, <_i, \mathcal{P}, L_i, T_i)$ over the same set of processes we define their product as the chart $M_1 M_2 = (\mathcal{E}_1 \,\dot{\cup}\, \mathcal{E}_2, <, \mathcal{P}, L_1 \,\dot{\cup}\, L_2, T_1 \,\dot{\cup}\, T_2)$, where $< \,=\, <_1 \cup <_2 \cup \bigcup_{P \in \mathcal{P}} (\mathcal{E}_1 \cap L^{-1}(P)) \times (\mathcal{E}_2 \cap L^{-1}(P))$.

Definition 2 (MSC graph). *An MSC graph $N = (\mathcal{S}, \tau, s_0, c, \mathcal{P})$ is given as a finite, directed graph $(\mathcal{S}, \tau, s_0)$ with state set $\mathcal{S}$, transition relation $\tau \subseteq \mathcal{S} \times \mathcal{S}$, source state $s_0 \in \mathcal{S}$, together with a mapping c associating each state s with a finite chart $c(s)$ over the process set $\mathcal{P}$.*

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 chart defined by ξ is given as $c(\xi) = c(s_1)c(s_2)\ldots$. We denote by $tr(\xi)$ the causal structure associated with $c(\xi)$, resp. by $\prec_\xi$ the associated causal order.

A maximal path of N is a path in the graph which starts with the source state and is either infinite or it ends in a sink state.

In order to simplify the presentation we assume that in an MSC graph N every state $s \in \mathcal{S}$ is associated with a single event. This is no restriction, due to the following observation: If $\alpha = \alpha_1 \cdots \alpha_k$ denotes any topological sorting of the visual order of a chart M, then $\mathrm{tr}(\alpha) = \mathrm{tr}(M)$. Let $M = c(s)$ for a state $s \in \mathcal{S}$. We can replace s by a sequence of states $s = s_0 \stackrel{\alpha_1}{\to} s_1 \stackrel{\alpha_2}{\to} \cdots s_k$ and add transitions from s_k to all successors of s.

Definition 3 (Matching). *Under a given semantics, a template M with the causal structure $tr(M) = (\mathcal{E}_M, \prec_M, L_M, T_M, \mathcal{P}_M)$ matches an MSC N with the causal structure $tr(N) = (\mathcal{E}_N, \prec_N, L_N, T_N, \mathcal{P}_N)$ if and only if $\mathcal{P}_M \subseteq \mathcal{P}_N$ and there exists an injective mapping (called* embedding*) $h : \mathcal{E}_M \to \mathcal{E}_N$ such that*

- *for each $e \in \mathcal{E}_M$, we have $L_N(h(e)) = L_M(e)$ and $T_N(h(e)) = T_M(e)$ (preserving processes and types),*
- *if $e_1 \prec_M e_2$ then $h(e_1) \prec_N h(e_2)$ (preserving the causal order), and*
- *if $e <_c f$ then $h(e) <_c h(f)$ (preserving messages).*

A path ξ_1 in a graph M matches a path ξ_2 in a graph N if the chart $c(\xi_1)$ matches the chart $c(\xi_2)$.

Under the fifo semantics it suffices to consider only the type and the location of events. Let $\mathcal{P} = \{P_1, \ldots, P_m\}$ be the set of processes and let $E = \{\mathsf{s}_{ij}, \mathsf{r}_{ij} \mid 1 \le i \ne j \le m\}$. The set E consists of *abstract events* and is related to any set of events $\mathcal{E}$ by means of a mapping ev : $\mathcal{E} \to E$. Consider a message $(e, f) \in \mathcal{M}$ with $L(e) = P_i$, $L(f) = P_j$. Then we let $\text{ev}(e) = \mathsf{s}_{ij}$ and $\text{ev}(f) = \mathsf{r}_{ij}$. Let χ be a path in an MSC graph, then $\text{msg}(\chi) \subseteq E$ denotes the set $\{\text{ev}(e) \mid e \text{ occurs in } c(\chi)\}$. The mapping ev extends to a homomorphism ev : $\mathcal{E}^\infty \to E^\infty$. Assume that α is a topological sorting of $c(\chi)$. Let $\text{ev}(\chi) \in E^\infty = E^* \cup E^\omega$ denote the (finite or infinite) sequence of abstract events $\text{ev}(\alpha)$.

For the remaining of the paper we will refer to the elements of E simply as *events*. All notions introduced so far, e.g. precedence rules, causal order, matchings, can be transferred to the events in E.

3 And-Or MSC Graphs

An and-or MSC graph $M = (\mathcal{S}, \tau, s_0, c, \mathcal{S}_\wedge, \mathcal{S}_\vee, \mathcal{P})$ is given as an MSC graph where the set of states is partitioned in two sets, the set of and-states $\mathcal{S}_\wedge$ and the set of or-states $\mathcal{S}_\vee$.

Definition 4 (And-or MSC graphs). *Let $M = (\mathcal{S}, \tau, s_0, c, \mathcal{S}_\wedge, \mathcal{S}_\vee, \mathcal{P})$ be an and-or MSC graph. A* run *R of M is a tree $R = (X, \to, x_0, \ell, ev)$ with root x_0 and nodes labelled by states, $\ell : X \to \mathcal{S}$, such that:*

1. *$x \to y$ in R implies that $(\ell(x), \ell(y)) \in \tau$.*
2. *Every node $x \in X$ with $\ell(x) = s \in \mathcal{S}_\vee$ has one successor in R, if $\{s' \mid \tau(s, s')\} \neq \emptyset$, otherwise it has no successor.*
3. *Every node $x \in X$ with $\ell(x) = s \in \mathcal{S}_\wedge$ has exactly k successors in R, all labelled differently, where $k = |\{s' \mid \tau(s, s')\}|$.*

Moreover, $ev : X \to E$ labels each node x by the event $ev(e)$, where $e = c(\ell(x))$.

A run R is called maximal *if all leaves are labelled by states without any successors in M and the root is labelled by the source state.*

An and-or MSC graph is an example for alternating transition systems. It can be used for specifying scenarios between a component of a system and the environment. Here, the moves of the environment are modelled as usually as universal moves, i.e. the component is required to meet its specification no matter how the environment behaves. Consider for example the and-or graph in Fig. 1, where state N_1 is an and-state and N_2, N_3 are or-states. For simplicity, the states are labelled by messages, not by single events. The MSC graph expresses that there are infinitely many connections requested by P_2 (from P_1), and all of them are either successful or P_2 gets into an infinite loop requesting a connection. Moreover, each time when process P_2 requests a connection, it expects (several) data transmissions from P_3.

Definition 5 (And-or matching). *Given a template M as an and-or MSC graph and a system N as an MSC graph. We say that the template matches the system if there is some maximal run R of M and a maximal path χ of N such that every path in R matches χ.*

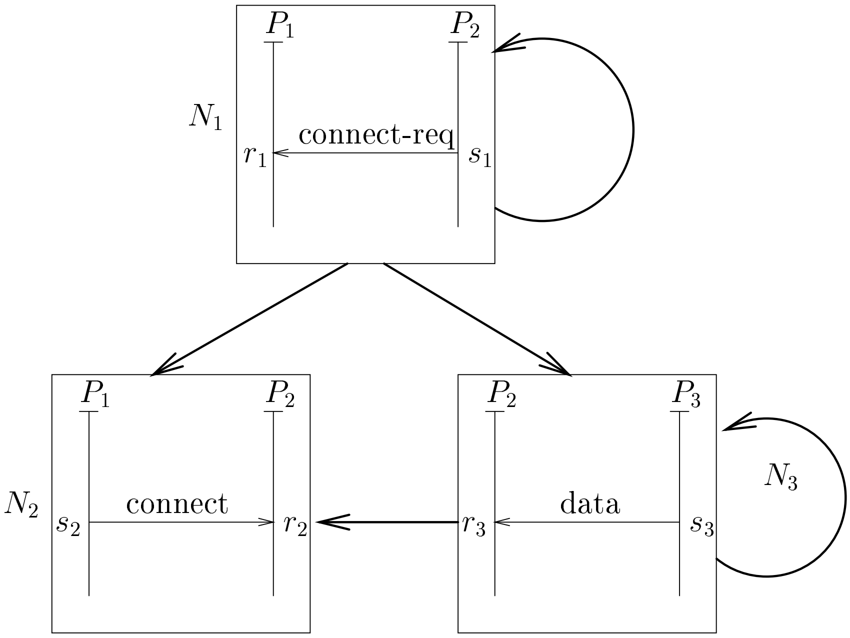

Fig. 1. An and-or MSC graph.

Consider again the and-or template graph M of the above example. In every maximal run of M the and-node N_1 occurs infinitely often. Therefore, a maximal run of M matches an MSC N only if N contains infinitely many events of the same type as s_1, r_1. This holds also for s_2, r_2, since every maximal run of M includes paths $N_1^k N_2$ for every k, and we have $r_1 \prec s_2 \prec r_2$. The situation differs for node N_3. We might have a maximal run where N_3 occurs on every path just once, before N_2. But we do not have any events e in N_1, f in N_3 such that $e \prec f$. Hence, s_3 for example can be mapped for each occurrence of the node N_3 in the run of M to the same event in the MSC N. (The specification given by M is in some sense incomplete, it does not correspond to the intuition given above). Moreover, the combination between states occurring infinitely often and dependency between events makes matching for and-or templates to a quite complex task.

3.1 The Complexity of And-Or Matching

Our main result is that matching and-or template graphs is PSPACE-complete. Thus, and-or matching appears to be computationally more difficult than matching or-templates, resp. and-templates. The latter problems have been shown to be NP-complete, [7].

The next proposition gives the lower bound for the and-or matching problem.

Proposition 1. *The problem whether an and-or template graph matches a system graph is PSPACE-hard.*

Proof. We give a reduction from TQBF, i.e. the question whether a quantified Boolean formula is true or not. Suppose that $Q_0 x_0 \cdots Q_l x_l \; F(x_0, \dots, x_l)$ is in prenex normal form, with $Q_i \in \{\exists, \forall\}$ and F a quantifier-free formula in 3-CNF. That is, $F = C_1 \wedge \cdots \wedge C_m$, with C_j denoting disjunctions of at most three literals. For simplifying notations, the graphs M, N described below are such that states are labelled by charts (instead of events). Recall that this can be easily translated to a labelling by events.

Let the set of processes be $\mathcal{P} = \{P_i, P_i', R_i, R_i' \mid 0 \le i \le l\}$. Let also p_i (resp. p_i') denote a message from P_i to P_i' (resp. from P_i' to P_i). Analogously, let n_i resp. n_i' denote a message from R_i to R_i', resp. from R_i' to R_i. The system graph N (see also Fig. 2) is the single MSC $p_1' p_1 n_1' n_1 \cdots p_l' p_l n_l' n_l$. Note that the events e with $L(e) \in \{P_i, P_i'\}$ are totally ordered in the causal order. Similarly, all events e with $L(e) \in \{R_i, R_i'\}$ are totally ordered.

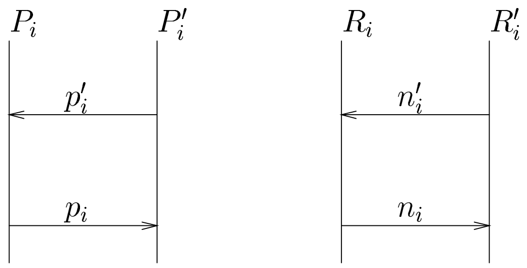

Fig. 2. The system graph.

The template graph $M = (\mathcal{S}, \tau, s_0, c, \mathcal{S}_\vee, \mathcal{S}_\wedge)$ is given by the vertex set

$$\mathcal{S} = \{s_i, s_i^+, s_i^- \mid 0 \le i \le l\} \cup \{t_i, t_{i,1}, t_{i,2}, t_{i,3} \mid 1 \le i \le m\} \cup \{t\},$$

and the transition relation $\tau \subseteq \mathcal{S} \times \mathcal{S}$:

$$\tau = \{(s_i, s_i^+), (s_i, s_i^-), (s_j^+, s_{j+1}), (s_j^-, s_{j+1}) \mid 0 \le i,j \le l, j \ne l\} \cup \\ \{(s_l^+, t), (s_l^-, t)\} \;\cup\; \{(t, t_i) \mid 1 \le i \le m\} \cup \{(t_i, t_{i,1}), (t_i, t_{i,2}), (t_i, t_{i,3}) \mid 1 \le i \le m\}$$

For $s \in \mathcal{S}$ we define $s \in \mathcal{S}_\wedge$ if and only if either $s = s_i$ and $Q_i = \forall$, or $s = t$. That is, the nodes s_i, $0 \le i < l$, are of type corresponding to the quantifier Q_i. Moreover, the node t is an and-node corresponding to the conjunction of the clauses. All remaining nodes are or-nodes. The states s_i^+, s_i^- are labelled by single messages (see also Fig. 3):

$$c(s) = \begin{cases} p_i & \text{for } s = s_i^+, 0 \le i \le l \\ n_i & \text{for } s = s_i^-, 0 \le i \le l \end{cases}$$

Let $C_j = x_{j,1} \vee x_{j,2} \vee x_{j,3}$ be a clause. If $x_{j,1} = x_k$ is a positive literal then let $c(t_{j,1}) = n'_k$. For a negative literal $x_{j,1} = \bar{x}_k$ we let $c(t_{j,1}) = p'_k$. The definition of $c(t_{j,2}), c(t_{j,3})$ is similar, depending on $x_{j,2}$ resp. $x_{j,3}$. For example, in Fig. 3 we represented the clause $x_1 \vee \bar{x}_2 \vee x_3$. All remaining states are labelled by $\emptyset$.

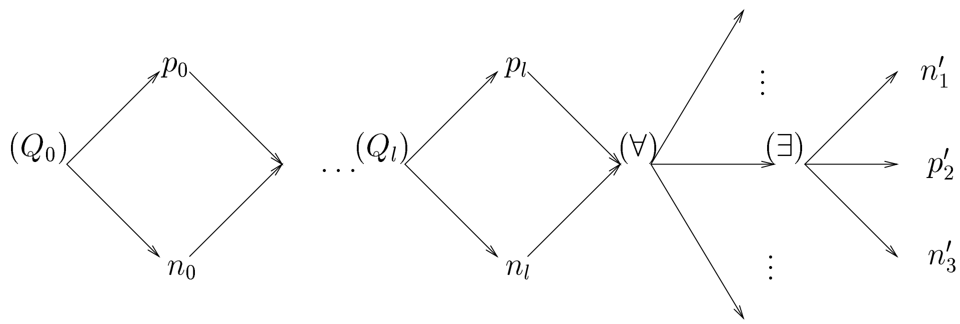

Fig. 3. The template graph.

There is a natural bijection between the assignments $\sigma : \{x_1, \dots, x_n\} \rightarrow \{\mathbf{t}, \mathbf{f}\}$ for F and the paths from s_0 to t in M. Formally, if s_i^+ belongs to a path then x_i is assigned the value $\mathbf{t}$, whereas if s_i^- belongs to a path, then x_i is assigned the value $\mathbf{f}$. It is not difficult to verify that a run R of M matches the system chart N if and only if every path in R from s_0 to t defines an accepting assignment for F. Hereby, we choose for every clause a leaf corresponding to a true literal. The (easy) proof is left to the reader.

3.2 Matching and-or templates in PSPACE

Showing the lower bound for the and-or matching problem was quite straightforward, which is not the case with the upper bound. Actually, it is a priori not clear why it would be possible to restrict the and-or matching problem to a finite instance of matching. The aim of the next propositions is to give a decomposition of maximal runs in an and-or template graph in two parts. The first part is a polynomially bounded tree, all paths of which have to be matched against a finite path in the system. The remaining part is the so-called recurrent part, for which only (abstract) events have to be recorded, all of which have to occur in a loop of the system graph.

Let $x = x_1, \dots, x_k = y$ be a path in a run $R = (X, \rightarrow, x_0, \ell, \mathrm{ev})$. We write $x \prec y$ whenever the event $\mathrm{ev}(x)$ precedes causally the event $\mathrm{ev}(y)$ in the sequence $\mathrm{ev}(x_1) \cdots \mathrm{ev}(x_k)$. Let $y{\downarrow} = \{x \in X \mid x \prec y\}$.

Definition 6. *Let $R = (X, \rightarrow, x_0, \ell, ev)$ be a run in an and-or MSC graph M. Let $X_e = \{x \in X \mid ev(x) = e\}$. We define a set of events $E_{fin}(R) \subseteq E$ as*

$$E_{fin}(R) = \{e \in E \mid \max\{|x{\downarrow}| \mid x \in X_e\} < \infty\}.$$

By $X_{fin}(R) \subseteq X$ we denote the subset of vertices $X_{fin}(R) = \{x \downarrow \mid ev(x) \in E_{fin}(R)\}$.

Intuitively, $E_{\text{fin}}(R) \subseteq E$ contains all events which require an exact matching for the corresponding nodes x (resp. paths from $x\downarrow$), i.e. for nodes x with $\text{ev}(x) \in E_{\text{fin}}(R)$. Equivalently, for nodes x with events from $E \setminus E_{\text{fin}}(R)$ the only information needed is $\text{ev}(x) \in E$. Consider again the and-or template of Fig. 1. We have $E_{\text{fin}}(R) \subseteq \{\mathsf{s}_{32}, \mathsf{r}_{32}\}$ for every maximal run R of M. Put it another way, $s_{21} \notin E_{\text{fin}}(R)$, since N_1 has to repeat infinitely often on some path of the run. Moreover, $\mathsf{s}_{12} \notin E_{\text{fin}}(R)$ since every maximal run includes paths $N_1^k N_2$ for every k and $r_1 \prec s_2$. The situation differs for node N_3. We might have a maximal run where N_3 occurs on every path just once, before N_2. Since there is no dependence from N_1 to N_3 all events e with $\text{ev}(e) = \mathsf{s}_{32}$ can be mapped to the same event on the system path (similarly for $\text{ev}(e) = \mathsf{r}_{32}$). Thus, node N_2 represents the bounded part, which has to be matched exactly against the system.

Notations: Let $M = (\mathcal{S}, \tau, s_0, c, \mathcal{S}_\wedge, \mathcal{S}_\vee)$ be an and-or graph and consider a run $R = (X, \rightarrow, x_0, \ell, \text{ev})$. Let $\rho = (x_1, x_2, \dots)$ be a sequence of vertices from X, then $\ell(\rho)$ denotes the sequence of states $(\ell(x_1), \ell(x_2), \dots) \in \mathcal{S}^\infty$. For $x \in X$ we denote by R_x the subtree of R with root x and by $\pi(x)$ the path $(x_0, \dots, x_k)$ from x_0 to $x = x_k$. By $\pi_{\text{fin}}(x)$ we denote the subsequence $(x_{i_1}, \dots, x_{i_l})$ of $\pi(x)$ containing exactly the vertices from $X_{\text{fin}}(R)$. (I.e. we have $x_j \in X_{\text{fin}}(R)$ if and only if $j = i_m$ for some m.) More generally, for a (finite or infinite) path ρ in R we denote by $\pi_{\text{fin}}(\rho)$ the subsequence of vertices of ρ belonging to $X_{\text{fin}}(R)$.

Remark 1. Note that for any path ρ in R the subsequence $\pi_{\text{fin}}(\rho)$ is finite. Moreover, there exists a sequence of events $\alpha \in (E \setminus E_{\text{fin}}(R))^\infty$ such that $\text{ev}(\rho)$ and $\text{ev}(\pi_{\text{fin}}(\rho))\alpha$ define the same causal structure, i.e. $\text{tr}(\text{ev}(\rho)) = \text{tr}(\text{ev}(\pi_{\text{fin}}(\rho))\alpha)$.

Lemma 1. *Let M be an and-or template graph and consider a run R' of M. Let $e \in E_{fin}(R')$. Then there exists a run $R = (X, \rightarrow, x_0, \ell, ev)$ of M satisfying the following conditions:*

1. *Every path of R is a subpath of some path in R'.*
2. *Let $x, y, z \in X$ be such that $x \in X_e$ and $y, z \in x{\downarrow}$. Then $\ell(y) = \ell(z)$ implies $y = z$.*
3. *We have $E_{fin}(R') \subseteq E_{fin}(R)$. Moreover, if R' is maximal, then R is maximal, too.*

The previous lemma says that it suffices to consider maximal runs R where for every path ρ the subsequence $\pi_{\text{fin}}(\rho)$ has length at most $|\mathcal{S}|$, with $\mathcal{S}$ denoting the set of states of M.

Lemma 2. *Let M be an MSC and let χ be a loop in an MSC graph such that $msg(M) \subseteq msg(\chi)$. Then M matches χ^ω.*

Lemma 3. *Let R be a run in an and-or template graph M and let N be a system graph. Then R matches N if and only if R matches some path in N of the form $\chi_0\chi_1^\omega$, where χ_0 is a finite path and χ_1 is a (possibly empty) loop.*

Lemma 4. *Let $R = (X, \rightarrow, x_0, \ell, ev)$ be a run in an and-or template graph M and let $\chi = \chi_0\chi_1^\omega$ be a path in a system graph N such that R matches χ. Then we have*

1. *$(E \setminus E_{fin}(R)) \subseteq msg(\chi_1)$*
2. *Every path ρ in R is such that $\pi_{fin}(\rho)$ matches $\chi_0\chi_1^{|\mathcal{S}|}$.*

The proposition below says that it is sufficient to consider maximal runs R such that any two vertices $x, y \in X_{\mathrm{fin}}(R)$ which are such that the subpaths $\pi_{\mathrm{fin}}(x)$ and $\pi_{\mathrm{fin}}(y)$ have the same sequence of state labels also have equal subtrees. We assume in the following w.l.o.g. that the source state of the template has no incoming edges, thus the root x_0 of a run R belongs to $X_{\mathrm{fin}}(R)$.

Proposition 2. *Let M be an and-or template graph and consider a run R of M. Then a run $R' = (X, \rightarrow, x_0, \ell, ev)$ of M exists such that:*

1. *For all vertices $x, y \in X_{fin}(R')$ with $\ell(\pi_{fin}(x)) = \ell(\pi_{fin}(y))$ we have $R'_x = R'_y$.*
2. *For every path ρ' in R', the causal structure $tr(\pi_{fin}(\rho'))$ is equal to $tr(\pi_{fin}(\rho))$, for some path ρ in R.*
3. *If R satisfies Lemma 1, then R' satisfies Lemma 1, too. Moreover, if R is maximal, then R' is also maximal and we have finally $E_{fin}(R) \subseteq E_{fin}(R')$.*

Proof. Consider a maximal run R satisfying Lem. 1. We assume that R does not satisfy the first condition of the statement and we consider two vertices x, y from $X_{\mathrm{fin}}(R)$ with $\ell(\pi_{\mathrm{fin}}(x)) = \ell(\pi_{\mathrm{fin}}(y))$, but $R_x \neq R_y$. Clearly, x, y are incomparable w.r.t. the successor relation in R, since they are labelled by the same state. We claim that the run R' obtained from R by replacing R_y by a copy of R_x satisfies Conds. (2) and (3) of the statement. To see this, note that for every maximal path ρ' in R' containing y there is a corresponding maximal path ρ in R containing x such that $\ell(\pi_{\mathrm{fin}}(\rho)) = \ell(\pi_{\mathrm{fin}}(\rho'))$ (here π_{fin} is meant w.r.t. R.) Moreover, the nodes in $\rho \setminus \pi_{\mathrm{fin}}(\rho)$ (resp. in $\rho' \setminus \pi_{\mathrm{fin}}(\rho')$) are labelled by events from $E \setminus E_{\mathrm{fin}}(R)$.

For subtrees R' of R we define $m(R')$ as

$$m(R') = \max\{|\pi_{\mathrm{fin}}(\rho)| \mid \rho \text{ is a path in } R'\}\,.$$

Note that $m(R')$ is defined w.r.t. R, i.e. $\pi_{\mathrm{fin}}(\rho)$ is given by R. We show below by induction on $m(R')$ how to obtain a run $\hat{R}$ satisfying the requirement of the proposition with regard to $X_{\mathrm{fin}}(R)$. This means that any two vertices x, y of $\hat{R}$ with $\ell(\pi_{\mathrm{fin}}(x)) = \ell(\pi_{\mathrm{fin}}(y))$ will have equal subtrees. However, we will obtain $E_{\mathrm{fin}}(R) \subseteq E_{\mathrm{fin}}(\hat{R})$. If $E_{\mathrm{fin}}(R) = E_{\mathrm{fin}}(\hat{R})$ then we are done, since in this case

$\pi_{\text{fin}}(\rho)$ as given by R is the same as $\pi_{\text{fin}}(\rho)$ as given by $\hat{R}$. Otherwise we repeat the construction with $\hat{R}$. After at most $|E|$ steps we obtain the desired run.

For defining $\hat{R}$ we consider all vertices $x \in X_{\text{fin}}(R)$ at depth 1, i.e. $\pi_{\text{fin}}(x) = (x_0, x)$ has length 1. Suppose that x is labelled by $s \in \mathcal{S}$ and let us choose some fixed vertex x_s of this kind, for each possible s. Consider the subtree R_{x_s} rooted at x_s. Clearly, $m(R_{x_s}) < m(R)$. By induction we may assume that any two vertices y, z in R_{x_s} with $\ell(\pi_{\text{fin}}(y)) = \ell(\pi_{\text{fin}}(z))$ have equal subtrees. We can replace all subtrees R_x for x with $\pi_{\text{fin}}(x) = (x_0, x)$, $\ell(x) = s$, by R_{x_s} and obtain the desired run $\hat{R}$.

The next proposition gives the characterization for deciding the existence of a matching from an and-or graph into an MSC graph.

Proposition 3. *Let M be an and-or template graph and let N be a system graph. Let $\mathcal{S}$ denote the set of states of M. Then M matches N if and only if there exist*

1. *A set of events $G \subseteq E$ of M.*
2. *A path χ in N to a strongly connected component C of N with $G \subseteq msg(C)$.*
3. *A tree $T_0 = (X, \rightarrow, x_0, \ell, ev)$ labelled by $\ell : X \rightarrow \mathcal{S}$, $ev : X \rightarrow E$, such that for any two distinct nodes $x, y \in X$ which are either siblings or comparable in T_0, $\ell(x) \neq \ell(y)$. Moreover, every path of T_0 matches the path χ in N.*
4. *A maximal run R of M such that for every maximal path ρ of R we have $tr(ev(\rho)) = tr(ev(\xi)\alpha)$, where ξ is a maximal path in T_0 and $\alpha \in G^\infty$.*

Proof. By Lem. 3, 4 and Prop. 2 we can assume that we have a maximal run $R = (X, \rightarrow, x_0, \ell, \text{ev})$ of M which satisfies the first condition of Prop. 2 and matches a path $\chi\chi_1^\omega$ in N. Let C denote the strongly connected component represented by χ_1. Let $G = E \setminus E_{\text{fin}}(R)$, then $G \subseteq \text{msg}(\chi_1)$ by Lem. 4. By assumption, the root x_0 of R belongs to $X_{\text{fin}}(R)$. Define first a tree $T_0' = (X_{\text{fin}}(R), \rightarrow, \ell, x_0)$ by letting $x \rightarrow y$ in T_0' whenever there is a path $x \rightarrow x_1 \rightarrow \cdots \rightarrow x_k \rightarrow y$ in R with $x_i \notin X_{\text{fin}}(R)$ for every i. The tree T_0 is now defined by identifying two vertices x, y in T_0' whenever $\ell(\pi(x)) = \ell(\pi(y))$. By Cond. (1) of Prop. 2 this step is well-defined. By Rem. 1 every maximal path ρ in R has the same causal structure as $\text{ev}(\pi_{\text{fin}}(\rho))\alpha$, for some $\alpha \in G^\infty$. Moreover, $\pi_{\text{fin}}(\rho)$ is a maximal path in T_0.

Remark 2. The length of the path χ in the previous proposition can be bounded by a polynomial in $|\mathcal{S}|$ and $|\mathcal{S}'|$, where $\mathcal{S}'$ is the set of states of N.

The next lemma implies that we can check the last condition of Prop. 3 in PSPACE.

Lemma 5. *Let $M = (\mathcal{S}, \tau, s_0, c, \mathcal{S}_\wedge, \mathcal{S}_\vee, \mathcal{P})$ be an and-or template graph. Suppose that we are given a state $r \in \mathcal{S}$, a set $L \subseteq \mathcal{S}$ of sink states in M and a set of events $I(s) \subseteq E$ for each $s \in L$, as well as a set $G \subseteq E$. Then we can check in PSPACE whether a maximal run $R = (X, \rightarrow, x_0, \ell, ev)$ of M exists satisfying the following conditions:*

1. *State r labels the root x_0 of R.*
2. *For every state $s \in L$ there is at least one leaf labelled by s. Moreover, every non-leaf node x is labelled by $ev(x) \in G$.*
3. *For every path $y \xrightarrow{+} x$ in R to a leaf x with $\ell(x) = s$ we have either $ev(y) \in I(s)$ or $y = x_0$.*

Theorem 1. *There is a PSPACE algorithm for deciding whether an and-or template graph M matches a system graph N.*

Proof. We have to check in PSPACE the existence of T_0, R, G, χ, C as in Prop. 3. Clearly, $G \subseteq E$ and C can be guessed and stored. By Rem. 2 the path χ can also be stored. The problem arises with T_0, since the size of T_0 might be exponential (however, the depth and the degree of T_0 are linear in $|\mathcal{S}|$, hence we can store paths of T_0.) The main idea is to guess the tree T_0 implicitly, in a DFS traversal where we store together with the current path also the (ordered sequence of) siblings of the intermediate nodes. Using Lem. 5 it is sufficient to verify the existence of a suitable run R piecewise, along with the DFS traversal.

The (nondeterministic) PSPACE algorithm works as follows (see also Fig. 4). Assume that the current path in T_0 is $s_0, \ldots, s_k$, $k < |\mathcal{S}|$, and that we also stored the (ordered) sequence L_i of siblings of s_i in T_0, $1 \leq i < k$. Moreover, we guessed sets of events $I(s) \subseteq E$ for all $s \in L_{i+1}$ and we verified the existence of maximal runs satisfying Lem. 5 with $r = s_i$, $L = L_{i+1}$, G. If $k < |\mathcal{S}|$ then we can either proceed with DFS and guess $s_{k+1}, L_{k+1}, \ldots$, or s_k is a leaf in T_0 and we apply Lem. 5 with $L = \emptyset$. Furthermore we verify that $(s_0, \ldots, s_k)$ is downward closed w.r.t. the causal order in the run built so far by using Lem. 5. For this, we use the sets $I(s_i)$. Using Lem. 5 we checked that a suitable run with root s_{i-1} exists s.t. every node on the path from s_{i-1} to s_i is labelled by an event from $I(s_i)$. Thus, it suffices to check for each $i < j \leq k$ and each $e' \in I(s_i)$ that $e' \lessdot \mathrm{ev}(s_j)$ is not satisfied. Finally, we verify that the path $(s_0, \ldots, s_k)$ matches χ and then we backtrack in the DFS traversal.

3.3 Specifying Properties by And-Or Templates

Consider a template given as an and-or MSC graph M. We want to determine the set of event sequences $L(M) \subseteq E^\infty$ which are described by M. That is, we let $\alpha \in L(M)$ if $\alpha = \mathrm{ev}(\chi)$ for some path χ in a system graph such that there is a matching of some maximal run of M into the path χ. Prop. 3 gives us the basic description of the set $L(M)$ as a finite union over languages $L(M, T_0, G) \subseteq E^\infty$, where T_0 is a tree labelled by states of M and $G \subseteq E$ is a set of events of M. Let $\alpha \in L(M, T_0, G)$ if

- every event $e \in G$ occurs infinitely often in α, and
- every path of T_0 matches α.

For any language $F \subseteq E^*$ we denote by $F \sqcup\!\sqcup E^*$ the shuffle ideal generated by F, i.e. $F \sqcup\!\sqcup E^* = \{v_0 u_1 v_1 \cdots u_n v_n \mid u_1 \cdots u_n \in F,\, v_i \in E^*\}$. For subsets $G \subseteq E$

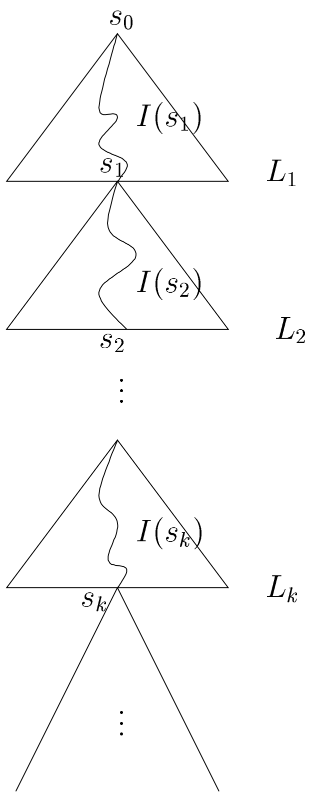

Fig. 4. Guessing the maximal run R, resp. the path $(s_0, \ldots, s_k)$ in T_0.

we denote by $\inf(G)$ the set of sequences $\alpha \in E^\infty$ where every event $e \in G$ occurs infinitely often. For $\alpha \in E^\infty$ we denote below by $\mathrm{tr}(\alpha) \subseteq E^\infty$ the set of linearizations of the causal order of α. Let $\mathrm{tr}(F) = \cup_{\alpha \in F} \mathrm{tr}(\alpha)$.

Proposition 4. *Let M be an and-or MSC template graph over the event set E. The language $L(M)$ associated with the template M has the form*

$$\bigcup_{F,G} (F \sqcup\!\sqcup E^*)\, \mathit{inf}(G)\,,$$

where $F \subseteq E^*$ is a finite set, $tr(F) = F$ and $G \subseteq E$.

Proof. Any language $L(M, T_0, G)$ as described above is a (finite) intersection of languages of the form $(\mathrm{tr}(\alpha) \sqcup\!\sqcup E^*)\inf(G)$, where α is such that $\mathrm{tr}(\alpha) \subseteq \mathrm{tr}(\beta)$ for some path χ in T_0 with $\beta = \mathrm{ev}(\chi)$. Let $\alpha, \beta \in E^*$, $G, H \subseteq E$. Then we have

$$(\mathrm{tr}(\alpha) \sqcup\!\sqcup E^*)\inf(G) \cap (\mathrm{tr}(\beta) \sqcup\!\sqcup E^*)\inf(H) = (\Gamma \sqcup\!\sqcup E^*)\inf(G \cup H)\,,$$

where Γ contains all sequences γ of minimal length such that for some $\alpha' \in \mathrm{tr}(\alpha)$, $\beta' \in \mathrm{tr}(\beta)$, both α', β' are subwords of γ. Moreover, it is not hard to check that Γ can be chosen s.t. $\mathrm{tr}(\Gamma) = \Gamma$.

4 Matching LTL Properties

In this section we consider properties (templates) specified by linear temporal logic. Our LTL formulas define finite or infinite strings over the alphabet $E = \{\mathsf{s}_{ij}, \mathsf{r}_{ij} \mid 1 \leq i \neq j \leq m\}$ of events. They are built over atomic propositions P_e, $e \in E$, using the temporal operators X (nexttime) and U (until), and the Boolean connectives $\wedge, \vee, \neg$.

Definition 7 (LTL Matching Problem). *Given a template M as an LTL formula ϕ and a system N as an MSC graph. We say that the template M matches the system N if there is some sequence of events $\alpha \in E^\omega$ such that $\alpha \models \phi$ and α matches some maximal path in N.*

Theorem 2. *The LTL matching problem is PSPACE-complete.*

We will show the above theorem in a more general setting using automata. Recall that every state of the system graph N is labelled by a single event. Thus we can easily associate to N a nondeterministic Büchi automaton $\mathcal{A}_N$ of the same size such that $L(\mathcal{A}_N)$ consists of all maximal executions of N. On the other hand it is well-known how to associate to every LTL formula ϕ an alternating Büchi automaton with $O(|\phi|)$ states accepting exactly the sequences satisfying ϕ, see e.g. [10]. Moreover, for every alternating Büchi automaton with n states there is an equivalent nondeterministic Büchi automaton with $2^{O(n)}$ states which can be built off-line using only $O(n)$ space, see [6]. We are thus led to a language-theoretical formulation of the LTL matching problem:

The matching problem for alternating automata: Given an alternating Büchi automaton $\mathcal{A}_M$ and a nondeterministic Büchi automaton $\mathcal{A}_N$ over E. Then we ask whether some sequences $\alpha \in L(\mathcal{A}_M)$, $\beta \in L(\mathcal{A}_N)$ exist such that α matches β.

Proposition 5. *The matching problem for alternating automata is PSPACE-complete.*

Proof. Due to the PSPACE-hardness of the satisfiability problem for LTL it suffices to show the upper bound. Let $\mathcal{A}_M$ denote an alternating Büchi automaton and let $\mathcal{B}_M = (Q, E, q_0, \delta, F)$ denote an equivalent nondeterministic Büchi automaton as given by [6]. Let $\mathcal{A}_N = (Q', E, q_0', \delta', F')$ denote the nondeterministic Büchi automaton representing the system. By an immediate modification and extension of Lem. 3 we note that M matches N if and only if final states $f \in F$, $f' \in F'$ exist such that

- for some paths $\pi = (q_0, \ldots, q_k = f)$ in $\mathcal{B}_M$, resp. $\pi' = (q_0', \ldots, q_l' = f')$ in $\mathcal{A}_N$, the execution of π matches the execution of π', i.e. $\mathrm{ev}(\pi)$ matches $\mathrm{ev}(\pi')$, and

- some loops ρ around f, resp. ρ' around f' exist satisfying $\text{msg}(\rho) \subseteq \text{msg}(\rho')$.

Clearly, we can choose π of length $k \leq |Q|$. But since $|Q| \in 2^{O(n)}$ we cannot guess π directly. On the other hand, we have to match π against π' and this means that we have to consider permutations, due to the causal order of the events. To overcome these problems we consider below the shape of π and π' in more details and exploit the fact that matching allows gaps.

We denote by $C_1, C_2, \ldots, C_m$ the maximal strongly connected components of the subgraph of $\mathcal{A}_N$ induced by π'. Thus, $\pi' = \pi'_1 \cdots \pi'_m$, such that w.l.o.g. π'_i induces C_i. The components C_i are naturally ordered, and we write $C_i < C_j$ for $i < j$.

Consider some mapping μ matching π into π'. We decompose π into maximal segments π_i, $\pi = \pi_1 \cdots \pi_r$, such that for every segment π_i all events of $\text{ev}(\pi_i)$ are matched by μ into the same component π'_j, for some j. Let $C(i) \in \{1, \ldots, m\}$ denote the component C_j with $\mu(e) \in \pi'_j$ for all events e in $\text{ev}(\pi_i)$. Note that for every $1 \leq i < j \leq r$, $C(i) > C(j)$ implies that $e \not\prec e'$ for all events e, e' with e occurring in $\text{ev}(\pi_i)$, resp. e' occurring in $\text{ev}(\pi_j)$. Otherwise, by the definition of matchings, we would require that $\mu(e) \prec \mu(e')$, contradicting $C(i) > C(j)$. The second basic observation is that every segment π_i matches *any* path ρ_i in $C(i)$ which visits every state of $C(i)$ sufficiently often. Therefore, we claim that a path π in $\mathcal{B}_M$ matches some path π' in $\mathcal{A}_N$ if and only if

- some maximal strongly connected components $C_1, \ldots, C_m$ exist in $\mathcal{A}_N$ with $q'_0 \in C_1$, as well as transitions from C_i to C_{i+1} for all i,
- π can be decomposed as $\pi = \pi_1 \cdots \pi_r$ such that for every i some component $C(i) \in \{C_1, \ldots, C_m\}$ exists such that $\text{msg}(\pi_i) \subseteq \text{msg}(C(i))$,
- for every $1 \leq i < j \leq r$ with $C(i) > C(j)$ we have $e \not\prec e'$ for all events e, e' occurring in $\text{ev}(\pi_i)$, resp. in $\text{ev}(\pi_j)$.

We already showed one direction of the claim. For the converse, assume that $C_i, \pi_j, C(k)$ exist as above. Let ρ_k denote the subpath of π consisting of all segments π_i with $C(i) = C_k$. Since $\text{msg}(\rho_k) \subseteq \text{msg}(C_k)$ and C_k is strongly connected, we can determine a path ρ'_k looping in C_k such that ρ_k matches ρ'_k (via some mapping μ). Moreover, ρ'_k can be chosen such that $\rho' = \rho'_1 \cdots \rho'_m$ is a path in $\mathcal{A}_N$. Finally the last condition above yields that π matches ρ'. This is seen by noting that for any events e, e' occurring in $\text{ev}(\rho_i)$, resp. $\text{ev}(\rho_j)$, $e \prec e'$ implies $i < j$, hence also $\mu(e) \prec \mu(e')$.

The PSPACE algorithm on input $\mathcal{A}_M, \mathcal{A}_N$ works as follows. First we guess $m \leq |Q'|$ strongly connected components $C_1, \ldots, C_m$ of $\mathcal{A}_N$, such that $q'_0 \in C_1$ and there exist edges from C_i to C_{i+1}, for all i. We guess the path π in $\mathcal{B}_M$ and its decomposition $\pi = \pi_1 \cdots \pi_r$ on-line. The only information which is (temporarily) stored concerns event types, which are required in order to check that $e \not\prec e'$ for all events e, e' occurring in $\text{ev}(\rho_i)$, resp. $\text{ev}(\rho_j)$, satisfying $i < j$ and $C(i) > C(j)$. For this step, we have to record at most m subsets of E. More precisely, assume that $E_1, \ldots, E_m \subseteq E$ are initially empty. Along with guessing π_i and some $C(i) = C_k$ we store $\text{msg}(\pi_i) \subseteq E$ and check that for no events $e \in E_j$, $e' \in \text{msg}(\pi_i)$ we have $e \prec e'$, whenever $C_k < C_j$ (i.e. $k < j$). Then we add $\text{msg}(\pi_i)$ to E_k. Moreover, we verify that $\text{msg}(\pi_i) \subseteq \text{msg}(C(i))$.

5 Conclusion

In this paper we characterized the complexity of matching template MSCs which are given as and-or graphs, resp. by LTL formulas. Under these semantics the matching problem becomes PSPACE-complete. However, our proofs show that the increased complexity (compared with previously investigated template semantics, e.g. or-graphs) is due rather to a more succinct representation than to more expressiveness. This leads to the question whether using negations resp. requiring that certain events occur only finitely often might increase the expressiveness.

Acknowledgments. The referees are kindly acknowledged for the careful reading and the suggestions for improving the presentation of the paper.

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 the Tools 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. H. Ben-Abdallah and S. Leue. Mesa: Support for scenario-based design of concurrent systems. In B. Steffen, editor, *Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems, 4th International Conference, TACAS'98*, number 1384 in Lecture Notes in Computer Science, pages 118–135, Lisbon, Portugal, 1998. Springer.
4. V. Diekert and G. Rozenberg, editors. *The Book of Traces.* World Scientific, Singapore, 1995.
5. A. Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.
6. S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.
7. A. Muscholl, D. Peled, and Z. Su. Deciding properties of message sequence charts. In M. Nivat, editor, *Proceedings of the 1st International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'98), Lisbon, Portugal, 1998*, number 1378 in Lecture Notes in Computer Science, pages 226–242, Berlin-Heidelberg-New York, 1998. Springer.
8. M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures and domains, part 1. *Theoretical Computer Science*, 13:85–108, 1981.
9. V. R. Pratt. Modelling concurrency with partial orders. *International Journal of Parallel Programming*, 15(1):33–71, 1986.
10. M. Y. Vardi. An automata-theoretic approach to linear-temporal logic. In F. Moller and G. Birtwistle, editors, *Logics for concurrency*, number 1043 in Lecture Notes in Computer Science, pages 238–266, Berlin-Heidelberg-New York, 1996. Springer.