

Anchored Concatenation of MSCs

Madhavan Mukund¹, K. Narayan Kumar¹, P.S. Thiagarajan², Shaofa Yang²

¹Chennai Mathematical Institute, Chennai, India
Email: madhavan,kumar@cmi.ac.in

²School of Computing, National University of Singapore, Singapore
Email: thiagu,yangsf@comp.nus.edu.sg

Abstract: We study collections of Message Sequence Charts (MSCs) defined by High-level MSCs (HMSCs) under a new type of concatenation operation called *anchored concatenation*. We show that there is no decision procedure for determining if the MSC language defined by an HMSC is regular and that it is undecidable if an HMSC admits an implied scenario. Further, the languages defined by locally synchronized HMSCs are precisely the finitely generated regular MSC languages. These results mirror the ones for the asynchronous concatenation case. On the other hand, the MSC language obtained by closing under implied scenarios is regular for *every* HMSC. Secondly, one can effectively determine whether a locally synchronized HMSC admits an implied scenario. Neither of these results hold in the asynchronous concatenation case.

1 Introduction

Message Sequence Charts (MSCs) are an appealing visual formalism that is suitable for modelling telecommunication software [12]. They are used in a number of software engineering notational frameworks such as SDL [18] and UML [5, 7]. A collection of MSCs is used to capture the scenarios that a designer might want the system to exhibit (or avoid). Hence it is fruitful to have suitable mechanisms to specify a collection of MSCs.

A common way to specify a collection of MSCs is to use a High-level (or Hierarchical) Message Sequence Chart (HMSC) [14]. An HMSC is a directed graph where each node is labelled an HMSC or an MSC. The HMSCs labelling the nodes are not allowed to reference each other. Hence, without loss of expressiveness, we shall conveniently assume that each node is labelled by just an MSC. From an HMSC one obtains MSCs by walking from an initial vertex to a terminal one, while *concatenating* the MSCs at the vertices visited. The collection of MSCs thus obtained is defined to be the MSC language of the HMSC.

In the literature, one encounters two extreme types of MSC concatenation, *asynchronous* and *synchronous concatenation*. In *asynchronous concatenation* the MSCs are concatenated along lifelines. If $M = M_1 \circ M_2$, then no event of an instance in M_2 may execute until all the events of the same instance in M_1 have finished executing. In *synchronous concatenation* one demands that *all the events* of M_1 must be executed before *any* event in M_2 can be executed. Asynchronous concatenation leads to a very expressive class of HMSC-definable MSC collections while synchronous concatenation gives rise to very restricted and impractical MSC collections.

We propose here a new and natural MSC concatenation termed *anchored concatenation*. In this operation, we demand that an agent which is active in both M_1 and M_2 can start executing in M_2 only after *all* the events in M_1 have finished executing; in effect, all—and

only—the agents participating in M_1 must synchronize before any agent of M_2 that was also active in M_1 can start executing again. This is a weaker form of synchronous concatenation since we impose no restrictions on the agents of M_2 that do not participate in M_1 .

We present here the resulting theory of MSC languages generated by HMSCs. We pay particular attention to their closures with respect to *implied scenarios* [1, 2, 20]. Briefly, implied scenarios arise naturally when one implements a collection of MSCs in a distributed setting. One of our main results is that the closure (with respect to implied scenarios) of *every* HMSC is a regular MSC language. This establishes that HMSCs can be a fruitful specification formalism if we interpret the set of scenarios defined by an HMSC to be its implied scenarios-closure under anchored concatenation. Such collections can be easily realized as a network of finite state automata with local acceptance conditions; they will communicate with each other via bounded fifoes as well as by performing common synchronization actions.

In common with the theory under asynchronous concatenation, there is no decision procedure for determining if the MSC language defined by an HMSC is regular or for determining if an HMSC admits an implied scenario. It turns out that the languages defined by HMSCs that satisfy the syntactic condition of being locally synchronized are precisely the finitely generated regular languages.

On the other hand, the language of MSCs obtained by closing under implied scenarios is both regular and finitely generated for *every* HMSC. Moreover, one can decide whether a locally synchronized HMSC admits an implied scenario. None of these results holds in the case of asynchronous concatenation.

There is a substantial theory of the MSC languages defined by HMSCs under asynchronous concatenation [1, 2, 3, 9, 10, 11, 13, 16]. Synchronous concatenation of MSCs is informally defined and some related verification problems and their complexities are discussed in [3]. In the framework of Live Sequence Charts [8], a restricted type of concatenation that is much closer to anchored and not synchronous concatenation is implicitly assumed.

In the next section we extend the usual notion of MSCs in order to admit synchronizations. This gives a convenient handle on the anchored concatenation operation. We then use a restricted type of these enriched MSCs to define the MSC languages generated by HMSCs under anchored concatenation. In Section 3, we present the related automaton model called *product* Message Passing Automata. In the subsequent two sections we establish our main results. In the final section, we briefly discuss the prospects for future work.

2 Message Sequence Charts

Let $\mathcal{P} = \{p, q, r, \dots\}$ be a finite set of *agents* (processes). These agents communicate with each other via fifo channels as well as multi-way synchronizations. The set of *channels* is $Ch = \{(p, q) \in \mathcal{P} \times \mathcal{P} \mid p \neq q\}$. Let Δ be a finite alphabet of *messages*. We define the *communication alphabet* to be $\Sigma_{com} = \{p!q(m), p?q(m) \mid (p, q) \in Ch, m \in \Delta\}$ and the *synchronization alphabet* to be $\Sigma_{syn} = \{P \subseteq \mathcal{P} \mid |P| > 0\}$. We set $\Sigma = \Sigma_{com} \cup \Sigma_{syn}$. The action $p!q(m)$ denotes p sending a message m to q , while the action $p?q(m)$ denotes p receiving a message m from q . The action $P \in \Sigma_{syn}$ represents the processes in P performing a multi-way synchronization. We do not explicitly model the exchange of information that takes place during such a synchronization. A singleton synchronization $\{p\}$ represents an internal action performed by p . Henceforth, we fix $\mathcal{P}, \Delta, Ch, \Sigma$ and let p, q range over \mathcal{P} , m over Δ and P over Σ_{syn} .

For $a \in \Sigma$, we define $loc(a)$, the *locations* of a , as follows: $loc(p!q(m)) = loc(p?q(m)) = \{p\}$ and $loc(P) = P$. Thus $loc(a)$ is the set of processes that take part in a . For $p \in \mathcal{P}$, we define $\Sigma_p = \{a \in \Sigma \mid p \in loc(a)\}$.

Message sequence charts (MSCs) are restricted Σ -labelled posets. A Σ -labelled poset is a structure $M = (E, \leq, \lambda)$ where (E, \leq) is a poset and $\lambda : E \rightarrow \Sigma$ a labelling function. For $e \in E$, we define $\downarrow e = \{e' \in E \mid e' \leq e\}$. For $p \in \mathcal{P}$, we define $E_p = \{e \in E \mid p \in loc(\lambda(e))\}$. Also, for $a \in \Sigma$, we let $E_a = \{e \in E \mid \lambda(e) = a\}$. We set $E_{com} = \{e \in E \mid \lambda(e) \in \Sigma_{com}\}$ and $E_{syn} = \{e \in E \mid \lambda(e) \in \Sigma_{syn}\}$.

For $(p, q) \in Ch$ and $m \in \Delta$, we define the relation $<_{pqm} \subseteq E \times E$ to capture the causal relationship between the send and receive actions of each message. For $e, e' \in E$, $e <_{pqm} e'$ iff $\lambda(e) = p!q(m)$, $\lambda(e') = q?p(m)$ and $|\downarrow e \cap E_{p!q(m)}| = |\downarrow e' \cap E_{q?p(m)}|$.

For $(p, q) \in Ch$, define $<_{pq} = \bigcup_{m \in \Delta} <_{pqm}$. For $p \in \mathcal{P}$, define $\leq_{pp} = (E_p \times E_p) \cap \leq$, with $<_{pp}$ standing for the largest irreflexive subset of \leq_{pp} . An event e is classified as a *send*, *receive* or *synchronization event* in the obvious way.

We are now ready to define MSCs.

An *MSC* (over \mathcal{P}) is a *finite* Σ -labelled poset $M = (E, \leq, \lambda)$ which satisfies the following conditions:

- \leq_{pp} is a linear order for each p .
- For each $(p, q) \in Ch$ and $m \in \Delta$, $|E_{p!q(m)}| = |E_{q?p(m)}|$.
- \leq is the reflexive, transitive closure of $\bigcup_{p, q \in \mathcal{P}} <_{pq}$.
- If $e_1 <_{pqm} e_2$ and $f_1 <_{pqm'} f_2$ where $m \neq m'$, then $e_1 <_{pp} f_1$ iff $e_2 <_{qq} f_2$.
- If $e_1, e_2 \in E_{com}, e \in E_{syn}$ such that $e_1 <_{pq} e_2$, $p \neq q$, and $\{p, q\} \subseteq \lambda(e)$, then either $e < e_1 < e_2$ or $e_1 < e_2 < e$.

The fourth clause ensures that each channel (p, q) is fifo. The last clause ensures that no message from p to q crosses a synchronization involving p and q .

Figure 1 shows an MSC. We depict the events of the MSC in *visual order*. The communication actions of each process are arranged in a vertical line. Members of $<_{pq}$ are shown with horizontal or downward-sloping arrows from the vertical line of p to that of q . The labels on these arrows indicate the message transmitted. A synchronization action is drawn as a rectangle that disconnects the vertical lines of the processes participating in this synchronization. In what follows, unless stated otherwise, by an ‘‘MSC’’ we shall mean an object defined as above.

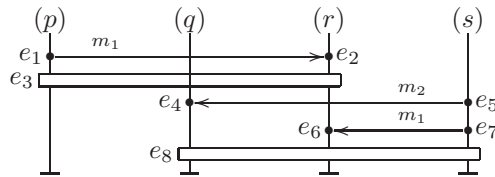


Figure 1: A simple MSC over $\{p, q, r, s\}$

We identify an MSC with its isomorphism class. Let $\mathcal{M}_{\mathcal{P}}$ be the set of MSCs over \mathcal{P} . The subscript \mathcal{P} will often be dropped. Let $M = (E, \leq, \lambda) \in \mathcal{M}$. Set $loc(e) = loc(\lambda(e))$ for $e \in E$

and $loc(M) = \bigcup\{loc(e) \mid e \in E\}$. The processes in $loc(M)$ are said to be *active* in M . We refer to the set of *linearizations* of M as $lin(M)$ —that is, $\sigma \in lin(M)$ iff $\sigma = \lambda(e_1) \dots \lambda(e_n)$, $E = \{e_1, \dots, e_n\}$ and for each pair e_i, e_j with $i < j$, $e_j \not\prec e_i$.

An *MSC language* (over \mathcal{P}) is a subset of \mathcal{M} . Let \mathcal{L} be an MSC language. Set $lin(\mathcal{L}) = \bigcup\{lin(M) \mid M \in \mathcal{L}\}$. We say \mathcal{L} is *regular* iff $lin(\mathcal{L})$ is a regular subset of Σ^* .

Concatenation of MSCs

Let $M_1 = (E_1, \leq_1, \lambda_1)$ and $M_2 = (E_2, \leq_2, \lambda_2)$ be MSCs. The *concatenation* $M_1 \circ M_2$ of M_1 and M_2 is the MSC $M = (E, \leq, \lambda)$ defined as follows:

- E is the disjoint union of E_1 and E_2 .
- \leq is the reflexive, transitive closure of $\leq_1 \cup \leq_2 \cup \sqsubset$, where $\sqsubset = \{(e_1, e_2) \mid e_1 \in E_1, e_2 \in E_2, loc(e_1) \cap loc(e_2) \neq \emptyset\}$.

We note that $M_1 \circ M_2$ is also an MSC. In what follows, by “concatenation” we shall always mean the operation \circ defined above.

Let $M = (E, \leq, \lambda)$ be an MSC. We say that M is *plain* iff $E_{syn} = \emptyset$. Plain MSCs correspond to the standard definition of MSCs in the literature [12]. For plain MSCs, the operation defined above corresponds to the usual definition of asynchronous concatenation.

Let $M_i = (E_i, \leq_i, \lambda_i)$, $i = 1, 2$, be a pair of plain MSCs. The *synchronous concatenation* of M_1 and M_2 is the MSC $M = (E, \leq, \lambda)$ where E is the disjoint union of E_1 and E_2 and \leq is the reflexive, transitive closure of $\leq_1 \cup \leq_2 \cup (E_1 \times E_2)$ (see, for instance, [3]). Thus synchronous concatenation for plain MSCs requires *all* events in M_1 be completed before *any* event in M_2 is executed. This can be captured in our setting by inserting a synchronous event involving the entire set of processes \mathcal{P} between M_1 and M_2 .

Synchronous concatenation is often the intended interpretation when describing a communication protocol as a sequence of phases, with a common set of participating agents across the different phases. However, if M_1 and M_2 are phases involving disjoint sets of agents, it is unnatural to force all events in M_1 to occur before those of M_2 . A more natural version of synchronous concatenation is the anchored version.

Let $M_i = (E_i, \leq_i, \lambda_i)$ with $i = 1, 2$ be plain MSCs. The *anchored concatenation* $M_1 \odot M_2$ of M_1 and M_2 is the MSC $M = (E, \leq, \lambda)$ where E is the disjoint union of E_1 and E_2 and \leq is the reflexive, transitive closure of $\leq_1 \cup \leq_2 \cup \prec$, where $\prec = \{(e_1, e_2) \mid e_1 \in E_1, e_2 \in E_2, loc(e_2) \subseteq loc(M_1)\}$.

To study anchored concatenation, we shall work with a specific subclass of our MSCs called *episodes*. An *episode* is an MSC $M = (E, \leq, \lambda)$ which contains a synchronization event e such that $E_{syn} = \{e\}$, $\downarrow e = E$ and $\lambda(e) = loc(E \setminus \{e\})$. In other words, an episode is a plain MSC equipped with a terminal synchronization event involving all the processes that participate in the “body” of the MSC. The requirement that the body of an episode should be a plain MSC is only for technical ease and is not an essential restriction.

Observe that for episodes, concatenation coincides with anchored concatenation. Hence, while working with episodes, we dispense with \odot and just use \circ to denote (anchored) concatenation.

We are now ready to define the main objects of our study in this new setting, namely, HMSCs.

HMSCs

An HMSC (over \mathcal{X}) is a structure $\mathcal{G} = (Q, \rightarrow, Q_{in}, F, \mathcal{X}, \Phi)$ where

- Q is a finite and nonempty set of states.
- $\rightarrow \subseteq Q \times Q$.
- $Q_{in} \subseteq Q$ is a set of initial states.
- $F \subseteq Q$ is a set of final states.
- \mathcal{X} is a finite set of episodes.
- $\Phi : Q \rightarrow \mathcal{X}$ is a labelling function.

A path π through an HMSC \mathcal{G} is a sequence $q_0 \rightarrow q_1 \rightarrow \dots \rightarrow q_n$ such that $(q_{i-1}, q_i) \in \rightarrow$ for $i \in \{1, 2, \dots, n\}$. The MSC generated by π is $M(\pi) = M_0 \circ M_1 \circ \dots \circ M_n$ where $M_i = \Phi(q_i)$. We say π is a *run* iff $q_0 \in Q_{in}$ and $q_n \in F$. The *MSC language* of \mathcal{G} is $\mathcal{L}(\mathcal{G}) = \{M(\pi) \mid \pi \text{ is a run through } \mathcal{G}\}$.

For an MSC M , we define the *communication graph* CG_M of M to be the *undirected* graph (\mathcal{P}, \mapsto) , where $(p, q) \in \mapsto$ iff there exists $e \in E$ with $\lambda(e) = p!q(m)$ or $\lambda(e) = p?q(m)$ or $\{p, q\} \subseteq \lambda(e)$. Note that this definition of CG_M is slightly different from the one used for asynchronous concatenation [3, 9], where a *directed* graph is constructed reflecting the flow of information through messages between processes. We say an HMSC \mathcal{G} is *locally synchronized* iff for every cycle $\pi = q \rightarrow q_1 \rightarrow q_2 \rightarrow \dots \rightarrow q_n \rightarrow q$, the communication graph of $M(\pi)$ consists of a single connected component (and isolated vertices).

We extend the concatenation operation \circ to MSC languages in the obvious way. That is, for $\mathcal{L}_1, \mathcal{L}_2 \subseteq \mathcal{M}$, $\mathcal{L}_1 \circ \mathcal{L}_2 = \{M_1 \circ M_2 \mid M_1 \in \mathcal{L}_1, M_2 \in \mathcal{L}_2\}$. Let \mathcal{X} be a set of episodes. Define $\mathcal{X}^1 = \mathcal{X}$ and $\mathcal{X}^{n+1} = \mathcal{X}^n \circ \mathcal{X}$. The MSC language $\mathcal{X}^* = \cup_{n \geq 1} \mathcal{X}^n$ is the *iteration* of \mathcal{X} . An MSC language \mathcal{L} is said to be *finitely generated* iff $\mathcal{L} \subseteq \mathcal{X}^*$ for some finite set \mathcal{X} of episodes.

Implied scenarios

For $\sigma \in A^*$ and $B \subseteq A$, let $\sigma \upharpoonright B$ denote the projection of σ onto B —that is, the string obtained from σ by erasing letters not in B . Let $M = (E, \leq, \lambda)$ be an MSC. For $p \in \mathcal{P}$, \leq_{pp} is a linear order. Hence, all linearizations $\sigma \in \text{lin}(M)$ generate the same sequence $\sigma \upharpoonright \Sigma_p$. We denote this sequence by $M \upharpoonright p$. We say that an MSC M is *implied* by the MSC language \mathcal{L} iff for each $p \in \mathcal{P}$, there exists $M_p \in \mathcal{L}$ with $M \upharpoonright p = M_p \upharpoonright p$. The MSC language \mathcal{L} is *closed* with respect to implied scenarios iff every MSC implied by \mathcal{L} is in \mathcal{L} . The *closure* of \mathcal{L} is the smallest closed set of MSCs which contains \mathcal{L} . For an HMSC \mathcal{G} , we denote the closure of $\mathcal{L}(\mathcal{G})$ by $\widehat{\mathcal{L}}(\mathcal{G})$ and call it the *closure* of \mathcal{G} . The MSCs in $\widehat{\mathcal{L}}(\mathcal{G}) \setminus \mathcal{L}(\mathcal{G})$ are called *implied scenarios* of \mathcal{G} . Figure 2 shows an HMSC with an implied scenario.

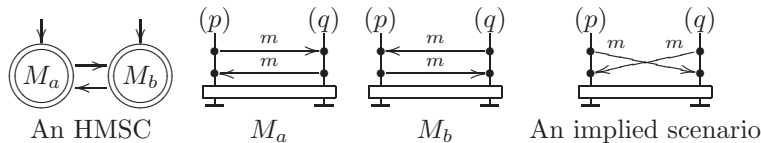


Figure 2: An implied scenario

3 Preliminaries

To avoid tedious repetition, we adopt the following linguistic convention for the rest of the paper.

- By “*our setting*”, we mean the framework in which all HMSC nodes are labelled by episodes and all MSCs are concatenations of episodes. Further, unless stated otherwise, we assume we are in our setting.
- By “*conventional setting*”, we mean the framework where the nodes of HMSCs are labelled by plain MSCs and all the MSCs are asynchronous concatenations of plain MSCs (and are hence themselves plain MSCs.)

We begin by characterizing the linearizations of our MSCs, using a straightforward extension of the results in [9]. For a word σ and a letter a , let $|\sigma|_a$ denote the number of occurrences of a in σ . Recall that Σ denotes our alphabet of communication and synchronization actions. A word $\sigma = a_1 \dots a_n \in \Sigma^*$ is *proper* iff for every $k \in \{1, \dots, n\}$, if $a_k = p?q(m)$, then there exists $j < k$ such that $a_j = q!p(m)$ and $\sum_{m' \in \Delta} |a_1 \dots a_j|_{q!p(m')} = \sum_{m' \in \Delta} |a_1 \dots a_j \dots a_k|_{p?q(m')}$. And further, if $a_k = P \in \Sigma_{syn}$, then for every $\{r, s\} \subseteq P$, $m' \in \Delta$, we have $|a_1 \dots a_k|_{r!s(m')} = |a_1 \dots a_k|_{s?r(m')}$. We say σ is *complete* iff it is proper and $|\sigma|_{p!q(m)} = |\sigma|_{q?p(m)}$ for $(p, q) \in Ch, m \in \Delta$. Let Σ° denote the set of complete words over Σ .

Define a *context-sensitive independence* relation $\mathcal{I} \subseteq \Sigma^* \times (\Sigma \times \Sigma)$ as follows: $(\sigma, a, b) \in \mathcal{I}$ iff σab is proper, $loc(a) \cap loc(b) = \emptyset$, and $|\sigma|_{p!q(m)} > |\sigma|_{q?p(m)}$ whenever $a = p!q(m), b = q?p(m)$. Note that if $(\sigma, a, b) \in \mathcal{I}$, then $(\sigma, b, a) \in \mathcal{I}$. Define $\approx \subseteq \Sigma^\circ \times \Sigma^\circ$ to be the least equivalence relation such that $\sigma ab\sigma' \approx \sigma ba\sigma'$ whenever $\sigma ab\sigma', \sigma ba\sigma' \in \Sigma^\circ$ and $(\sigma, a, b) \in \mathcal{I}$. It is straightforward to establish that \mathcal{M} and Σ°/\approx are in one-to-one correspondence via the mapping $M \mapsto lin(M)$. Thus MSCs can be identified with equivalence classes in Σ°/\approx .

In the conventional setting, the machine model for recognizing a set of MSCs is a *message-passing automaton* (MPA)[9]. We modify this model to handle multi-way synchronization actions and local acceptance conditions. A *product* MPA (over Σ) is a structure $\mathcal{A} = \{\mathcal{A}_p = (S_p, S_p^{in}, \rightarrow_p, F_p) \mid p \in \mathcal{P}\}$ where for each p , S_p is a finite set of local states, $S_p^{in} \subseteq S_p$ a finite set of local initial states, $\rightarrow_p \subseteq S_p \times \Sigma_p \times S_p$ the p -local transition relation, and $F_p \subseteq S_p$ a finite set of local final states.

The set of *global states* of \mathcal{A} is $\prod_{p \in \mathcal{P}} S_p$. For a global state s , we let s_p denote the local state of p in s . A *configuration* is a pair (s, χ) where $s \in \prod_{p \in \mathcal{P}} S_p$ and $\chi : Ch \rightarrow \Delta^*$ specifies the queue of messages currently residing in each channel. The set of *initial configurations* is $Conf_{\mathcal{A}}^{in} = \{(s, \chi_\varepsilon) \mid s \in \prod_{p \in \mathcal{P}} S_p^{in}\}$, where $\chi_\varepsilon : (p, q) \mapsto \varepsilon$ assigns every channel an empty queue. The set of *final configurations* is $\{(s, \chi_\varepsilon) \mid s \in \prod_{p \in \mathcal{P}} F_p\}$. The product MPA \mathcal{A} defines a transition system $(Conf_{\mathcal{A}}, \Sigma, Conf_{\mathcal{A}}^{in}, \Longrightarrow_{\mathcal{A}})$ where the set of *reachable configurations* $Conf_{\mathcal{A}}$ and the transition relation $\Longrightarrow_{\mathcal{A}} \subseteq Conf_{\mathcal{A}} \times \Sigma \times Conf_{\mathcal{A}}$ are defined inductively as follows.

- $Conf_{\mathcal{A}}^{in} \subseteq Conf_{\mathcal{A}}$.
- Suppose $(s, \chi) \in Conf_{\mathcal{A}}$, (s', χ') is a configuration and $p!q(m) \in \Sigma$ such that $(s_p, p!q(m), s'_p) \in \rightarrow_p$, $s_r = s'_r$ for $r \neq p$, $\chi'((p, q)) = \chi((p, q)) \cdot m$ and $\chi'(c) = \chi(c)$ for $c \neq (p, q)$. Then $(s', \chi') \in Conf_{\mathcal{A}}$ and $(s, \chi) \xrightarrow{p!q(m)}_{\mathcal{A}} (s', \chi')$.

- Suppose $(s, \chi) \in \text{Conf}_{\mathcal{A}}$, (s', χ') is a configuration and $p?q(m) \in \Sigma$ such that $(s_p, p?q(m), s'_p) \in \rightarrow_p$, $s_r = s'_r$ for $r \neq p$, $\chi((q, p)) = m \cdot \chi'((q, p))$ and $\chi'(c) = \chi(c)$ for $c \neq (q, p)$. Then $(s', \chi') \in \text{Conf}_{\mathcal{A}}$ and $(s, \chi) \xrightarrow{p?q(m)}_{\mathcal{A}} (s', \chi')$.
- Suppose $(s, \chi) \in \text{Conf}_{\mathcal{A}}$, (s', χ') is a configuration and $P \in \Sigma_{\text{syn}}$ such that $(s_p, P, s'_p) \in \rightarrow_p$ for $p \in P$, $s_r = s'_r$ for $r \notin P$, $\chi = \chi'$, and further, for $c \in \text{Ch} \cap (P \times P)$, $\chi(c) = \varepsilon$. Then $(s', \chi') \in \text{Conf}_{\mathcal{A}}$ and $(s, \chi) \xrightarrow{P}_{\mathcal{A}} (s', \chi')$.

A run of \mathcal{A} over $\sigma \in \Sigma^*$ is a map ρ from the set of prefixes of σ to the reachable configurations of \mathcal{A} such that $\rho(\varepsilon) \in \text{Conf}_{\mathcal{A}}^{\text{in}}$ and for each prefix τa of σ , $\rho(\tau) \xrightarrow{a}_{\mathcal{A}} \rho(\tau a)$. We say that ρ is *accepting* iff $\rho(\sigma)$ is a final configuration. The *language* $L(\mathcal{A})$ accepted by \mathcal{A} is the set of words in Σ^* which have an accepting run. It is easy to observe that $L(\mathcal{A})$ is a \approx -closed subset of Σ° and hence can be viewed as an MSC language. For an integer B , we say that \mathcal{A} is *B-bounded* iff for every channel $c \in \text{Ch}$ and for every reachable configuration (s, χ) of \mathcal{A} , it is the case that $|\chi(c)| \leq B$. It is clear that if \mathcal{A} is *B-bounded*, then $L(\mathcal{A})$ is a regular subset of Σ° . It is also easy to see that the following holds.

Proposition 1. *The MSC language accepted by a product MPA is closed (under implied scenarios).*

4 MSC Languages of HMSCs

We shall show that it is undecidable whether the MSC language of an HMSC is regular whereas the MSC language of a locally synchronized HMSC is always regular. We shall also show that locally synchronized HMSCs capture exactly the class of finitely generated regular MSC languages.

Let $\mathcal{G} = (Q, \rightarrow, Q_{\text{in}}, F, \mathcal{X}, \Phi)$ be an HMSC over a set of episodes \mathcal{X} . We define an independence relation $I_{\mathcal{X}}$ over \mathcal{X} as follows: $(X, Y) \in I_{\mathcal{X}}$ iff $\text{loc}(X) \cap \text{loc}(Y) = \emptyset$. We can then interpret $(\mathcal{X}, I_{\mathcal{X}})$ as a (Mazurkiewicz) trace alphabet. Let $\sim_{\mathcal{X}} \subseteq \mathcal{X}^* \times \mathcal{X}^*$ be the trace equivalence relation induced by $(\mathcal{X}, I_{\mathcal{X}})$. For $L \subseteq \mathcal{X}^*$, the *trace closure* of L with respect to $I_{\mathcal{X}}$ is denoted by $[L]_{I_{\mathcal{X}}}$.

It will be convenient to work with the strings of MSCs generated by an HMSC. To distinguish this language from the language of all linearizations of the MSCs generated by the HMSC, we use the term “episodic-string language” or “e-string language” for short. We define the *e-string language* of \mathcal{G} , $L_e(\mathcal{G})$, to be the set of strings $M_0 M_1 \dots M_n \in \mathcal{X}^*$ for which there exists a run $q_0 \rightarrow q_1 \rightarrow \dots \rightarrow q_n$ with $M_i = \Phi(q_i)$ for $i \in \{0, 1, \dots, n\}$.

Lemma 2. *Let \mathcal{G} be an HMSC over a set of episodes \mathcal{X} . Its MSC language $\mathcal{L}(\mathcal{G})$ is regular iff the trace closure of its e-string language $L_e(\mathcal{G})$ with respect to $I_{\mathcal{X}}$ is a regular subset of \mathcal{X}^* .*

Proof. The proof is immediate from two basic observations.

Firstly, for $M_1 M_2 \dots M_n, M'_1 M'_2 \dots M'_n \in \mathcal{X}^*$, $M_1 M_2 \dots M_n \sim_{\mathcal{X}} M'_1 M'_2 \dots M'_n$ iff $M_1 \circ M_2 \circ \dots \circ M_n = M'_1 \circ M'_2 \circ \dots \circ M'_n$. It follows that $\mathcal{L}(\mathcal{G}) = \{M_1 \circ M_2 \circ \dots \circ M_n \mid M_1 M_2 \dots M_n \in [L_e(\mathcal{G})]_{I_{\mathcal{X}}}\}$.

Secondly, we can effectively construct a *finite transduction* ([21]) $\varphi : \text{lin}(\mathcal{X}^\circ) \rightarrow \mathcal{X}^*$ such that for $\tau = b_1 \dots b_n \in \text{lin}(\mathcal{X}^\circ)$, $\varphi(\tau) = M_1 \dots M_n \in \mathcal{X}^*$ where $\tau \in \text{lin}(M_1 \circ \dots \circ M_n)$ and $\tau \upharpoonright_{\Sigma_{\text{syn}}} = P_1 \dots P_n$ with $P_i = M_i \upharpoonright_{\Sigma_{\text{syn}}}$ for $i \in \{1, \dots, n\}$. It then follows that $\varphi(\text{lin}(\mathcal{L}(\mathcal{G}))) = [L_e(\mathcal{G})]_{I_{\mathcal{X}}}$.

In $\tau = b_1 \dots b_n$, let j be the least index such that $b_j \in \Sigma_{syn}$. We can effectively identify a *unique* episode $X \in \mathcal{X}$ such that $lin(X)$ is a subsequence of $b_1 \dots b_j$. Further, for any action b_i in $b_1 \dots b_j$ that is not from X , we have $loc(b_i) \cap loc(X) = \emptyset$. Thus, we can reorder τ as $w_X w' b_{j+1} \dots b_n$ where $w_X \in lin(X)$ and w' is the subsequence of $b_1 \dots b_j$ obtained by erasing all the actions from $lin(X)$. For $w' b_{j+1} \dots b_n$, we can inductively identify a sequence $M_2 \dots M_n \in \mathcal{X}^*$, as required. For τ , the corresponding sequence is then $X M_2 \dots M_n$. \square

Theorem 3. *There is no effective decision procedure to determine if the MSC language of an HMSC is regular.*

Proof. It is known ([19]) that it is undecidable if the trace closure of a regular language $L \subseteq A^*$ with respect to a trace alphabet (A, I) is regular. We reduce this problem to ours. Let (A_1, \dots, A_n) be a distributed alphabet implementing (A, I) . Create a set of processes $\mathcal{P} = \{p_i, p'_i \mid i \in \{1, \dots, n\}\}$ and a message alphabet $\Delta = A$. Encode each $a \in A$ by an episode M_a shown in Fig. 3, where $A_{i_1}, A_{i_2}, \dots, A_{i_k}$ are the components containing a . Construct an HMSC \mathcal{G} over $\mathcal{X} = \{M_a \mid a \in A\}$ with $L_e(\mathcal{G}) = L$. It follows that $I = I_{\mathcal{X}}$. By Lemma 2, $[L]_I$ is regular iff $\mathcal{L}(\mathcal{G})$ is regular. \square

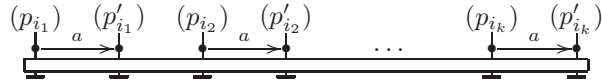


Figure 3: The episode M_a

Theorem 4. *The MSC language of a locally synchronized HMSC is regular.*

Proof. Let $\mathcal{G} = (Q, \rightarrow, Q_{in}, F, \mathcal{X}, \Phi)$ be a locally synchronized HMSC. By Lemma 2, it suffices to show that $[L(\mathcal{G})]_{I_{\mathcal{X}}}$ is regular. Observe that the communication graph of every episode is a complete graph. Hence, for $\sigma = M_1 \dots M_n \in \mathcal{X}^*$, the communication graph of $M_1 \circ \dots \circ M_n$ is connected iff σ is a *connected trace* [6, Sec. 6.3.1]. It is known that if $L \subseteq \mathcal{X}^*$ is regular and every word in L is connected, then $[L^*]_{I_{\mathcal{X}}}$ is also regular [6, Prop. 6.3.11]. The claim then follows. \square

Theorem 5. *Every finitely generated regular MSC language can be represented as the MSC language of a locally synchronized HMSC.*

Proof. Let $\mathcal{L} \subseteq \mathcal{X}^{\otimes}$ be a regular MSC language where \mathcal{X} is a finite set of episodes. Following the proof of Lemma 2, there exists a regular trace language $L \subseteq \mathcal{X}^*$ such that $\mathcal{L} = \{M_1 \circ \dots \circ M_n \mid M_1 \dots M_n \in L\}$. Fix a strict linear order \sqsubseteq on \mathcal{X} , which then induces a lexicographic order \sqsubseteq on \mathcal{X}^* . Define $LEX \subseteq \mathcal{X}^*$ as follows: $\sigma \in LEX$ iff σ is the \sqsubseteq -least element in the trace containing σ . Set $lex(L) = L \cap LEX$. Following [6, Sec. 6.3.1], we have the following:

- $lex(L)$ is a regular subset of \mathcal{X}^* and $L = [lex(L)]_{I_{\mathcal{X}}}$.
- If $\sigma_1 \sigma_2 \in LEX$, then $\sigma \in LEX$.
- If $\sigma \in \mathcal{X}^*$ is *not* connected, then $\sigma \notin LEX$.

Create an HMSC \mathcal{G} such that $L(\mathcal{G}) = lex(L)$. It then follows that $\mathcal{L} = \mathcal{L}(\mathcal{G})$ and \mathcal{G} is locally synchronized. \square

5 Closure of HMSCs with respect to implied scenarios

In the conventional setting, it is easy to observe that the closure of an MSC language defined by an HMSC is, in general, *not* regular. A trivial example is the HMSC whose MSC language is $\{M\}^\circledast$, where M is the MSC whose sole linearization is $p!q(m) q?p(m)$. The closure of this language is itself and it is obviously not regular. In fact, it is not difficult to show it is undecidable if the closure of a (locally synchronized) HMSC is regular. However, in our setting, the closure of an HMSC language is *always* regular.

Theorem 6. *The closure of every HMSC language is regular.*

Proof. Let $\mathcal{G} = (Q, \rightarrow, Q_{in}, F, \mathcal{X}, \Phi)$ be an HMSC. We construct a *bounded* product MPA $\mathcal{A} = \{\mathcal{A}_p = (S_p, S_p^{in}, \rightarrow_p, F_p) \mid p \in \mathcal{P}\}$ accepting $\widehat{\mathcal{L}}(\mathcal{G})$ as follows. For $p \in \mathcal{P}$, set L_p to be the projection of $\text{lin}(\mathcal{L}(\mathcal{G}))$ onto Σ_p . It is easy to see that each L_p is regular. Set \mathcal{A}_p to be the minimal deterministic finite state automaton accepting L_p . It follows that \mathcal{A} accepts $\widehat{\mathcal{L}}(\mathcal{G})$. It is easy to observe that \mathcal{A} is bounded by the maximum length of $\{X \upharpoonright p \mid X \in \mathcal{X}\}$. \square

From the proof of Theorem 6, it follows that the closure of an HMSC language can be effectively represented as a bounded product MPA. Hence the set of linearizations of the MSCs in the closure of an HMSC language can also be effectively computed. From Theorem 4 and the fact that the equivalence of regular string languages can be effectively determined, the next result is immediate.

Corollary 7. *We can effectively decide whether a locally synchronized HMSC admits an implied scenario.*

In the conventional setting, it is easy to observe that the closure of an HMSC language is in general *not* finitely generated. A simple example is the HMSC whose MSC language is $\{M_1, M_2\}^\circledast$, where M_1 (respectively M_2) is the MSC whose sole linearization is $p!q(m) q?p(m)$ (respectively $q!p(m) p?q(m)$). However, in our setting, the closure of an HMSC is always finitely generated.

Theorem 8. *The closure of every HMSC language is finitely generated.*

Proof. Let $\mathcal{G} = (Q, \rightarrow, Q_{in}, F, \mathcal{X}, \Phi)$ be an HMSC. Let \mathcal{Y} be the set of episodes M such that for each $p \in \mathcal{P}$, there exists $M_p \in \mathcal{X}$ with $M \upharpoonright p = M_p \upharpoonright p$. Let \mathcal{H} be an HMSC with $L(\mathcal{H}) = \mathcal{X}^*$. Since $\widehat{\mathcal{L}}(\mathcal{G}) \subseteq \widehat{\mathcal{L}}(\mathcal{H})$, it suffices to show that $\widehat{\mathcal{L}}(\mathcal{H}) \subseteq \mathcal{Y}^\circledast$.

Let $M = (E, \leq, \lambda) \in \widehat{\mathcal{L}}(\mathcal{H})$. Note that for any $M' \in \mathcal{L}(\mathcal{H})$, all maximal events in M' are synchronization events. Hence all maximal events in M are synchronization events too. Pick $e \in E_{syn}$ such that $\downarrow e \cap E_{syn} = \{e\}$. We shall show that $Y = (\downarrow e, \leq_{\downarrow e}, \lambda_{\downarrow e}) \in \mathcal{Y}$, where, $\leq_{\downarrow e}$ and $\lambda_{\downarrow e}$ are, respectively, the restrictions of \leq and λ to $\downarrow e$. With this, we can remove Y from M , and it is clear that inductively $M \in \mathcal{Y}^\circledast$.

It remains to prove that Y is an episode. Set $P = \lambda(e)$ and pick $p \in P$. There must exist $X \in \mathcal{X}$ such that $X \upharpoonright p = Y \upharpoonright p$ and $\text{loc}(X) = P$. Hence for any $e' < e$, if $\lambda(e') = p!q(m)$ or $\lambda(e') = p?q(m)$, then $q \in P$. It follows that $P = \text{loc}(Y)$. \square

The proof above also yields the following useful observation.

Corollary 9. *Let \mathcal{G} be an HMSC over a set of episodes \mathcal{X} such that $\mathcal{L}(\mathcal{G}) = \mathcal{X}^\circledast$. Then \mathcal{G} admits no implied scenario iff $\mathcal{X} = \{M \mid M \text{ is an episode and } \forall p \exists X_p \in \mathcal{X}, X_p \upharpoonright p = M \upharpoonright p\}$.*

The following result however mirrors the situation in the conventional setting.

Theorem 10. *It is undecidable whether an HMSC admits an implied scenario.*

Proof. We shall make use of the reduction from the Post Correspondence Problem (PCP) in [17] for proving the undecidability of determining if the trace closure of a star-free language remains star-free. An instance of PCP consists of two morphisms $g, h : K^* \rightarrow \Gamma^*$ where K, Γ are disjoint finite alphabets. A solution is a word $w \in K^+$ such that $g(w) = h(w)$.

We briefly describe the main ingredients of the reduction in [17]. Create a trace alphabet (A, I) where $A = K \cup \Gamma \cup \{c\}$, $c \notin K \cup \Gamma$ and $I = \{(x, c), (c, x) \mid x \in K \cup \Gamma\}$. Define W_g to be the trace closure with respect to I of $\{w \cdot g(w) \cdot c^{|g(w)|} \mid w \in K^+\}$ and a regular language $L_g \subseteq A^*$ such that $[L_g]_I = A^* \setminus W_g$. Analogously define W_h and L_h . The construction has the following property. If the PCP instance has no solution, then $[L_g \cup L_h]_I = A^*$. Otherwise, $[L_g \cup L_h]_I$ is *not regular*.

As in the proof of Theorem 3, we construct an HMSC \mathcal{G} over $\mathcal{X} = \{M_a \mid a \in A\}$ using the *distributed alphabet* $(K \cup \Gamma, \{c\})$. If $[L_g \cup L_h]_I = A^*$, then $\mathcal{L}(\mathcal{G}) = \mathcal{X}^{\otimes}$, and $\mathcal{L}(\mathcal{G})$ is easily seen to admit no implied scenario by Corollary 9. If not, then $[L_g \cup L_h]_I$ is not regular and thus $\mathcal{L}(\mathcal{G})$ is not regular. Consequently \mathcal{G} must admit an implied scenario, by Theorem 6. Thus \mathcal{G} admits an implied scenario iff the original instance of PCP has a solution. \square

6 Conclusions

We have proposed here the notion of anchored concatenation and studied MSC languages defined by HMSCs under this operation. Our results show that the resulting theory is non-trivial and bears both commonalities and differences with the corresponding theory in the conventional setting.

We have considered here only finite MSCs. It will be interesting to explore our theory for infinite MSCs by adapting the techniques developed in [13]. It will also be worthwhile to consider realizations in the form of netcharts [4, 15] instead of product MPAs.

References

- [1] R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence graphs. In *Proc. of ICSE '00*, pages 304–313. ACM, 2000.
- [2] R. Alur, K. Etessami, and M. Yannakakis. Realizability and verification of MSC graphs. In *ICALP '01, LNCS 2076*, pages 797–808. Springer, 2001.
- [3] R. Alur and M. Yannakakis. Model checking of message sequence charts. In *CONCUR '99, LNCS 1664*, pages 114–129. Springer, 1999.
- [4] N. Baudru and R. Morin. The pros and cons of netcharts. In *CONCUR '04, LNCS 3170*, pages 99–114. Springer, 2004.
- [5] G. Booch, I. Jacobson, and J. Rumbaugh. *Unified Modeling Language User Guide*. Addison-Wesley, 1997.
- [6] V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, 1995.

- [7] D. Harel and E. Gery. Executable object modeling with statecharts. *IEEE Computer*, 31(7):31–42, 1997.
- [8] D. Harel and R. Marelly. *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer, 2003.
- [9] J.G. Henriksen, M. Mukund, K. Narayan Kumar, M. Sohoni, and P.S. Thiagarajan. A theory of regular MSC languages. *Information and Computation*, 202(1):1–38, 2005.
- [10] J.G. Henriksen, M. Mukund, K. Narayan Kumar, and P.S. Thiagarajan. On message sequence graphs and finitely generated regular MSC languages. In *ICALP '00, LNCS 1854*, pages 675–686. Springer, 2000.
- [11] J.G. Henriksen, M. Mukund, K. Narayan Kumar, and P.S. Thiagarajan. Regular collections of message sequence charts. In *MFCS '00, LNCS 1893*, pages 405–414. Springer, 2000.
- [12] ITU-TS. ITU-TS Recommendation Z.120: Message sequence charts. 1997.
- [13] D. Kuske. A further step towards a theory of regular MSC languages. In *STACS '02, LNCS 2285*, pages 489–500. Springer, 2002.
- [14] S. Mauw and M.A. Renier. High-level message sequence charts. In *Proc. of SDL '97: Time for Testing – SDL, MSC and Trends*, pages 291–306. Elsevier, 1997.
- [15] M. Mukund, K. Narayan Kumar, and P.S. Thiagarajan. Netcharts: Bridging the gap between HMSCs and executable specifications. In *CONCUR '03, LNCS 2761*, pages 296–310. Springer, 2003.
- [16] A. Muscholl and D. Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. In *MFCS '99, LNCS 1672*, pages 81–91. Springer, 1999.
- [17] A. Muscholl and H. Peterson. A note on the commutative closure of star-free languages. *Information Processing Letters*, 57(2):71–74, 1996.
- [18] E. Rudolph, P. Graubmann, and J. Grabowski. Tutorial on message sequence charts. In *Comp. Networks and ISDN Sys. – SDL and MSC 28*. 1996.
- [19] J. Sakarovitch. The “last” decision problem for rational trace languages. In *LATIN'92, LNCS 583*, pages 460–473. Springer, 1992.
- [20] S. Uchitel, J. Kramer, and J. Magee. Detecting implied scenarios in message sequence chart specifications. In *Proc. of FSE '01*. ACM, 2001.
- [21] S. Yu. Regular languages. In *Handbook of Formal Languages, Vol. 1*. 1997.