FCT 2007, 16th Int Symp on Fundamentals of Computation Theory
Proceedings: Erzsébet Csuhaj-Varjú, Zoltán Ésik (eds.)
Springer Lecture Notes in Computer Science (2007), *to appear*

Local testing of message sequence charts is difficult ⋆

Puneet Bhateja[1], Paul Gastin[2],
Madhavan Mukund[1], and K. Narayan Kumar[1]

[1] Chennai Mathematical Institute, Chennai, India
{puneet,madhavan,kumar}@cmi.ac.in
[2] LSV, ENS Cachan & CNRS, France
Paul.Gastin@lsv.ens-cachan.fr

Abstract. Message sequence charts are an attractive formalism for specifying communicating systems. One way to test such a system is to substitute a component by a test process and observe its interaction with the rest of the system. Unfortunately, local observations can combine in unexpected ways to define implied scenarios not present in the original specification. Checking whether a scenario specification is closed with respect to implied scenarios is known to be undecidable when observations are made one process at a time. We show that even if we strengthen the observer to be able to observe multiple processes simultaneously, the problem remains undecidable. In fact, undecidability continues to hold even without message labels, provided we observe two or more processes simultaneously. On the other hand, without message labels, if we observe one process at a time, checking for implied scenarios is decidable.

1 Introduction

Message Sequence Charts (MSCs) [7] are an appealing visual formalism that are used in a number of software engineering notational frameworks such as SDL [15] and UML [4]. A collection of MSCs is used to capture the scenarios that a designer might want the system to exhibit (or avoid).

A standard way to generate a set of MSCs is via Hierarchical (or High-level) Message Sequence Charts (HMSCs) [10]. Without losing expressiveness, we consider only a subclass of HMSCs called Message Sequence Graphs (MSGs). An MSG is a finite directed graph in which each node is labeled by an MSC. An MSG defines a collection of MSCs by concatenating the MSCs labeling each path from an initial vertex to a terminal vertex.

A natural way to test a distributed implementation against an MSG specification is to substitute test processes for one or more components and record the interactions between the test process(es) and the rest of the system. We refer to this form of testing of distributed message-passing systems as *local testing*.

⋆ Partially supported by *Timed-DISCOVERI*, a project under the Indo-French Networking Programme.

The implementation is said to pass a local test if the observations at the test process(es) are consistent with the MSG specification.

An important impediment to local testing is the possibility of implied scenarios. Let $T = \{P_1, P_2, \ldots, P_k\}$ be a collection of subsets of processes. We say that an MSC M is T-implied by an MSC language $\mathcal{L}$ if the projections of M onto each subset $P_i \in T$ agree with the projections onto P_i of some good MSC $M_{P_i} \in \mathcal{L}$. Implied scenarios have been studied in [2, 3], where the observations are restricted to individual processes rather than arbitrary subsets.

Let T_k denote the set of all subsets of processes of size k. We say that an MSC language $\mathcal{L}$ is *k-testable* if every T_k-implied scenario is already present in $\mathcal{L}$. In other words, if a specification is k-testable, it is possible to accurately test an implementation by performing a collection of local tests with respect to T_k. On the other hand, if $\mathcal{L}$ is not k-testable, even an exhaustive set of local tests with respect to T_k cannot rule out an undesirable implied scenario.

It has been shown in [3] that 1-testability is undecidable, even for regular MSG specifications. (The results of [3] are formulated in the context of distributed synthesis, but they can also be interpreted in terms of local testing.) We extend the results of [3] to show that for any n, k-testability of an MSG specification with n processes is undecidable, for all $k \in \{1, 2, \ldots, n-1\}$.

We also consider MSG specifications over n processes without message labels. Somewhat surprisingly, k-testability remains undecidable for $k \in \{2, \ldots, n-1\}$. However, for unlabelled MSG specifications, 1-testability is decidable.

The paper is organized as follows. We begin with preliminaries about MSCs, before we formally define k-testability in Section 3. The next section establishes various undecidability results. In Section 5, we show that 1-testability is decidable for unlabelled MSG specifications. We conclude with a brief discussion.

2 Preliminaries

2.1 Message sequence charts

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a finite set of processes (agents) that communicate with each other through messages via reliable FIFO channels using a finite set of message types $\mathcal{M}$. For $p \in \mathcal{P}$, let $\Sigma_p = \{p!q(m), p?q(m) \mid p \neq q \in \mathcal{P}, m \in \mathcal{M}\}$ be the set of communication actions in which p participates. The action $p!q(m)$ is read as *p sends the message m to q* and the action $p?q(m)$ is read as *p receives the message m from q*. We set $\Sigma = \bigcup_{p \in \mathcal{P}} \Sigma_p$. We also denote the set of *channels* by $Ch = \{(p, q) \in \mathcal{P}^2 \mid p \neq q\}$.

Labelled posets A Σ-labelled poset is a structure $M = (E, \leq, \lambda)$ where $(E, \leq)$ is a partially ordered set and $\lambda : E \to \Sigma$ is a labelling function. For $e \in E$, let $\downarrow e = \{e' \mid e' \leq e\}$. For $p \in \mathcal{P}$ and $a \in \Sigma$, we set $E_p = \{e \mid \lambda(e) \in \Sigma_p\}$ and $E_a = \{e \mid \lambda(e) = a\}$, respectively. For $(p, q) \in Ch$, we define the relation $<_{pq}$:

$$e <_{pq} e' \stackrel{\text{def}}{=} \exists m \in \mathcal{M} \text{ such that } \lambda(e) = p!q(m),\ \lambda(e') = q?p(m) \text{ and } |\downarrow e \cap E_{p!q(m)}| = |\downarrow e' \cap E_{q?p(m)}|$$

The relation $e <_{pq} e'$ says that channels are FIFO with respect to *each message*—if $e <_{pq} e'$, the message m read by q at e' is the one sent by p at e.

Finally, for each $p \in \mathcal{P}$, we define the relation $\leq_{pp} = (E_p \times E_p) \cap \leq$, with $<_{pp}$ standing for the largest irreflexive subset of $\leq_{pp}$.

Definition 1. *An MSC over $\mathcal{P}$ is a* finite *Σ-labelled poset $M = (E, \leq, \lambda)$ where:*

1. *Each relation $\leq_{pp}$ is a linear (total) order.*
2. *If $p \neq q$ then for each $m \in \mathcal{M}$, $|E_{p!q(m)}| = |E_{q?p(m)}|$.*
3. *If $e <_{pq} e'$, then $|{\downarrow}e \cap \left(\bigcup_{m\in\mathcal{M}} E_{p!q(m)}\right)| = |{\downarrow}e' \cap \left(\bigcup_{m\in\mathcal{M}} E_{q?p(m)}\right)|$.*
4. *The partial order $\leq$ is the reflexive, transitive closure of $\bigcup_{p,q\in\mathcal{P}} <_{pq}$.*

The second condition ensures that every message sent along a channel is received. The third condition says that every channel is FIFO across all messages.

In diagrams, the events of an MSC are presented in *visual order*. The events of each process are arranged in a vertical line and messages are displayed as horizontal or downward-sloping directed edges. Fig. 1 shows an example with three processes $\{p, q, r\}$ and six events $\{e_1, e'_1, e_2, e'_2, e_3, e'_3\}$ corresponding to three messages—m_1 from p to q, m_2 from q to r and m_3 from p to r.

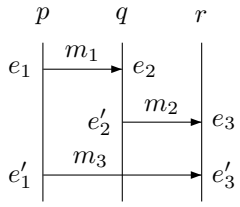

Fig. 1. An MSC

For an MSC $M = (E, \leq, \lambda)$, we let $\mathrm{lin}(M) = \{\lambda(\pi) \mid \pi \text{ is a linearization of } (E, \leq)\}$. For instance, $p!q(m_1)\ q?p(m_1)\ q!r(m_2)\ p!r(m_3)\ r?q(m_2)\ r?p(m_3)$ is one linearization of the MSC in Fig. 1.

MSC languages An *MSC language* is a set of MSCs. We can also regard an MSC language $\mathcal{L}$ as a word language over Σ given by $\mathrm{lin}(\mathcal{L}) = \bigcup\{\mathrm{lin}(M) \mid M \in \mathcal{L}\}$.

Definition 2. *An MSC language $\mathcal{L}$ is said to be a* regular MSC language *if the word language* $\mathrm{lin}(\mathcal{L})$ *is a regular language over Σ.*

Let M be an MSC and $B \in \mathbb{N}$. We say that $w \in \mathrm{lin}(M)$ is B-bounded if for every prefix v of w and for every channel $(p, q) \in Ch$, $\sum_{m\in\mathcal{M}} |\pi_{p!q(m)}(v)| - \sum_{m\in\mathcal{M}} |\pi_{q?p(m)}(v)| \leq B$, where $\pi_\Gamma(v)$ denotes the projection of v on $\Gamma \subseteq \Sigma$. This means that along the execution of M described by w, no channel ever contains more than B-messages. We say that M is (universally) B-bounded if every $w \in \mathrm{lin}(M)$ is B-bounded. An MSC language $\mathcal{L}$ is B-bounded if every $M \in \mathcal{L}$ is B-bounded. Finally, $\mathcal{L}$ is bounded if it is B-bounded for some B.

We then have the following result [5].

Theorem 3. *If an MSC language $\mathcal{L}$ is regular then it is bounded.*

2.2 Message sequence graphs

Message sequence graphs (MSGs) are finite directed graphs with designated initial and terminal vertices. Each vertex in an MSG is labelled by an MSC. The edges represent (asynchronous) MSC concatenation, defined as follows.

Let $M_1 = (E^1, \leq^1, \lambda_1)$ and $M_2 = (E^2, \leq^2, \lambda_2)$ be a pair of MSCs such that E^1 and E^2 are disjoint. The *(asynchronous) concatenation* of M_1 and M_2 yields the MSC $M_1 \circ M_2 = (E, \leq, \lambda)$ where $E = E^1 \cup E^2$, $\lambda(e) = \lambda_i(e)$ if $e \in E^i$, $i \in \{1, 2\}$, and $\leq = (\leq^1 \cup \leq^2 \cup \bigcup_{p \in \mathcal{P}} E^1_p \times E^2_p)^*$.

A *Message Sequence Graph* is a structure $\mathcal{G} = (Q, \rightarrow, Q_{in}, F, \Phi)$, 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 and Φ labels each state with an MSC.

A *path* π through an MSG $\mathcal{G}$ is a sequence $q_0 \rightarrow q_1 \rightarrow \cdots \rightarrow q_n$ such that $(q_{i-1}, q_i) \in \rightarrow$ for $i \in \{1, 2, \ldots, n\}$. The MSC generated by π is $M(\pi) = M_0 \circ M_1 \circ M_2 \circ \cdots \circ M_n$, where $M_i = \Phi(q_i)$. A path $\pi = q_0 \rightarrow q_1 \rightarrow \cdots \rightarrow q_n$ is a *run* if $q_0 \in Q_{in}$ and $q_n \in F$. The language of MSCs accepted by $\mathcal{G}$ is $L(\mathcal{G}) = \{M(\pi) \mid \pi \text{ is a run through } \mathcal{G}\}$. We say that an MSC language $\mathcal{L}$ is *MSG-definable* if there exists and MSG $\mathcal{G}$ such that $\mathcal{L} = L(\mathcal{G})$.

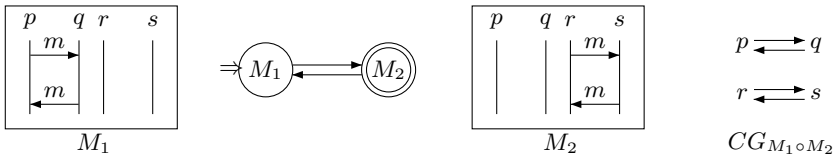

Fig. 2. A message sequence graph

An example of an MSG is depicted in Fig. 2. The initial state is marked $\Rightarrow$ and the final state has a double line. The language $\mathcal{L}$ defined by this MSG is *not* regular: $\mathcal{L}$ projected to $\{p!q(m), r!s(m)\}^*$ consists of $\sigma \in \{p!q(m), r!s(m)\}^*$ such that $|\sigma\restriction_{p!q(m)}| = |\sigma\restriction_{r!s(m)}| \geq 1$, which is not a regular string language.

In general, it is undecidable whether an MSG describes a regular MSC language [5]. However, a sufficient condition for the MSC language of an MSG to be regular is that the MSG be *locally synchronized*.

Communication graph For an MSC $M = (E, \leq, \lambda)$, let CG_M, *the communication graph of* M, be the directed graph $(\mathcal{P}, \mapsto)$ where:

- $\mathcal{P}$ is the set of processes of the system.
- $(p, q) \in \mapsto$ iff there exists an $e \in E$ with $\lambda(e) = p!q(m)$.

M is said to be *com-connected* if CG_M consists of one nontrivial strongly connected component and isolated vertices.

Locally synchronized MSGs The MSG $\mathcal{G}$ is *locally synchronized* [12] (or *bounded* [1]) if for every loop $\pi = q \rightarrow q_1 \rightarrow \cdots \rightarrow q_n \rightarrow q$, the MSC $M(\pi)$ is com-connected. In Fig. 2, $CG_{M_1 \circ M_2}$ is not com-connected, so the MSG is not locally synchronized. We have the following result for MSGs [1].

Theorem 4. *If $\mathcal{G}$ is locally synchronized, $L(\mathcal{G})$ is a regular MSC language.*

3 Locally testable MSC languages

In local testing, we substitute test process(es) for one or more components and record the interactions between the test process(es) and the rest of the system. The implementation is said to pass a local test if the observations at the test process(es) are consistent with the MSG specification. An important impediment to local testing is the possibility of implied scenarios.

Definition 5. *Let $M = (E, \leq, \lambda)$ be an MSC and $P \subseteq \mathcal{P}$ a set of processes. The* P-observation of M, *$M{\restriction_P}$, is the collection of local observasions $\{(E_p, \leq_{pp})\}_{p \in P}$, where $\leq_{pp} = \leq \cap (E_p \times E_p)$. The collection $\{(E_p, \leq_{pp})\}_{p \in P}$ can also be viewed as a labelled partial order $(E_P, \leq_P)$ where $E_P = \bigcup_{p \in P} E_p$ and $\leq_p = \left(\bigcup_{p,q \in P} <_{pq}\right)^*$.*

Let $T \subseteq 2^{\mathcal{P}}$ be a family of subsets of processes. An MSC M is said to be T-implied *by an MSC-language $\mathcal{L}$ if for every subset $P \in T$ there is an MSC $M_P \in \mathcal{L}$ such that $M_P{\restriction_P} = M{\restriction_P}$.*

We denote by T_k the set $\{P \subseteq \mathcal{P} \mid |P| = k\}$ of all subsets of $\mathcal{P}$ of size k and we say that an MSC is k-implied if it is T_k-implied.

Fig. 3 illustrates the idea of implied scenarios. The MSC M' is 1-implied by $\{M_1, M_2\}$. However, M' is not 2-implied by $\{M_1, M_2\}$ because the $\{p, s\}$-observation of M' does not match either M_1 or M_2.

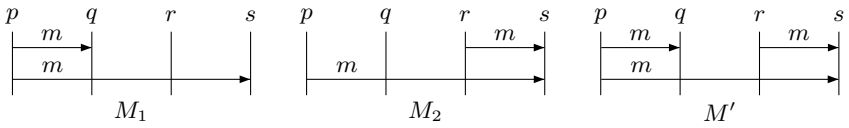

Fig. 3. An example of implied scenarios

We are interested in checking the global behaviour of a distributed implementation by testing it locally against an MSG specification. For this to be meaningful, the MSG should be closed with respect to implied scenarios generated by the test observations. This leads to the following definition.

Definition 6. *Let $|\mathcal{P}| = n$. An MSG $\mathcal{G}$ is said to be k-testable if every scenario M that is k-implied by $L(\mathcal{G})$ is already a member of $L(\mathcal{G})$.*

We have the following negative result from [3] (adapted to our context).

Theorem 7. *Let $\mathcal{G}$ be a locally-synchronized MSG, so that $L(\mathcal{G})$ is a regular MSC language. It is undecidable whether $L(\mathcal{G})$ is 1-testable.*

This result is somewhat surprising, since the analogous problem for synchronous systems is decidable [16]. The root cause of this undecidability is the fact that even when a MSC language $\mathcal{L}$ is regular, and hence B-bounded for some B, the set of scenarios implied by $\mathcal{L}$ may not be bounded. An example is shown in Fig. 4—all messages are labelled m and labels are omitted.

Since M_1 and M_2 are both comconnected, the language $(M_1 + M_2)^*$ is a regular MSG-definable language.

On the other hand, for each $k \in \mathbb{N}$, the MSC in which the p-observation matches $M_1^{2k}M_2^k$ and the q-observation matches $M_2^kM_1^{2k}$ has a global cut where the channel (p, q) has capacity $k + 1$. The figure shows the case $k = 2$. The dotted line marks the global cut where the channel (p, q) has maximum capacity.

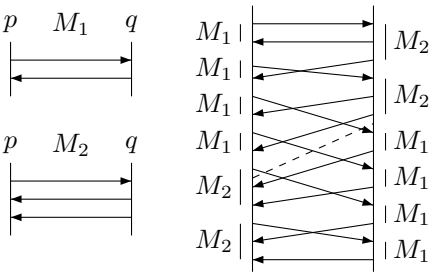

Fig. 4.

4 Undecidability

We know from [3] that 1-testability is undecidable for regular MSG-definable languages. The example in Fig. 3 suggests that it might be possible to determine the smallest $k < n$ such that an MSG specification with n processes is k-testable. (Observe that every MSC language over n processes is trivially n-testable.) Unfortunately, this is not the case. For all $k < n$, the problem of determining whether a regular MSG specification is k-testable is undecidable.

The undecidability proofs in this section use reductions from the *Modified Post's Correspondence Problem* (MPCP) [6]. An instance of MPCP is a collection $\{(v_1, w_1), (v_2, w_2) \ldots, (v_r, w_r)\}$ of pairs of words over an alphabet Σ. A solution is a sequence $i_2 i_3 \ldots i_m$ of indices from $\{1, 2, \ldots, r\}$ such that $v_1 v_{i_2} v_{i_3} \cdots v_{i_m} = w_1 w_{i_2} w_{i_3} \cdots w_{i_m}$. It is proved in [6] that checking whether an instance of MPCP admits a solution is undecidable. A careful examination of the proof in [6] shows that MPCP is undecidable even under following assumptions:

1. For each word u in the list $\{(v_1, w_1), (v_2, w_2) \ldots, (v_r, w_r)\}$, $1 \leq |u| \leq 4$.
2. w_1 is a strict prefix of v_1 and is shorter by at least 2 letters.
3. If the instance has a solution then it has a solution of the form $i_2 i_3 \ldots i_m$ such that $w_1 w_{i_2} \ldots w_{i_k}$ is a strict prefix of $v_1 v_{i_2} \ldots v_{i_k}$ for each $k < m$.

Theorem 8. *For $3 \leq k \leq n$, $(k-1)$-testability is undecidable for regular 1-bounded MSG-definable languages over n processes.*

Proof. Let $\Delta = \{(v_1, w_1), (v_2, w_2), \ldots, (v_r, w_r)\}$ be an instance of MPCP satisfying the assumptions described above. For each pair (v_ℓ, w_ℓ), we construct k MSCs M_{v_ℓ}, M_{w_ℓ} and $\{M^j_{v_\ell, w_\ell} \mid 1 < j < k\}$ over processes $\{1, 2, \ldots, n\}$, such that only processes $\{1, 2, \ldots, k\}$ are active in these k MSCs. The message alphabet for these MSCs is the alphabet of the MPCP instance along with the integers $\{1, 2, \ldots, r\}$. In the definition below, v^j_ℓ and w^j_ℓ are the j^{th} symbols in the strings v_ℓ and w_ℓ, respectively. Also, $i \xrightarrow{m} j$ denotes the MSC generated by the sequence $i!j(m)\, j?i(m)$ where i sends message m to j. For $m \in \mathcal{M}$ and $i < j$ we define

$$N^{i,j}_m = (i \xrightarrow{m} i+1) \cdots (j-1 \xrightarrow{m} j)(j \xrightarrow{m} j-1) \cdots (i+1 \xrightarrow{m} i).$$

In this MSC, the message m is sent from i to j through the intermediate processes $i+1, \ldots, j-1$ and an acknowledgment is sent back from j to i through the same route. We also let $N_\ell = (k \xrightarrow{\ell} 1)$ and define for $1 < j < k$ the MSCs

$$\begin{aligned}
M_{v_\ell} &= N_\ell N^{1,k}_{v^1_\ell} \cdots N^{1,k}_{v^{|v_\ell|}_\ell} \\
M_{w_\ell} &= N_\ell N^{1,k}_{w^1_\ell} \cdots N^{1,k}_{w^{|w_\ell|}_\ell} \\
M^j_{v_\ell,w_\ell} &= N_\ell N^{1,j}_{v^1_\ell} N^{j,k}_{w^1_\ell} \cdots N^{1,j}_{v^{|w_\ell|}_\ell} N^{j,k}_{w^{|w_\ell|}_\ell} N^{1,j}_{v^{|w_\ell|+1}_\ell} \cdots N^{1,j}_{v^{|v_\ell|}_\ell} \qquad \text{if } |w_\ell| \le |v_\ell| \\
M^j_{v_\ell,w_\ell} &= N_\ell N^{1,j}_{v^1_\ell} N^{j,k}_{w^1_\ell} \cdots N^{1,j}_{v^{|v_\ell|}_\ell} N^{j,k}_{w^{|v_\ell|}_\ell} N^{j,k}_{w^{|v_\ell|+1}_\ell} \cdots N^{j,k}_{w^{|w_\ell|}_\ell} \qquad \text{otherwise.}
\end{aligned}$$

Since each word in the MPCP instance is nonempty, each of these MSCs is comconnected, so any MSG whose node labels are drawn from this set of MSCs is guaranteed to be locally-synchronized. For $1 < j < k$, we define

$$\begin{aligned}
L_v &= M_{v_1}\{M_{v_\ell} \mid 1 \le \ell \le r\}^* \\
L_w &= M_{w_1}\{M_{w_\ell} \mid 1 \le \ell \le r\}^* \\
L^j_{v,w} &= M^j_{v_1,w_1}\{M^j_{v_\ell,w_\ell} \mid 1 \le \ell \le r\}^* \\
L_\Delta &= L_v \cup L_w \cup \bigcup_{1<j<n} L^j_{v,w}.
\end{aligned}$$

Claim. Δ has a solution iff L_Δ is not $(k-1)$-testable.

Let $i_2, i_3, \ldots\ldots, i_m$ be a solution of Δ that satisfies Condition 3 listed above. Let $v_1 v_{i_2} \ldots v_{i_m} = w_1 w_{i_2} \ldots w_{i_m} = a_1 a_2 \ldots a_\ell$. Then, we first construct the MSC $M' = N^{1,k}_{a_1} N^{1,k}_{a_2} \cdots N^{1,k}_{a_\ell}$. In M', we insert events labelled $k!1(1)$, $k!1(i_2)$,…,$k!1(i_m)$ into k so as to partition its communications with $k-1$ as w_1,w_{i_2}, …, w_{i_m}. Finally, we insert events labelled $1?k(1)$, $1?k(i_2)$, …, $1?k(i_m)$ into 1 to partition its communications with 2 as v_1, v_{i_2}, …, v_{i_m}. Call this MSC M. To observe that M is indeed a valid MSC, we note that for each $j < m$, $w_1 w_{i_2} \ldots w_{i_j}$ is a prefix of $v_1 v_{i_2} \ldots v_{i_j}$, so the receive event $1?k(i_j)$ inserted into 1 can occur later than the corresponding send event $k!1(i_j)$ inserted into n.

It is easy to verify that $M\restriction_{\{1,2,\ldots,k-1\}} = (M_{v_1} M_{v_{i_2}} \cdots M_{v_{i_m}})\restriction_{\{1,2,\ldots,k-1\}}$. Similarly, $M\restriction_{\{2,3,\ldots,k\}} = (M_{w_1} M_{w_{i_2}} \cdots M_{w_{i_m}})\restriction_{\{2,3,\ldots,k\}}$. Finally, for $1 < j < k$ we have $M\restriction_{\{1,\ldots,j-1,j+1,\ldots,k\}} = (M^j_{v_1 w_1} M^j_{v_{i_2} w_{i_2}} \cdots M^j_{v_{i_m} w_{i_m}})\restriction_{\{1,\ldots,j-1,j+1,\ldots,k\}}$. Thus M is $(k-1)$-implied by L_Δ.

To see that M is not already in L_Δ, simply observe that there is at least one event in M between the second $k!1$ event and the second $1?k$ event and this is not the case for any MSC in L.

Conversely, suppose there is an MSC $M \notin L_\Delta$ that is $(k-1)$-implied by L_Δ. The MSC M must be of one of the following two types:

Type 1 $M\restriction_{\{j\}} \notin (\{N^{1,k}_m \mid m \in \mathcal{M}\}^*)\restriction_{\{j\}}$ for some $1 < j < k$.
Type 2 $M\restriction_{\{j\}} \in (\{N^{1,k}_m \mid m \in \mathcal{M}\}^*)\restriction_{\{j\}}$ for all $1 < j < k$.

If M is of type 1 as witnessed by j, it must be the case that $M{\restriction}_{\{1,2,\ldots,k-1\}} = (M^j_{v_1w_1}M^j_{v_{i_2}w_{i_2}}\cdots M^j_{v_{i_m}w_{i_m}}){\restriction}_{\{1,2,\ldots,k-1\}}$. Similarly, we also have $M{\restriction}_{\{2,3,\ldots,k\}} = (M^j_{v_1w_1}M^j_{v_{i_2}w_{i_2}}\cdots M^j_{v_{i_m}w_{i_m}}){\restriction}_{\{2,3,\ldots,k\}}$. Hence, $M = M^j_{v_1w_1}M^j_{v_{i_2}w_{i_2}}\cdots M^j_{v_{i_m}w_{i_m}}$, which in turn implies that $M \in L_\Delta$ thus contradicting our initial assumption. Therefore M cannot be of type 1.

On the other hand, if M is of type 2, we show that if M is 1-implied by L_Δ then either $M \in L_v \cup L_w$ or Δ has a solution. Note that this is a stronger result since we only assume that M is 1-implied instead of $(k-1)$-implied.

We have $(L^j_{v,w}){\restriction}_1 = (L_v){\restriction}_1$ and $(L^j_{v,w}){\restriction}_k = (L_w){\restriction}_k$. Hence, $M{\restriction}_1 \in (L_v \cup L_w){\restriction}_1$ and $M{\restriction}_k \in (L_v \cup L_w){\restriction}_k$. Using in addition the fact that M is of type 2, we deduce that if we remove from M the messages from k to 1 we obtain an MSC $M' = N^{1,k}_{a_1}N^{1,k}_{a_2}\cdots N^{1,k}_{a_\ell}$ for some word $a_1a_2\cdots a_\ell$.

Now, if $M \notin L_v \cup L_w$ then we must have $M{\restriction}_1 \in (L_v){\restriction}_1$ and $M{\restriction}_k \in (L_w){\restriction}_k$ (otherwise the second message from k to 1 would induce a cycle in the MSC). Therefore, the sequence of messages from k to 1 parses on the left the sequence $a_1a_2\cdots a_\ell$ into some $v_1v_{i_2}\cdots v_{i_m}$ and on the right the same sequence into $w_1w_{i_2}\cdots w_{i_m}$ and Δ has a solution. □

Remark 9. We can modify the proof to obtain the undecidability of 1-testability even for regular 1-bounded MSG-definable languages over $n \geq 3$ processes. Below, we get down to 2 processes but the regular language is only 4-bounded.

Undecidability of 1-testability for 2 processes

The argument in [3] shows that 1-testability is undecidable for regular MSG-definable languages with four processes. We tighten this result to show that 1-testability is undecidable for regular MSG-definable languages over 2 processes.

Theorem 10. *For $n \geq 2$, 1-testability is undecidable for regular 4-bounded MSG-definable languages over n processes.*

Proof. As before, let $\Delta = \{(v_1, w_1), (v_2, w_2), \ldots, (v_r, w_r)\}$ be an instance of MPCP satisfying the assumptions (1–3) stated earlier. With each word $v_i = a_1a_2\ldots a_k$ we associate an MSC M_{v_i} as indicated in Fig. 5.

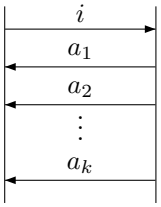

Fig. 5.

Similarly we construct the MSCs M_{w_i}. First, observe that each of these MSCs is com-connected, so any MSG that uses these MSCs as node labels is locally synchronized. Also, from assumption 1 of the MPCP instance, the MSCs are 4-bounded and therefore, any language generated by these MSCs is 4-bounded.

Let $L_v = \{M_{v_i} \mid 1 \leq i \leq r\}$ and $L_w = \{M_{w_i} \mid 1 \leq i \leq r\}$. Consider the MSG-definable regular language $L_\Delta = M_{v_1}.(L_v)^* + M_{w_1}.(L_w)^*$.

If M is any MSC in L then $M{\restriction}_1$ is a word of the form $1!2(1)\ 1?2(x_1)\ 1!2(i_2)\ 1?2(x_{i_2})\ \cdots\ 1!2(i_k)\ 1?2(x_{i_k})$ where either each x_{i_j} is v_{i_j} or each x_{i_j} is w_{i_j}. A similar property holds for $M{\restriction}_2$.

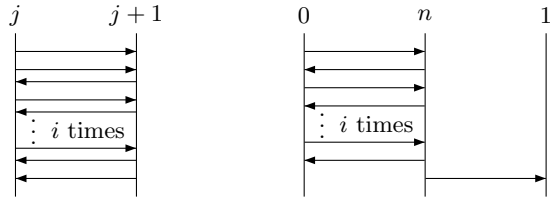

Fig. 6.

Suppose there is a 1-implied MSC M that is not in L_Δ. Then, $M\restriction_1 = (M_{w_1} M_{w_{i_2}} \cdots M_{w_{i_k}})\restriction_1$ and $M\restriction_2 = (M_{v_1} M_{v_{i_2}} \cdots M_{v_{i_k}})\restriction_2$. It follows that the MPCP instance Δ has a solution.

Conversely, from any solution $i_2 i_3 \ldots i_k$ to the MPCP instance Δ, it is quite easy to construct a 1-implied scenario where p_1 witnesses the $w_1 w_{i_2} \cdots w_{i_k}$ and p_2 witnesses the $v_1 v_{i_2} \cdots v_{i_k}$. □

Finally, we turn our attention to MSGs over a singleton message alphabet. As we shall see in the next section, 1-testability is decidable for locally-synchronized MSG languages over singleton message alphabets. However, k-testability is undecidable for any $k > 1$.

Theorem 11. *Let n and k be any two integers with $n > 2$ and $1 < k < n - 1$. There is a constant B such that the problem of deciding whether a B-bounded HMSC language over a singleton alphabet is k-testable is undecidable.*

Proof. (Sketch) Following the proof of Theorem 10, it suffices to prove the result for $k = n-2$. We modify the reduction used in the proof of Theorem 8 to use a singleton message alphabet. Let us assume that the message alphabet is $\{a_1, a_2, \ldots a_k\}$. A communication a_i between process j and $j + 1$ is replaced by the communication pattern at the left of Fig. 6. Since $k > 1$, any subset containing j and $j + 1$ would witness that the communication between j and $j + 1$ is uniquely and correctly parsed. We still have to deal with the message from process n to 1. We add an additional process 0 and simulate the act of sending i from n to 1 by the MSC at the right of Fig. 6. Since $k > 1$, the pair $\{n, 0\}$ will jointly witness that i is sent from n to 1. □

5 Decidability

In this section we consider the 1-testability problem for regular MSC languages where the message alphabet for each channel is a singleton. In this case, we may omit the message content in any event. Throughout this section, we write $p!q$ and $q?p$ rather than $p!q(m)$ and $q?p(m)$.

Proper and complete words For a word w and a letter a, $\#_a(w)$ denotes the number of times a appears in w. We say that $\sigma \in \Sigma^*$ is *proper* if for every prefix τ of σ and every pair p, q of processes, $\#_{p!q}(\tau) \geq \#_{q?p}(\tau)$. We say that σ is *complete* if σ is proper and $\#_{p!q}(\sigma) = \#_{q?p}(\sigma)$ for every pair p, q of processes.

Every linearization of any MSC is a complete word and every complete word is the linearization of a unique MSC.

Suppose L is the set of linearizations of a MSC language. Let $L_p = \{w\restriction_p \mid w \in L\}$. Let, *1-closure*$(L) = \{w \mid w$ is complete and $\forall p.\ w\restriction_p \in L_p\}$. Observe that *1-closure*(L) is the set of 1-implied words of L.

Let L be the set of linearizations of some regular MSC language over a singleton message alphabet. From any finite automaton $A = (Q, \Sigma, \delta, i, F)$ accepting L we can easily construct for each $p \in \mathcal{P}$ an automaton $A_p = (Q_p, \Sigma\restriction_p, \delta_p, i_p, F_p)$ that accepts L_p. Note that *1-closure*(L) is exactly the set of complete words accepted by the (free) product $\prod_p A_p$ of these automata. The product automaton accepts a regular language. The difficulty is in ensuring that a word that is accepted is complete. However, since the message alphabet is a singleton, it suffices to keep track of the number of sent and as yet unreceived messages along any channel. This leads us naturally to the following idea.

From these automata $(A_p)_{p\in\mathcal{P}}$, we construct a labelled Petri net N whose firing sequences are related to words in *1-closure*(L).[3]

1. The set of places is $\bigcup_{p\in\mathcal{P}} Q_p \cup \{c_{pq} \mid p, q \in \mathcal{P}\}$.
2. The set of transtions is $\bigcup_{p\in\mathcal{P}} \delta_p$.
3. The transition $(s, p!q, t) \in \delta_p$ removes a token from the place s and deposits a token each at the places t and c_{pq}.
4. The transition $(s, q?p, t) \in \delta_p$ removes a token each from the places s and c_{qp} and deposits a token at t.
5. The initial marking has one token in each place i_p, $p \in \mathcal{P}$, corresponding to the initial states of the automata A_p.
6. The label on the transition (s, x, t) is $x \in \Sigma$.

From the definition of N it follows that in any reachable marking, for any $p \in \mathcal{P}$, exactly one place in Q_p has a token. We say that a marking of this net is *final* if every place of the form c_{pq} is empty and for each $p \in \mathcal{P}$ there is $f_p \in F_p$ such that f_p is marked. There are only finitely many final markings.

It is quite easy to observe that a word $w \in$ *1-closure*(L) if and only if there is a firing sequence labelled w from the initial marking to some final marking. This leads us naturally to the following proposition:

Proposition 12. *Let B be any integer. We can decide if* 1-closure(L) *contains a word that is not B-bounded.*

Proof. The set of markings where exactly one of the places of the form c_{pq} has $B + 1$ tokens (and all other places have at most B tokens) is finite. Since reachability is decidable for Petri nets [8, 9], we can check for each such marking χ whether χ is reachable from the initial marking and if some final marking is reachable from χ. □

[3] Due to lack of space, we are constrained to omit basic definitions concerning Petri nets. See [14] for a detailed introduction.

Now, if the given MSC language $\mathcal{L}$ is regular we can compute a bound B from its presentation such that $\mathcal{L}$ is B-bounded. Using the proposition above, we can check if *1-closure*(L) contains words that are not B-bounded. If the answer is yes, then L is not 1-testable. On the other hand, if there are no words in *1-closure*(L) that violate the B bound on any channel, we can look for 1-implied scenarios using the following proposition.

Proposition 13. *Let L be a B-bounded MSC regular language. We can decide if* 1-closure(L) *contains any B-bounded words not in L.*

Proof (Sketch). Construct the net N corresponding to the product automaton $\prod_p A_p$ as described earlier. Explore all reachable configurations in which each place in $\{c_{pq} \mid p, q \in \mathcal{P}\}$ has no more than B tokens. This results in a finite automaton that accepts all the B-bounded words in *1-closure*(L). □

From the two propositions described above, we conclude that:

Theorem 14. *The 1-testability problem for regular MSC languages over a singleton message alphabet is decidable.*[4]

In fact, in this case we can even decide if *1-closure*(L) is regular.

Theorem 15. *Let L be a regular MSC language over a singleton message alphabet. Then, it is decidable whether* 1-closure(L) *is regular.*

Proof. We reduce this to the Intermediate Marking Problem (IMP) for Petri nets, which is known to be decidable [17].

Consider the Petri net constructed above. Define an *intermediate marking* to be one that can be reached from the initial configuration and from which some final marking is reachable. If the number of intermediate markings is finite, there is a bound B such that along any firing sequence from the initial marking to a final marking, no place ever contains more than B tokens. In other words, if w is the word generated by some firing sequence from the initial to a final configuration then the number of unreceived messages at any prefix of w is bounded by B. Thus, the language *1-closure*(L) is the language of a bounded Petri net and hence regular.

On the other hand, if the number of intermediate markings is infinite, we may conclude that for any B there is a word $w \in$ *1-closure*(L) which has a prefix with B sent and as yet unreceived messages. Thus *1-closure*(L) is not B-bounded for any B and hence not regular. □

6 Discussion

We have seen in this paper that developing a framework for locally testing MSC based specifications is hard. This is because MSG-based specifications permit unintended implied scenarios that cannot, in general, be detected algorithmically.

[4] This theorem can also be viewed as a special case of the result proved in [11] that 1-testability is decidable for MSCs without fifo channels, but our proof for this special case is simpler than the general proof in [11].

There are two approaches to attack the problem of local testing in light of this bottleneck. One is to characterize structural conditions for k-testability. This is analogous to identifying locally synchronized MSGs as those that generate regular MSC specifications, even though the general problem of checking whether an MSG specification describes a regular MSC language is undecidable [5].

Another tactic would be to recognize that practical implementations always work with bounded buffers and impose an upper bound B on the buffer size. The set of B-bounded MSCs in the k-closure of a regular MSC language is again regular, so the B-bounded k-testability problem is decidable for all regular MSG-definable languages. The focus could now be on efficiently identifying the smallest k for which an MSG specification is k-testable. Another interesting problem is to identify a minimal set of tests to validate a k-testable specification.

References

1. Alur, R., Yannakakis, M.: Model checking of message sequence charts. *Proc. CONCUR 1999)*, Springer LNCS **1664** (1999) 114–129.
2. Alur, R., Etessami, K., Yannakakis, M.: Inference of message sequence graphs. *IEEE Trans. Software Engg* **29(7)** (2003) 623–633.
3. Alur, R., Etessami, K., Yannakakis, M.: Realizability and Verification of MSC Graphs. *Theor. Comput. Sci.* **331(1)** (2005) 97–114.
4. Booch, G., Jacobson, I., Rumbaugh, J.: *Unified Modeling Language User Guide.* Addison-Wesley (1997).
5. Henriksen, J.G., Mukund, M., Narayan Kumar, K., Sohoni, M., and Thiagarajan, P.S.: A Theory of Regular MSC Languages. *Inf. Comp.*, **202(1)** (2005) 1–38.
6. Hopcroft, J.E., and Ullman, J.D.: *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley (1979).
7. ITU-TS Recommendation Z.120: *Message Sequence Chart (MSC).* ITU-TS, Geneva (1997).
8. Kosaraju, S.R.: Decidability of Reachability in Vector Addition Systems. *Proc 14th ACM STOC*, (1982) 267–281.
9. Mayr, E.W.: An Algorithm for the General Petri Net Reachability Problem. *SIAM J. Comput*, **13(3)** (1984) 441–460.
10. Mauw, S., Reniers, M. A.: High-level message sequence charts, *Proc SDL'97*, Elsevier (1997) 291–306.
11. Morin, R.: Recognizable Sets of Message Sequence Charts. *Proc. STACS 2002*, Springer LNCS **2285** (2002) 523–534.
12. Muscholl, A., Peled, D.: Message sequence graphs and decision problems on Mazurkiewicz traces. *Proc. MFCS 1999*, Springer LNCS **1672** (1999) 81–91.
13. A. Muscholl and H. Peterson: A note on the commutative closure of star-free languages. *Information Processing Letters*, **57(2)** (1996) 71–74.
14. Reisig, W., and Rozenberg, G. (Eds.): *Lectures on Petri Nets I: Basic Models*, Advances in Petri Nets, Springer LNCS **1491** (1998).
15. Rudolph, E., Graubmann, P., Grabowski, J.: Tutorial on message sequence charts. In *Computer Networks and ISDN Systems — SDL and MSC* **28** (1996).
16. Thiagarajan, P. S.: A Trace Consistent Subset of PTL. *Proc. CONCUR '95*, Springer LNCS **962** (1995) 438–452.
17. Wimmel, H.: Infinity of Intermediate States Is Decidable for Petri Nets. *Proc. ICATPN 2004*, Springer LNCS **3099** (2004) 426–434.