FSTTCS 2006, 26th International Conference on Foundations of Software Technology and Theoretical Computer Science
Proceedings: R Ramanujam, Sandeep Sen (eds.)
Springer Lecture Notes in Computer Science **3821** (2005), 335–347

Causal closure for MSC languages

Bharat Adsul, Madhavan Mukund, K. Narayan Kumar, and Vasumathi Narayanan

Chennai Mathematical Institute, Chennai, India
{abharat,madhavan,kumar,vasumathi}@cmi.ac.in

Abstract. Message sequence charts (MSCs) are commonly used to specify interactions between agents in communicating systems. Their visual nature makes them attractive for describing scenarios, but also leads to ambiguities that can result in incomplete or inconsistent descriptions.

One such problem is that of implied scenarios—a set of MSCs may imply new MSCs which are "locally consistent" with the given set. If local consistency is defined in terms of local projections of actions along each process, it is undecidable whether a set of MSCs is closed with respect to implied scenarios, even for regular MSC languages [3].

We introduce a new and natural notion of local consistency called *causal closure*, based on the causal view of a process—all the information it collects, directly or indirectly, through its actions. Our main result is that checking whether a set of MSCs is closed with respect to implied scenarios modulo causal closure is decidable for regular MSC languages.

1 Introduction

Message Sequence Charts (MSCs) [10] are an appealing visual formalism that are used in a number of software engineering notational frameworks such as SDL [15] and UML [4, 8]. 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) [12]. 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.

Though the visual nature of MSGs makes them attractive for describing scenarios, it also leads to ambiguities that can result in incomplete or inconsistent descriptions. An important issue is the presence of implied scenarios [2, 3]. An MSC M is (weakly) implied by an MSC language $\mathcal{L}$ if the local actions of each process p along M agree with its local actions along some good MSC $M_p \in \mathcal{L}$.

Implied scenarios are naturally tied to the question of *realizability*—when is an MSG specification implementable as a set of communicating finite-state machines? In a distributed model with local acceptance conditions, it is natural to expect the specification to be closed with respect to local projections. Thus, a language is said to be weakly realizable if all weakly implied scenarios are included in the language. Unfortunately, weak realizability is undecidable, even for regular MSC languages [3].

Weak implication presumes that the only information a process can maintain locally about an MSC is the sequence of actions that it participates in. However, we can augment the underlying message alphabet of an MSC by tagging auxiliary information to each message. Using this extra information, processes can maintain a bounded amount of information about the global state of the system [13]. With this, we arrive at a stronger notion of implied scenario that we call causal closure, based on the local view that each process has of an MSC from the information it receives, directly or indirectly, about the system.

Our main result is that causal closure preserves regularity for MSC languages, in contrast to the situation with weak closure. From this it follows that causal realizability is effectively checkable for regular MSC languages, both in the case of implementations with deadlocks and for *safe*, or deadlock-free, implementations.

Our result also allows us to interpret MSGs as incomplete specifications whose semantics is given in terms of the causal closure. Thus, we can retain relatively simple visual specifications without compromising on the completeness and consistency of verification.

The paper is organized as follows. We begin with some basic definitions regarding MSCs, message sequence graphs and message-passing automata. In the next section, we recall the results for weakly implied scenarios. In Section 4, we define the notion of causal closure and establish our main result, that causal closure preserves regularity for MSC languages. Finally, in Section 5, we examine the feasibility of using causal closure as a semantics for MSGs.

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 $\Sigma_{\mathcal{P}} = \bigcup_{p \in \mathcal{P}} \Sigma_p$. We also denote the set of *channels* by $Ch = \{(p, q) \mid p \neq q\}$. Whenever the set of processes $\mathcal{P}$ is clear from the context, we write Σ instead of $\Sigma_{\mathcal{P}}$, etc.

Labelled posets A Σ-labelled poset is a structure $M = (E, \leq, \lambda)$ where $(E, \leq)$ is a poset and $\lambda : E \to \Sigma$ is a labelling function. For $e \in E$, let $\downarrow e = \{e' \mid e' \leq e\}$. For $X \subseteq E$, $\downarrow X = \cup_{e \in X} \downarrow e$. We call $X \subseteq E$ a *prefix* of M if $X = \downarrow X$.

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 each $(p, q) \in Ch$, we define the relation $<_{pq}$ as follows:

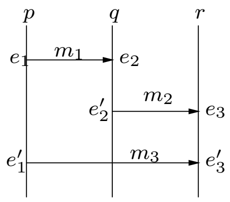

Fig. 1. An MSC over $\{p, q, r\}$.

$$e <_{pq} e' \iff \lambda(e) = p!q(m),\ \lambda(e') = q?p(m) \text{ and } |{\downarrow}e \cap E_{p!q(m)}| = |{\downarrow}e' \cap E_{q?p(m)}|$$

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 $M = (E, \leq, \lambda)$ that satisfies the following conditions.*

1. *Each relation $\leq_{pp}$ is a linear 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 the relation $\bigcup_{p,q \in \mathcal{P}} <_{pq}$.*

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.

For an MSC $M = (E, \leq, \lambda)$, we let $\text{lin}(M) = \{\lambda(\pi) \mid \pi$ 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 L over Σ consisting of all linearizations of the MSCs in $\mathcal{L}$. For an MSC language $\mathcal{L}$, we set $\text{lin}(\mathcal{L}) = \bigcup\{\text{lin}(M) \mid M \in \mathcal{L}\}$.

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

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.

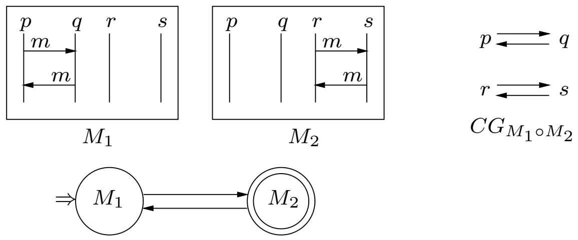

Fig. 2. A message sequence graph

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 $\leq\, = (\bigcup_{p,q \in \mathcal{P}} <_{pq})^*$, where $<_{pp}=<^1_{pp} \cup <^2_{pp} \cup \{(e_1, e_2) \mid e_1 \in E_1, e_2 \in E_2, \lambda(e_1) \in \Sigma_p, \lambda(e_2) \in \Sigma_p\}$ and for $(p, q) \in Ch$, $<_{pq}=<^1_{pq} \cup <^2_{pq}$.

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 $L(\mathcal{G}) = \{M(\pi) \mid \pi \text{ is a run through } \mathcal{G}\}$.

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 $|\sigma\lceil_{p!q(m)}| = |\sigma\lceil_{r!s(m)}| \geq 1$, which is not a regular string language.

In general, it is undecidable whether an MSG describes a regular MSC language [9]. 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* [14] (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 3. *If $\mathcal{G}$ is locally synchronized, $L(\mathcal{G})$ is a regular MSC language.*

2.3 Message-passing automata

Message-passing automata are natural recognizers for MSC languages.

Definition 4. *A* message-passing automaton (MPA) *over Σ is a structure $\mathcal{A} = (\{\mathcal{A}_p\}_{p\in\mathcal{P}}, \Delta, s_{in}, F)$ where:*

- *Δ is a finite alphabet of* auxiliary messages.
- *Each component $\mathcal{A}_p$ is of the form $(S_p, \rightarrow_p)$ where S_p is a finite set of p-local states and $\rightarrow_p \subseteq S_p \times \Sigma_p \times \Delta \times S_p$ is the p-local transition relation.*
- *$s_{in} \in \prod_{p\in\mathcal{P}} S_p$ is the global initial state.*
- *$F \subseteq \prod_{p\in\mathcal{P}} S_p$ is the set of global final states.*

The local transition relation $\rightarrow_p$ specifies how the process p sends and receives messages. The transition $(s, p!q(m), x, s')$ says that in state s, p can send the message m to q tagged with auxiliary information x and move to state s'. Similarly, the transition $(s, p?q(m), x, s')$ signifies that at state s, p can receive the message m from q tagged with information x and move to state s'.

A global state of $\mathcal{A}$ is an element of $\prod_{p\in\mathcal{P}} S_p$. For a global state s, s_p denotes the pth component of s. A *configuration* is a pair (s, χ) where s is a global state and $\chi : Ch \rightarrow (\mathcal{M} \times \Delta)^*$ is the *channel state* describing the message queue in each channel c. The *initial configuration* of $\mathcal{A}$ is $(s_{in}, \chi_\varepsilon)$ where $\chi_\varepsilon(c)$ is the empty string ε for every channel c. The set of *final configurations* of $\mathcal{A}$ is $F \times \{\chi_\varepsilon\}$.

The set of reachable configurations of $\mathcal{A}$, $Conf_{\mathcal{A}}$, is defined in the obvious way. The initial configuration $(s_{in}, \chi_\varepsilon)$ is in $Conf_{\mathcal{A}}$. If $(s, \chi) \in Conf_{\mathcal{A}}$ and $(s_p, p!q(m), x, s'_p) \in \rightarrow_p$, then there is a global move $(s, \chi) \overset{p!q(m)}{\Longrightarrow} (s', \chi')$ where for $r \neq p$, $s_r = s'_r$, for each $r \in \mathcal{P}$, $\chi'((p, q)) = \chi((p, q)) \cdot (m, x)$, and for $c \neq (p, q)$, $\chi'(c) = \chi(c)$. Similarly, if $(s, \chi) \in Conf_{\mathcal{A}}$ and $(s_p, p?q(m), x, s'_p) \in \rightarrow_p$, then there is a global move $(s, \chi) \overset{p?q(m)}{\Longrightarrow} (s', \chi')$ where for $r \neq p$, $s_r = s'_r$, for each $r \in \mathcal{P}$, $\chi((q, p)) = (m, x) \cdot \chi'((q, p))$, and for $c \neq (q, p)$, $\chi'(c) = \chi(c)$.

Let $\mathrm{prf}(\sigma)$ denote the set of prefixes of a word $\sigma \in \Sigma^*$. A run of $\mathcal{A}$ over σ is a map $\rho : \mathrm{prf}(\sigma) \rightarrow Conf_{\mathcal{A}}$ such that $\rho(\varepsilon) = (s_{in}, \chi_\varepsilon)$ and for each $\tau a \in \mathrm{prf}(\sigma)$, $\rho(\tau) \overset{a}{\Longrightarrow} \rho(\tau a)$. The run ρ is *accepting* if $\rho(\sigma)$ is a final configuration.

We define $L(\mathcal{A}) = \{\sigma \mid \mathcal{A}$ has an accepting run over $\sigma\}$. $L(\mathcal{A})$ corresponds to the set of linearizations of an MSC language. To simplify notation, we write $L(\mathcal{A}) = \mathcal{L}$, where $\mathcal{L}$ is an MSC language, rather than $L(\mathcal{A}) = \mathrm{lin}(\mathcal{L})$.

For $B \in \mathbb{N}$, we say that a configuration (s, χ) of $\mathcal{A}$ is *B-bounded* if $|\chi(c)| \leq B$ for every channel $c \in Ch$. We say that $\mathcal{A}$ is a B-bounded automaton if every reachable configuration $(s, \chi) \in Conf_{\mathcal{A}}$ is B-bounded.

The MPA $\mathcal{A}$ in Fig. 3 has two components, p and q, with initial state (s_1, t_1) and only one final state, (s_2, t_3). A typical MSC in $L(\mathcal{A})$ is displayed at the right.

Deterministic message-passing automata We say that $\mathcal{A}$ is *deterministic* if the transition relation $\rightarrow_p$ for each component satisfies the following conditions:

- $(s, p!q(m), x', s') \in \rightarrow_p$ and $(s, p!q(m), x'', s'') \in \rightarrow_p$ imply $x' = x''$, $s' = s''$.

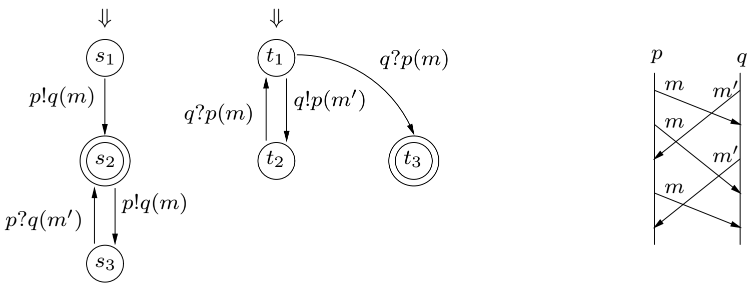

Fig. 3. A message-passing automaton.

– $(s, p?q(m), x, s') \in \rightarrow_p$ and $(s, p?q(m), x, s'') \in \rightarrow_p$ imply $s' = s''$.

For deterministic MPAs, the global state at the end of an MSC is independent of the choice of linearization.

Proposition 5. *Let $\mathcal{A}$ be a deterministic MPA, $M = (E, \leq, \lambda)$ an MSC and $E' \subseteq E$ a prefix of M. Let w and w' be linearizations of E' and let ρ and ρ' be the runs of $\mathcal{A}$ on w and w', respectively. Then, $\rho(w) = \rho(w')$.*

If $\mathcal{A}$ is deterministic, for any prefix E' of an MSC M, we can unambiguously write $\rho(E')$ to denote the unique run of $\mathcal{A}$ on E'. In particular, the unique run of $\mathcal{A}$ on M can be written as $\rho(M)$. The following theorem characterizes regular MSC languages in terms of message-passing automata [9].

Theorem 6. *An MSC language $\mathcal{L}$ is regular iff there is a deterministic B-bounded MPA $\mathcal{A}$ such that $L(\mathcal{A}) = \mathcal{L}$.*

3 Implied scenarios: the weak case

When we use MSC languages to specify sets of scenarios, it is important to identify whether the specification is complete. A natural requirement is that the language be closed with respect to local views—for an MSC M, if every process locally believes that M belongs to the language $\mathcal{L}$, M should in fact be in $\mathcal{L}$.

One way to formalize closure with respect to local views is in terms of local projections [2, 3].

Definition 7.
– *Let $M = (E, \leq, \lambda)$ be an MSC and $p \in \mathcal{P}$ a process. The projection of M onto p, $M\lceil_p$, is the Σ-labelled partial order $(E_p, \leq_p, \lambda_p)$, where $\leq_p = \leq \cap (E_p \times E_p)$ is a total order and $\lambda_p = \lambda\lceil_{E_p}$.*
– *An MSC M is said to be* weakly implied *by $\mathcal{L}$ if for every process $p \in \mathcal{P}$ there is an MSC $M_p \in \mathcal{L}$ such that $M_p\lceil_p = M\lceil_p$.*
– *The* weak closure *of $\mathcal{L}$ is the collection of MSCs*

$$\mathit{WeakCl}(\mathcal{L}) \triangleq \{M \mid M \text{ is weakly implied by } \mathcal{L}\}.$$

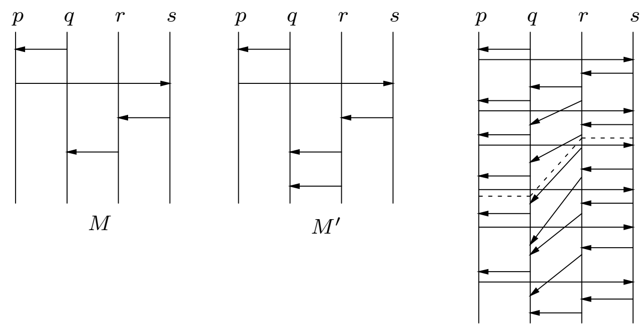

Fig. 4. A regular MSC language whose weak closure has unbounded buffers

Unfortunately, the weak closure of a language can admit unbounded channels even when every channel in the original language is uniformly bounded. An example is shown in Fig. 4—all messages are labelled m and labels are omitted.

Both M and M' are com-connected, so the MSC language consisting of arbitrary concatenations of M and M' is regular. However, for every natural number k, the weak closure of this language contains the MSC M_k in which the actions of p and q correspond to the sequence $M^{2k} \circ M'^k$ while the actions of r and s match the sequence $M'^k \circ M^{2k}$. In M_k, the buffer from p to s contains k messages at the global state where p and q make the transition from M to M' and r and s make the transition from M' to M. The figure shows the case $k = 2$. The dotted line marks the global cut where the channel from p to s has maximum capacity.

Implied scenarios have a close link to implementability, or *realizability*. An MSC language recognized by a communicating finite-state machine with a local acceptance condition must be closed with respect to local views. We say that an MSC language $\mathcal{L}$ is *weakly realizable* if $\mathcal{L} = \mathit{WeakCl}(\mathcal{L})$. We have the following negative result [3], which arises from the fact that the weak closure of a regular MSC language may have unbounded buffers.

Theorem 8. *Let $\mathcal{G}$ be a locally synchronized MSG. It is undecidable if $L(\mathcal{G})$ is weakly realizable.*

To overcome this negative result, a more restrictive notion of realizability is proposed in [2, 11]. An MSC language is said to be *safely realizable* if it admits a deadlock-free implementation—a *deadlock* is a global state from which no accepting state is reachable. In a safe implementation, it turns out that all implied scenarios must have bounded buffers, yielding the following result [3].

Theorem 9. *Let $\mathcal{G}$ be a locally synchronized MSG. It is decidable if $L(\mathcal{G})$ is safely realizable.*

4 Causal closure for regular MSC languages

Weak closure assumes that the only information that a process can maintain locally about the current MSC is the sequence of actions that it participates

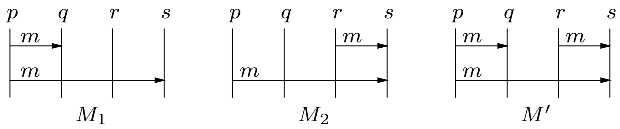

Fig. 5. Illustrating the difference between weak and causal closure

in. However, as we have observed when characterizing regular MSC languages in terms of MPAs, we can tag each underlying message with extra data using which processes can maintain a bounded amount of information about the global state of the system. This leads us to a stronger notion of local view called causal view.

We begin by defining p-views. For an MSC $M = (E, \leq, \lambda)$ and $p \in \mathcal{P}$, let $\max_p(M)$ denote the maximum event from E_p in M—since all p events in M are linearly ordered by $\leq_{pp}$, $\max_p(M)$ is well-defined whenever $E_p \neq \emptyset$.

Definition 10. *Let $M = (E, \leq, \lambda)$ be an MSC and $p \in \mathcal{P}$ a process. The p-view of M, $\partial_p(M)$, is the Σ-labelled partial order $(E', \leq', \lambda')$ where $E' = \{e \mid e \leq \max_p(M)\}$, $\leq' = \leq \cap (E' \times E')$ and $\lambda' = \lambda\restriction_{E'}$. If $E_p = \emptyset$, $\partial_p(M) = (\emptyset, \emptyset, \emptyset)$.*

It is easy to observe that $\partial_p(M)$ is always a prefix of M. Causal realizability captures the intuition that each process p can keep track of the events in $\partial_p(M)$.

Definition 11. *Let $\mathcal{L}$ be an MSC language.*

– *An MSC M is said to be* causally implied *by $\mathcal{L}$ if for every process $p \in \mathcal{P}$ there is an MSC $M_p \in \mathcal{L}$ such that $\partial_p(M) = \partial_p(M_p)$.*
– *The* causal closure *of $\mathcal{L}$ is the collection of MSCs*
$$CausalCl(\mathcal{L}) \stackrel{\triangle}{=} \{M \mid M \text{ is causally implied by } \mathcal{L}\}.$$
– *The language $\mathcal{L}$ is said to be* causally realizable *if $\mathcal{L} = CausalCl(\mathcal{L})$.*

An MSC language is causally realizable if each local process can recognize whether an MSC belongs to the language based purely on its causal view of the MSC. Observe that we always have $\mathcal{L} \subseteq CausalCl(\mathcal{L})$. Thus, $\mathcal{L}$ is not causally realizable iff there is an MSC $M \in CausalCl(\mathcal{L})$ such that $M \notin \mathcal{L}$.

We also have the inclusion $CausalCl(\mathcal{L}) \subseteq WeakCl(\mathcal{L})$. In general, $CausalCl(\mathcal{L}) \neq WeakCl(\mathcal{L})$—for instance, the implied MSCs in Fig. 4 are not in the causal closure of the language. Fig. 5 illustrates the difference between weak and causal closure. Here M' is in the weak closure of $\{M_1, M_2\}$ but not in the causal closure, because the causal view of process s includes information about whether or not p has sent a message to q.

Every regular MSC language $\mathcal{L}$ is recognized by a deterministic B-bounded MPA $\mathcal{A}_\mathcal{L}$. To construct an MPA for $CausalCl(\mathcal{L})$, we simulate $\mathcal{A}_\mathcal{L}$ and make each process go into a local accepting state if its current history is consistent with some accepting run of $\mathcal{A}_\mathcal{L}$. To achieve this, we make use of a bounded time-stamping protocol for B-bounded MPA, described in [13], by which each process can maintain the latest known state of every other process and channel. Using this protocol, we can derive the following result.

Theorem 12. *Let* $\mathcal{A} = (\{\mathcal{A}_p\}_{p\in\mathcal{P}}, \Delta, s_{in}, F)$ *be a deterministic* B*-bounded MPA. We can augment* $\mathcal{A}$ *with time-stamping information to get a deterministic* B*-bounded MPA* $\mathcal{A}_\tau = (\{\mathcal{A}^\tau_p\}_{p\in\mathcal{P}}, \Delta^\tau, s^\tau_{in}, F^\tau)$ *such that:*

- *For each process* p*, let* $\mathcal{A}_p = (S_p, \rightarrow_p)$ *and* $\mathcal{A}^\tau_p = (S^\tau_p, \rightarrow^\tau_p)$ *be the* p *components of* $\mathcal{A}$ *and* $\mathcal{A}_\tau$*, respectively. Then:*
  - S^τ_p *is of the form* $S_p \times \Gamma$*, where* Γ *contains a bounded amount of time-stamped data about all the processes and channels in the system.*
  - $\rightarrow^\tau_p$ *is such that for every MSC* $M = (E, \leq, \lambda)$ *and for every prefix* E' *of* M*, the unique run* $\rho_\tau(E')$ *of* $\mathcal{A}_\tau$ *on* E' *corresponds to the unique run* $\rho(E')$ *of* $\mathcal{A}$ *on* E' *in the sense that* $\rho(E')$ *matches the first component of* $\rho_\tau(E')$*.*
- $s^\tau_{in} = \prod_{p\in\mathcal{P}}(s_p, \gamma^0_p)$ *where* $s_{in} = \prod_{p\in\mathcal{P}} s_p$ *and* $\prod_{p\in\mathcal{P}} \gamma^0_p$ *is a fixed set of initial time-stamps.*
- $F^\tau = \left\{\prod_{p\in\mathcal{P}}(s_p, \gamma_p) \mid \prod_{p\in\mathcal{P}} s_p \in F\right\}$*.*
- *Let* $M = (E, \leq, \lambda)$ *be an MSC. For each* $p \in \mathcal{P}$*, from the* p*-state* (s_p, γ_p) *assigned by* $\rho_\tau(\partial_p(M))$ *we can recover the configuration* (s, χ) *reached by* $\mathcal{A}$ *at the end of* $\partial_p(M)$*.*

When we augment a deterministic MPA $\mathcal{A}$ with time-stamping data to obtain $\mathcal{A}_\tau$, after any sequence of actions w, the local state of p in $\mathcal{A}_\tau$ allows us to recover the global configuration reached by $\mathcal{A}$ after processing all actions in the p-view of w. Thus, incrementally each process can keep track of the global configuration of the automaton $\mathcal{A}$ for the portion of the MSC that it has seen so far.

Theorem 13. *Let* $\mathcal{L}$ *be a regular MSC language. Then, its causal closure* $CausalCl(\mathcal{L})$ *is also a regular MSC language.*

Proof. Since $\mathcal{L}$ is a regular MSC language, by Theorem 6 there is a deterministic B-bounded MPA $\mathcal{A}$ such that $L(\mathcal{A}) = \mathcal{L}$.

We apply Theorem 12, to obtain a new automaton $\mathcal{A}_\tau$ that augments $\mathcal{A}$ with time-stamped information. For each process p, we define a subset of *local* final states $F^\tau_p \subseteq S^\tau_p$ as follows. A state (s_p, γ_p) belongs to F^τ_p iff, starting from the configuration (s, χ) of $\mathcal{A}$ that we recover from (s_p, γ_p), $\mathcal{A}$ can reach a configuration (f, χ_ε), where $f \in F$, without performing any actions involving process p. Notice that the sets F^τ_p are effectively computable.

In $\mathcal{A}_\tau$, we replace the set of global final states F^τ by the product of local final states $\prod_{p\in\mathcal{P}} F^\tau_p$ and call this modified MPA $\mathcal{A}^{Cl}_\tau$.

Claim $L(\mathcal{A}^{Cl}_\tau) = CausalCl(\mathcal{L})$.

Proof of claim ($\Leftarrow$) Suppose that $M \in CausalCl(\mathcal{L})$. Then, for every process p, there is an MSC $M_p \in \mathcal{L}$ such that $\partial_p(M) = \partial_p(M_p)$. Fix $p \in \mathcal{P}$ and let (s_p, γ_p) be the state of p in the run ρ_τ of $\mathcal{A}_\tau$ on $\partial_p(M)$. The configuration (s, χ) recorded in (s_p, γ_p) is the configuration reached by $\mathcal{A}$ at the end of $\partial_p(M) = \partial_p(M_p)$. Since $M_p \in \mathcal{L}$, $\mathcal{A}$ can reach a configuration (f, χ_ε), $f \in F$, starting from (s, χ), without performing any actions involving p. Thus, $(s_p, \gamma_p) \in F^\tau_p$. Since p does not make any moves outside $\partial_p(M)$, the state of p in $\rho_\tau(M)$ also belongs to F_p^τ. Since every process p reaches a state in F_p^τ at the end of M, $M \in L(\mathcal{A}_\tau^{Cl})$.

($\Rightarrow$) Suppose that $M \in L(\mathcal{A}_\tau^{Cl})$. Fix a process p, and let $(s_p, \gamma_p) \in F_p^\tau$ be the state of p in the accepting run $\rho_\tau(M)$ of $\mathcal{A}_\tau^{Cl}$ on M. Since p does not participate in any action outside $\partial_p(M)$, the state of p in $\rho_\tau(\partial_p(M))$ must also be (s_p, γ_p).

Let (s, χ) be the configuration of $\mathcal{A}$ that we recover from (s_p, γ_p)—by Theorem 12, (s, χ) is the configuration reached by $\mathcal{A}$ at the end of $\partial_p(M)$. From the definition of F_p^τ, we know that $\mathcal{A}$ can reach a configuration (f, χ_ε) from (s, χ), where $f \in F$, without perfoming any actions involving p. Let w_p be linearization of $\partial_p(M)$ and let w be the sequence of actions processed by $\mathcal{A}$ when going from the configuration (s, χ) to the configuration (f, χ_ε). All messages sent during w_p but not received in w_p must be received in w since all channels are empty in the final configuration. It is not difficult to see that $w_p w$ corresponds to the linearization of an MSC M_p. By construction, $\mathcal{A}$ has an accepting run on M_p, so $M_p \in \mathcal{L}$, with $\partial_p(M) = \partial_p(M_p)$.

Thus, whenever p reaches a state in F_p^τ after M, there is an MSC $M_p \in \mathcal{L}$ such that $\partial_p(M) = \partial_p(M_p)$. If M is in $L(\mathcal{A}_\tau)$, then we find such a witness M_p for every $p \in \mathcal{P}$, so $M \in \mathit{CausalCl}(\mathcal{L})$. ☐

Since we can construct a B-bounded MPA recognizing $\mathit{CausalCl}(\mathcal{L})$ for any regular MSC language $\mathcal{L}$, we can effectively check whether $\mathcal{L} = \mathit{CausalCl}(\mathcal{L})$. Thus, we have the following.

Corollary 14. *For any regular MSC language $\mathcal{L}$ (respectively, locally synchronized MSG $\mathcal{G}$), it is decidable if $\mathcal{L}$ (respectively, $L(\mathcal{G})$) is causally realizable.*

Every regular MSC language is recognized by a deterministic MPA. Our construction for the causal closure preserves determinacy. We can check this deterministic MPA for deadlocked states, which immediately yields the following.

Corollary 15. *Let $\mathcal{L}$ be a regular MSC language. It is decidable if $\mathcal{L}$ is causally realizable and admits a deadlock-free implementation.*

Not all regular MSC languages are MSG-definable. A regular MSC language is MSG-definable precisely when it is *finitely generated*—that is, there is a finite set of MSCs $\mathit{Atoms} = \{M_1, M_2, \ldots, M_k\}$ such that every MSC in the language can be written out as a sequence $M_{i_1} \circ M_{i_2} \circ \cdots \circ M_{i_\ell}$ where each $M_{i_j} \in \mathit{Atoms}$ [9].

Proposition 16. *There exist regular MSC languages $\mathcal{L}$ such that $\mathcal{L}$ is MSG-definable but $\mathit{CausalCl}(\mathcal{L})$ is not.*

Proof. The MSG in Fig. 6 is locally synchronized and hence defines a regular MSC language. In the same figure is shown a family of MSCs $\{M_n\}_{n \in \mathbb{N}}$, each of which is causally implied by the language of this MSG. However, observe that each M_n is an *atomic* MSC that cannot be written as the asynchronous concatenation $M_1 \circ M_2$ of two nontrivial MSCs. Thus, the causal closure of this MSG language is regular but not MSG-definable. ☐

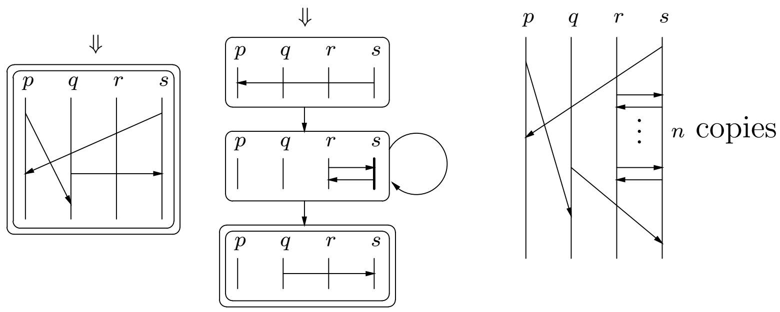

Fig. 6. MSG definability and causal closure

Moving to arbitrary MSGs takes us into the realm of undecidability. We have the following results, whose proofs are omitted.

Theorem 17. *For an arbitrary MSG $\mathcal{G}$, it is undecidable whether $L(\mathcal{G})$ is causally realizable. It is also undecidable whether the causal closure of $L(\mathcal{G})$ is a regular MSC language.*

5 MSGs as partial specifications

The realizability question for MSGs asks whether the set of scenarios represented by an MSG corresponds exactly to the language of a suitably defined MPA. To ensure that a specification is realizable, we need to impose severe restrictions on the structure of the MSG. This leads to an explosion in the complexity of the MSG and detracts significantly from the main motivation for using this notation, which is to have a transparent and visually appealing formalism to describe the behaviour of communicating systems.

An alternative is to view MSGs as partial specifications and interpret them modulo the closure conditions required by distributed implementations. This approach was studied in the context of Petri nets in [5]. For MPAs, we cannot use weak closure as the semantics of MSGs because weak closure does not preserve regularity. However, since the causal closure does preserve regularity, it is feasible to use this as a semantics for MSGs.

Under the exact interpretation, locally synchronized MSGs correspond to the class of finitely-generated regular MSC languages [9]. If we interpret MSGs modulo causal closure, an MSG can represent languages that are not finitely generated (like the example in Fig. 6). This increases the expressive power of the MSG notation. Another advantage is that we can sensibly analyze less complicated MSG specifications, making the notation more usable.

MSC languages can also be used to specify desirable properties of communicating systems. Two interpretations are possible: positive scenarios are those that the system must be able to exhibit, while negative scenarios should be avoided.

Suppose we use MSC languages both to specify the communicating system as well as to describe the scenarios that the system should exhibit. To verify that a set of positive scenarios P is included in the set of system behaviours S, it is important that both P and S are causally closed to avoid missing out some scenarios when checking this inclusion.

When model checking a negative property, it is again important to use the causal closure of the property. We have argued that reasonable implementations are causally closed. If the property is not causally closed, the implementation may exhibit an implied scenario that is forbidden, but this fact could go undetected.

In [7], it is shown that model checking of positive and negative scenarios under the exact interpretation can be performed for MSC languages where channels are existentially bounded—there is at least one linearization for each MSC in the language for which all channels are uniformly bounded. This is contrast to regular MSC languages, where channels are universally bounded across *all* linearizations. The properties of existentially bounded MSC languages are further elaborated in [6]. Unfortunately, the causal closure of an existentially bounded MSC language need not be existentially bounded. It would be interesting to identify when the causal closure of an existentially bounded MSC languages remains existentially bounded so that the results of [7] can be applied modulo causal closure.

References

[1] Alur, R., Yannakakis, M.: Model checking of message sequence charts. *Proc. CONCUR 1999)*, Springer Lecture Notes in Computer Science **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] Caillaud, B., Darondeau, P., Hélouët, L. and Lesventes, G.: HMSCS as partial specifications . . . with PNs as completions. In *Modeling and Verification of Parallel Processes, 4th Summer School, MOVEP 2000*, Nantes, France (2000).

[6] Genest, B., Muscholl, A., and Kuske, D.: A Kleene Theorem for a Class of Communicating Automata with Effective Algorithms. *Proc DLT 2004*, Springer Lecture Notes in Computer Science **3340** (2004) 30–48.

[7] Genest, B., Muscholl, A., Seidl, H. and Zeitoun, M.: Infinite-State High-Level MSCs: Model-Checking and Realizability. *Proc ICALP 2002*, Springer Lecture Notes in Computer Science **2380** (2002) 657–668.

[8] Harel, D., Gery, E.: Executable object modeling with statecharts. *IEEE Computer*, July 1997 (1997) 31–42.

[9] 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.

[10] ITU-TS Recommendation Z.120: *Message Sequence Chart (MSC).* ITU-TS, Geneva (1997).

[11] Lohrey, M.: Safe Realizability of High-Level Message Sequence Charts. *Proc CONCUR 2002*, Springer Lecture Notes in Computer Science **2421** (2002) 177–192.

[12] Mauw, S., Reniers, M. A.: High-level message sequence charts, *Proc SDL'97*, Elsevier (1997) 291–306.

[13] Mukund, M., Narayan Kumar, K., Sohoni, M.: Bounded time-stamping in message-passing systems. *Theoretical Computer Science*, **290(1)** (2003) 221–239.

[14] Muscholl, A., Peled, D.: Message sequence graphs and decision problems on Mazurkiewicz traces. *Proc. MFCS 1999)*, Springer Lecture Notes in Computer Science **1672** (1999) 81–91.

[15] Rudolph, E., Graubmann, P., Grabowski, J.: Tutorial on message sequence charts. In *Computer Networks and ISDN Systems — SDL and MSC* **28** (1996)