# High-Level Message Sequence Charts and Projections

Blaise Genest[1], Loïc Hélouët[2], and Anca Muscholl[1]

[1] LIAFA, Université Paris VII, 2 place Jussieu, 75251 Paris, France
[2] IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France*
{genest,anca}@liafa.jussieu.fr,loic.helouet@irisa.fr

**Abstract.** Abstraction is a key issue in automatic verification, and it is often performed by a projection on a subsystem that is relevant for the property to check. This paper defines projections for the scenario language of High-level Message Sequence Charts (HMSC). We first show that the projection of an HMSC is not representable as an HMSC, in general. However, we show how that projections of HMSCs can be represented by a larger class of scenario languages, namely by (realizable) compositional HMSCs (cHMSCs). Moreover, we propose an algorithm that checks whether the projection of an HMSC can be represented by an HMSC, constructing the HMSC representation, when possible. This can be used in model-checking the projection of an HMSC specification.

## 1 Introduction

Scenario languages such as Harel's Live Sequence Charts [7], UML sequence diagrams, interworkings, etc. have seen a growing interest this last decade. Among them, the ITU standardized notation of Message Sequence Charts (MSCs, [10]) has received a lot of attention, both in the area of formal methods and in automatic verification [1, 2, 8, 13, 11, 15]. MSCs can be considered as an abstract representation of communications between asynchronous processes. They are usually used as requirements, documentations, abstract test cases, and so on. A common approach for modeling the behavior of distributed systems is to describe them by means of parallel composition of communicating instances. MSCs and high-level MSCs (HMSCs for short) propose a pictorial way of modeling behaviors, combining parallel composition (processes) with sequential composition (transition system). The main advantage of such a representation is to have a local, explicit description of the communication and the causalities appearing in the system. Even if HMSCs seem to be a very simple formalism, the explicit use of parallel composition leads to model-checking being in general undecidable ([2, 13]). By model-checking we mean validating HMSC specifications versus simple properties, that are either given in some sequential logics/automata or as HMSCs. Some weaker decidable comparison criteria have been proposed, such as matching with gaps [12, 14]. Notice also that model-checking partial order

* The work of this author has been partially supported by CAFE Eureka ITEA ip00004 $\Sigma$! 2023

properties (i.e., specified using partial order logics) is decidable, albeit of high complexity, [11, 15].

Abstraction often appears as a central issue for automatic verification of large systems [3, 5, 16], since it can decrease the complexity of the verification process. In general, the issue is how to obtain an abstraction that allows to transfer the result back to the initial system. Scenarios are supposed to remain rather concise, but in some cases they may have been designed with too many details, which are not relevant for the property to check. Moreover, the details may hide important information concerning the causalities. Abstraction for HMSCs can be performed by collapsing nodes of the graph, as for Kripke structures. The problem is that such an abstraction can produce more behaviors than the initial model. Thus the result of model-checking is only an approximation for the real model. Abstraction can also be defined by projecting away some events or instances, without changing the graph. This abstraction of an HMSC is still an HMSC, but has as a negative side effect the loss of certain causalities between events. For example, if one wants to know whether an event $a$ on process $p$ appears before $b$ on process $q$, then she might want to verify it only on the subsystem obtained by projecting away all events not on $\{p, q\}$. However, in this subsystem, the events $a, b$ might become unordered, which means that the abstraction will not preserve the property.

The first motivation of this paper is to give an exact abstraction of HMSCs, i.e. with no approximation. Our abstraction hides (projects away) specific events while preserving the causalities. This can have several benefits. First, it can provide a better comprehension of the interactions between particular instances (see Figure 3 for a concrete example). Second, when comparing two MSCs, a designer might be interested in comparing the behaviors involving common features of both scenarios. Hiding information while preserving causal dependencies becomes then a central point for this kind of comparison. Our abstraction will preserve the causal order, hence the property $(\neg b)\mathcal{U}a$ of the example above. More generally, if the projection is defined on events on a given set of processes $P$, then any formula defined by $\phi = a \mid \neg\phi \mid \phi\mathcal{U}_P\phi$ is satisfied by the projection if and only if it is satisfied by the concrete system. Here, the atomic propositions $a$ concern only processes in $P$. The operator $\mathcal{U}_P$ is a restriction of $\mathcal{U}$ meaning that the 'until' is taken among the processes in $P$. That is, $\phi\mathcal{U}_P\psi = (\phi \vee \neg P)\mathcal{U}\psi$. Finally, a motivation for hiding actions in an HMSC is to be able to verify properties on a model that is hopefully smaller.

The main problem raised by projections of HMSCs that preserve the causalities is the representation of the projected HMSC. First, the projection of an MSC is not always an MSC, as hiding may produce events that represent at the same time sends and receives. A more severe problem is that, projected HMSCs (even bounded ones, [2, 13]) cannot be represented by means of a *finite HMSC*, in general. The first main result of the paper is that we can always represent the projection of an HMSC by a realizable *compositional* HMSC (cHMSCs, [6]). Moreover, we give an algorithm that tests whether the projection of an HMSC can be represented by an HMSC. The second main result is an effective construction of an HMSC representing the projection of HMSC, whenever this is possible.

This paper is organized as follows. Section 2 introduces basic notions related to MSCs, and section 3 defines the projections. Section 4 shows on a concrete protocol (RMTP2) the reasons that can prevent an HMSC projection to be represented as an HMSC. Section 5 establishes the effective equivalence between projections of HMSCs and realizable cHMSCs. The main result of the paper is given in section 6. It states that we can decide in polynomial space whether a realizable cHMSCs (in particular, the projection of an HMSC) can be represented by an HMSC. In the affirmative case, we can build effectively such an HMSC. Finally, in section 7 we show that model-checking projections of HMSCs is decidable under some reasonable assumptions. Due to space limitations most proofs are omitted.

## 2 Preliminaries

Message Sequence Charts (MSC for short) is a scenario language standardized by the ITU ([10]). They represent simple diagrams depicting the activity and communications in a distributed system. The entities participating in the interactions are called instances (or processes) and are represented by vertical lines. Message exchanges are depicted by arrows from the sender to the receiver. In addition to messages, atomic actions and timer operations (set, reset and timeout) can also be represented. Figure 1-a gives an example of an MSC modeling interactions between a *Sender*, a *Medium* and a *Receiver*.

Formally, an *MSC* $M$ is a partially ordered set (poset) described by a tuple $M = (E, \leq, A, \mathcal{P}, t, P, m)$, where $E$ is a finite set of events, $\leq \subseteq E \times E$ is a partial order on $E$ and $A = A^s \cup A^r \cup A^{at}$ is a set of actions, which can be sending, receiving or atomic actions (internal operations or events related to timers). $\mathcal{P}$ is a finite set of $\wp$ processes (instances), $t : E \longrightarrow A$ associates an action with each event and $P : E \longrightarrow \mathcal{P}$ associates a process with each event. The message relation $m \subseteq E \times E$ pairs up sending and receiving events such that each send has a unique associated receive, and vice-versa. A send action is denoted by $p!q(a)$, meaning that $p$ sends to $q$ the message $a$. A receive action is denoted by $q?p(a)$, meaning that $q$ receives message $a$ from $p$. The message relation $m$ is consistent with the mapping $t$, i.e., if $(e, f) \in m$, then $t(e) = p!q(a)$ and $t(f) = q?p(a)$ for some $p, q, a$.

The *visual order* of $M$ is given by a total order on each process $p \in \mathcal{P}$, i.e. on each set of events $E \cap P^{-1}(p)$ (process ordering) and by the message ordering $e \leq f$ for every message $(e, f) \in m$. The graphical representation of an MSC diagram actually corresponds to the visual order, consisting of vertical lines (process ordering) and message arrows (message ordering). The relation $\leq$ is the partial order generated by the visual order. We write $e \lessdot f$ when $e < f$ and there is no event $g$ with $e < g < f$. It is a common assumption that interprocess communication is FIFO, i.e., there is no overtaking of messages on any channel. Figure 1-b depicts the Hasse diagram of the partial order of the MSC in Figure 1-a.

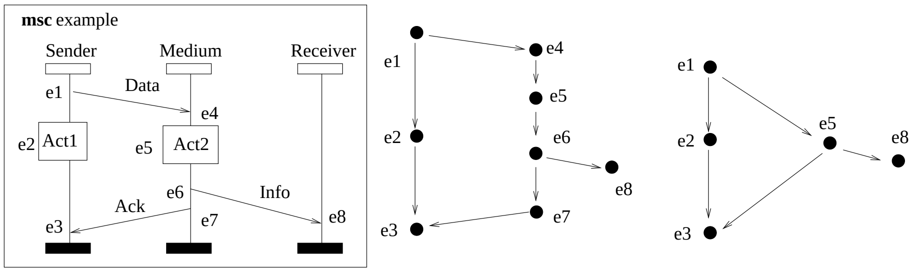



**Fig. 1.** a) MSC example b) partial order associated c) projection

MSCs are not expressive enough for specifying sets of scenarios and need operators such as choice or iteration to model interesting behaviors. This leads to High-level Message Sequence Charts (HMSCs) [10], that are just transition systems with nodes labeled by MSCs. Their semantics is defined by the *sequential composition* of two MSCs $M_1, M_2$. Let $M_i = (E_i, \leq_i, A_i, \mathcal{P}_i, t_i, P_i, m_i)$. Their sequential composition is the MSC $M_1 \circ M_2 = (E_1 \uplus E_2, \leq, A_1 \cup A_2, \mathcal{P}_1 \cup \mathcal{P}_2, t_1 \cup t_2, P_1 \cup P_2, m_1 \cup m_2)$, where $\leq$ is the transitive closure of $\leq_1 \cup \leq_2 \cup \{(e_1, e_2) \in E_1 \times E_2 \mid P_1(e_1) = P_2(e_2)\}$ defined on the disjoint union of events $E_1 \uplus E_2$. Intuitively, the sequential composition glues two MSCs along their common instance axis.

An *HMSC* is a tuple $H = (V, \rightarrow, v^0, F, \lambda)$ where $(V, \rightarrow, v^0, F)$ is a transition system with set of nodes $V$, transition relation $\rightarrow$, initial node $v^0$ and set of final nodes $F$. Each node $v$ is labeled by an MSC, denoted $\lambda(v)$. An initial path of $H$ is defined as a sequence of transitions $\rho = (v_1 \rightarrow v_2 \cdots \rightarrow v_k)$ with $v_1 = v^0$. If moreover $v_k \in F$, then $\rho$ is an accepting path. The MSC $\lambda(\rho)$ associated with a path $\rho$ is the sequential composition of the MSCs labeling the nodes, $\lambda(\rho) = \lambda(v_1) \circ \cdots \circ \lambda(v_k)$. An HMSC $H$ defines a set of (finite) MSCs $\mathcal{L}(H) = \{\lambda(\rho) \mid \rho \text{ is an accepting path of } H\}$. Figure 2-a depicts an HMSC involving processes $A, B, C$.

*Compositional MSCs* (cMSC for short) is a notation that extends MSCs [6]. The difference between a cMSC and an MSC is that the message function $m$ is a partial function, i.e., there can be sends or receives for which no matching event is defined. Such an event is called here *isolated*. The sequential composition of cMSCs is defined as above, with the additional requirement that the message function of $M$ eventually matches isolated sends of $M_1$ with isolated receives of $M_2$ in such a way that $M$ respects the FIFO condition. Hence, if two isolated sends from process 1 to 2 are concatenated with two isolated receives from process 1 to 2, then the resulting MSC consists of two successive messages from 1 to 2. See [6] for details.

A *compositional* HMSC $H$ is an HMSC with nodes labeled by cMSCs. It defines a set of cMSCs $\mathcal{L}(H) = \{\lambda(\rho) \mid \rho \text{ is an accepting path of } H\}$. Moreover, a cHMSC is called *realizable* in [6], if there is no accepting path with isolated events.

We end this section by some properties needed later in our algorithms. An MSC $M$ is called *atomic*, if it cannot be written as $M = M_1 \circ M_2$, where $M_1, M_2$ are non-empty MSCs. [9] describes an algorithm for testing whether an MSC $M$ is atomic. This algorithm tests in linear time whether the following graph is strongly connected:

**Definition 1** *Let $M = (E, \leq, A, \mathcal{P}, t, P, m)$ be an MSC over set of events $E$. The connection graph $CG(M) = (E, \rightarrow_{CG})$ of $M$ is defined by $v_1 \rightarrow_{CG} v_2$ if either $P(v_1) = P(v_2)$ and $v_1 \lessdot v_2$, or one of $(v_1, v_2)$, $(v_2, v_1)$ is a message in $M$ (edges are added from receives to associated sends).*

**Proposition 1** *[9] An MSC $M$ is atomic if and only if the connection graph $CG(M)$ is strongly connected.*

## 3 Projections of MSCs

Consider an MSC $M = (E, \leq, A, \mathcal{P}, t, P, m)$ and a subset of events $E' \subseteq E$. The *projection* of $M$ on $E'$ is noted $\pi_{E'}(M)$, and is obtained by erasing the events in $E \setminus E'$, and inheriting the causal dependencies from $M$. The set $E'$ can represent for example all the events located on a subset of processes (instances). Formally, $\pi_{E'}(M)$ is the restriction of the poset $M$ to $E'$, defined as $\pi_{E'}(M) = (E', \leq', \mathcal{P}, P', m')$, where $\leq', P'$ are the restrictions of $\leq, P$ to $E'$. Events from the set $E'$ will be called *non-erased* events. In the same way as for MSCs we depict the causal dependency between different processes by a causality (aka message) relation $m'$. We let $(e, f) \in m'$ for two non-erased events $e, f \in E'$, if $e \lessdot' f$ and $P(e) \neq P(f)$. That is, $e$ and $f$ are events located on different processes, $e < f$ in $M$ and there is no intermediate non-erased event $g \in E'$ with $e < g < f$. The projection of an MSC will be called a pMSC for short.

Note that a pMSC is not necessarily an MSC, since an event in the projection may gather several actions of the initial MSC (these events will be called *multi-type events*). The example of Figure 1-c shows the projection of the MSC in Figure 1-a on $E' = \{e_1, e_2, e_3, e_5, e_8\}$. Since $e_1 \lessdot e_5$, we have to create a message between $e_1$ and $e_5$ to keep ordering. Similarly, as $e_5 \lessdot e_8$, we have to create a message between $e_5$ and $e_8$. In the projection, event $e_5$ has several types, a receive from the *Sender* process and two sends to *Receiver* and *Sender*. However, multi-type events are not a real problem for modeling, as pMSCs are still partially ordered event sets.

We can define *atomic* pMSCs similarly to MSCs: A pMSC $M$ is atomic if the connection graph $CG(M)$ is strongly connected. For example, the pMSC $M' = \pi_{\{e_1,e_2,e_3,e_5,e_8\}}(M)$ in Figure 1-c is atomic. Thus, in addition to the edges represented in the Hasse diagram of Figure 1-c the connection graph $CG(M')$ contains the back edges $(e_5, e_1), (e_3, e_5)$ and $(e_8, e_5)$, and is strongly connected. Note that projections do not preserve atomicity. In general, atoms of $\pi_E(M)$ can be larger than those of $M$.

For an HMSC $H = (V, \rightarrow, v^0, F, \lambda)$, the projection $\pi_{E'}(H)$ is defined as the projection of each MSC obtained from an accepting path of $H$ on all occurrences of events of $E'$. Let $v$ be a node of $H$, and let us denote by $E_v$ the events associated to the MSC $\lambda(v)$. Let $E'' = \biguplus_{v \in V} (E_v \cap E')$ be the set of occurrences of events of $E'$. Then, the set of pMSCs defined by the pHMSC $H' = \pi_{E'}(H)$ is the set $\mathcal{L}(H') = \{\pi_{E''}(\lambda(v_1) \circ \cdots \circ \lambda(v_k)) \mid v_1 \to \cdots \to v_k$ is an accepting path of $H\}$. We will call the projection of an HMSC a *pHMSC*.

## 4 Comparing pHMSCs with HMSCs

The description of a pHMSC by an HMSC, together with a projection function, has several drawbacks. First, since causal dependencies are only implicitly given by the HMSC, a projected scenario is difficult to understand. Second, an implicit representation is not convenient for algorithmic manipulations. Third, by projecting an HMSC we usually want to obtain a smaller object, with a more compact representation. An immediate question appears when projecting an HMSC $H$ to some pHMSC $H' = \pi_E(H)$, namely whether there exists some equivalent HMSC $G$, i.e., such that $\mathcal{L}(G) = \mathcal{L}(H')$? In particular, if $H'$ is equivalent to some HMSC, then there exists a finite set $X$ of generators for $\mathcal{L}(H')$. That is, there exists a finite set $X$ of MSCs such that every $M \in \mathcal{L}(\pi_E(H))$ is a product of elements from $X$. We show below two situations that can prevent the existence of such a set $X$.

The first case is called an *unbounded crossing*. Intuitively, a pHMSC contains an unbounded crossing if there is a communication pattern that can be iterated an arbitrary number of times between two events situated on different processes that are causally related. For example, the HMSC of Figure 2-a generates an unbounded crossing for a projection on the instances $A$ and $B$. Figure 2-b shows the partial orders generated by the HMSC of Figure 2-a, and Figure 2-c shows the partial orders after the projection on $A$ and $B$. The MSCs in the projection are all atomic, hence there is no finite set generating them.

A second situation ruling out a finite representation is called a *crown*. Let us illustrate the presence of a crown on an example. Consider the HMSC of Figure 3. This HMSC describes scenarios for data transmission and acknowledgment for a multi-cast protocol called RMTP2. The RMTP2 network is organized as a tree, propagating data packets from a data source in the network, and aggregating acknowledgments in order to retransmit missed packets. Some nodes are designated to store a copy of the data sent, and retransmit each missed packet to the child subnetwork, if necessary. When a child receives a data packet, it may send an acknowledgment message *Hack* to its parent. A receiver may also decide to send an acknowledgment after a certain delay *tHackval*. The situation depicted by HMSC of Figure 3 shows communications between a node and two children. $\text{Child}_1$ always sends an acknowledgment to its parent upon data reception, while $\text{Child}_2$ always acknowledges data packets after the delay expiration. Furthermore, packets are never missed, but $\text{Child}_1$ can receive corrupted data, and retransmission and *Hack* packets may cross. The left part of Figure 3 shows the partial order associated with $(\textit{Hack\_incomplete} \circ \textit{Crossing})^*$. The right part of Figure 3 shows the same order after hiding the *Parent*, and all timer events.

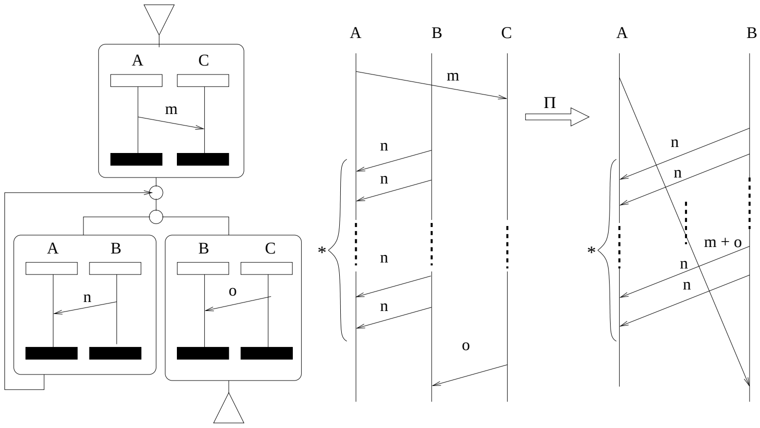



**Fig. 2.** a) HMSC generating an unbounded crossing b) MSC c) pMSC

It is clear that without breaking messages (or more precisely causal dependencies between distinct instances) the order obtained after projection can not be defined as a composition of finite communication patterns.

# 5 Projections and Compositional HMSCs

In the previous section we saw that pHMSCs can describe behaviors that cannot be captured by HMSCs. Still, we might ask whether there is some graphical model for pHMSCs that involves only the non-erased events. Natural candidates for such a model are *compositional HMSCs* (cHMSC) [6], which correspond to a message relation that is defined on a given path, rather than on each node. Actually, both situations described in section 4 can be easily described by cHMSCs. The main result of this section states that pHMSCs have equal expressive power with realizable cHMSCs.

Since pHMSCs involve in general multi-type events we actually need enriched versions of MSCs and cMSCs, by allowing multi-type events. For simplicity we will assume in this section that the projections do not generate multi-type events. However, our constructions can be easily adapted to this case.

We first note that cHMSCs are at least as expressive as pHMSCs. A formal proof will be given by showing later that one can decide whether a given pHMSC is equivalent to an HMSC, whereas this question is undecidable for cHMSCs.

Figure 4 below gives a rough idea of what kind of behavior of cHMSCs cannot be expressed as the projection of an HMSC. In the cHMSC in the left part of Figure 4 two or more isolated send events $\alpha$ are matched after an arbitrary number of $\beta$ messages. If we have to describe this behavior using a pHMSC, we would need for each event $\alpha$ a new process, that disappears through projection (processes $C_1, C_2$ in the middle part of Figure 4). More formally, for matching

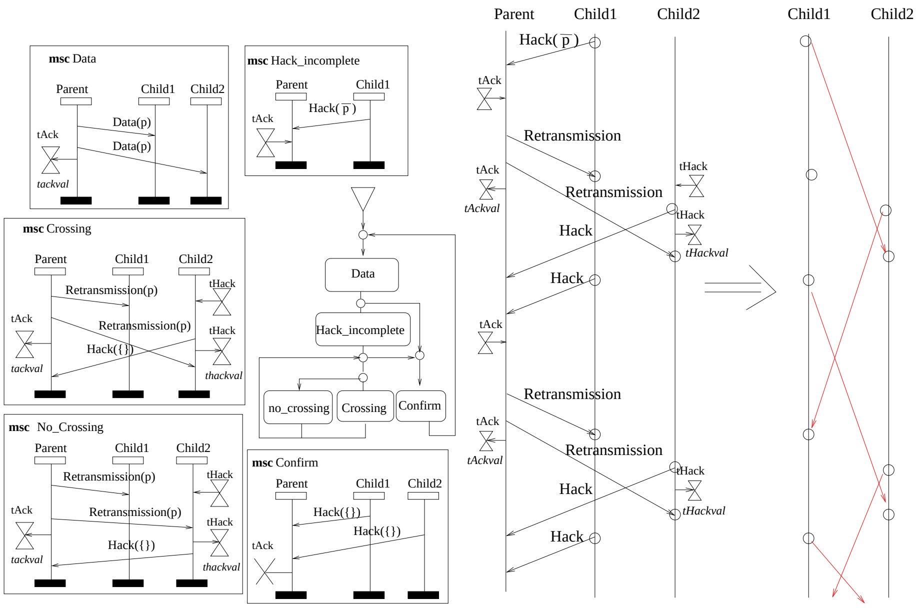


**Fig. 3.** A part of RMTP2 protocol and a crown generated by hiding instance Parent

a sequence $e_1 < \cdots < e_n$ of sends on process $A$ with a sequence $f_1 < \cdots < f_n$ of receives on process $B$, we need for each pair $e_i, f_i$ a new process $C_i$. If for instance $C_1 = C_2$, then $e_1 < e_2 < f_1 < f_2$, hence we do not have $e_2 \lessdot f_2$, that is we do not create a message from $e_2$ to $f_2$. Since the number of processes is fixed we therefore cannot describe the behavior of the cHMSC in the right part of Figure 4 by a pHMSC.

### 5.1 Comparing pHMSCs and cHMSCs

From the previous examples, cHMSCs seem good candidates for describing pHMSCs without multi-type events. We show in this section that pHMSCs correspond precisely to a subclass of cHMSCs, that of realizable cHMSCs. We recall that a cHMSC is called *realizable*, if every accepting path is labeled by an MSC. Throughout this section, $H$ will denote an HMSC and $H' = \pi(H)$ its projection. An event of the pHMSC $H'$ will be denoted as *non-erased* event.

Given a pHMSC, we will construct an equivalent cHMSC by guessing the type of the non-erased events. We check the types by restricting the transition relation of the cHMSC. Let $M$ be an MSC with event set $E$ and let $M' = \pi_{E'}(M)$ be the projection of $M$ onto the set of events $E' \subseteq E$. We have to guess the type of some events $e \in M'$ (send/receive/local, to which process etc). For example, in Figure 3 we have to guess that the first event on $\text{Child}_1$ becomes a send to $\text{Child}_2$ in the projection. In order to verify that the guess was correct, we need to

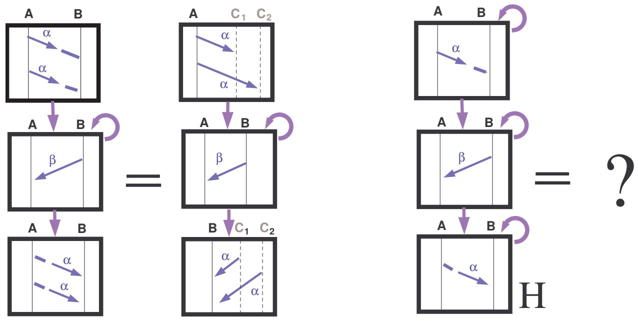



**Fig. 4.** The cHMSC in the right part is not a pHMSC.

keep track of the processes occurring in the future of $e$ within $M$. Among these processes we need also to know which processes are seen by a non-erased event $e' > e$ (such processes are called *dead*, since no event $f$ with $e \lessdot f$ can occur on such a process). Hence, let us define for each non-erased event $e \in E'$:

$$\mathrm{F}(e) = \{P(f) \mid e \leq f \in E\} \quad \text{(future processes)}$$
$$\mathrm{DeadF}(e) = \bigcup_{e < e' \in E'} F(e') \quad \text{(dead processes)}$$
$$\mathrm{LiveF}(e) = \mathrm{F}(e) \setminus \mathrm{DeadF}(e) \quad \text{(live processes)}$$

A non-erased event $e \in E'$ is called *unchecked* if $\mathrm{LiveF}(e) \neq \emptyset$. When an event $e$ is created, it is unchecked. Intuitively, $e$ becomes checked as soon as we have the proof that it can have no more immediate successor in the projection. It is easy to test the guess for a checked event. If the guess is correct, then we can forget the event, else the guess was wrong and the current path is not accepting. We show that $\mathrm{LiveF}(e) = \emptyset$ means that $e$ has no more immediate successor. Let $e < e'$ be non-erased and $p \in F(e')$. Assume by contradiction that $e$ is matched by some non-erased event $f$ after $M$ with $P(f) = p$. We have that $e < e' < f$, hence a contradiction.

The set of unchecked events of pMSC $M'$ is denoted $\mathrm{ToCheck}(M')$. The next lemma bounds the number of unchecked events on any path of a pHMSC $H' = \pi(H)$ polynomially in the number of processes.

**Lemma 1.** *Let $\rho$ be an initial path of an HMSC $H$ over $\wp$ processes, and $M'$ the pMSC defined by projecting $\rho$. Then $|ToCheck(M')| \leq \wp^2$.*

**Theorem 1** *Let $H$ be an HMSC with $n$ nodes over $\wp$ processes, and consider a projection $H'$ of $H$. Then we can construct a realizable cHMSC $G$ that is equivalent to $H'$, of size $n2^{O(\wp^3)}$.*

Our construction can be easily modified in the presence of multi-type events, for obtaining an extended realizable cHMSC.

A cHMSC is called *b-bounded* if for each initial path $\rho$, each prefix of $\rho$ is labeled by a cMSC with at most $b$ unmatched sends. Moreover, a realizable cHMSCs of size $s$ is $s$-bounded. Although the cHMSC obtained in Theorem 1 has exponential size, the number of unmatched sends is polynomial in the size of $H$ (see Lemma 1). We can restate Lemma 1 as follows:

**Proposition 2** *Let $H' = \pi(H)$ be a pHMSC and let $G$ be the realizable cHMSC constructed in Theorem 1. Then $G$ is $\wp^2$-bounded, where $\wp$ is the number of processes of $H$.*

For the converse direction, we construct a pHMSC from a cHMSC by introducing new processes for isolated events:

**Theorem 2** *Realizable cHMSCs and pHMSCs (with no multi-type events) have the same expressive power.*

For example, for an HMSC $H$ defined by a unique node with a self loop, labeled by a single message $(s, r)$ from $p$ to $q$, the cHMSC equivalent to the projection of $H$ on $\{p\}$ is a self loop on a local event of $p$ corresponding to $s$. This event is not a send anymore, since it has no immediate successor on a different process. We actually need unmatched sends only when dependencies are not preserved by the trivial projection (without rebuilding the order).

## 6 Atoms as Generators

In this section we show how to construct a compact representation of the generators (atoms) of a given realizable cHMSC $G$. This problem is directly related to the construction for a given realizable cHMSC $G$ (or a pHMSC) of an equivalent HMSC, if it exists, and it involves the computation of the smallest MSCs that are factors of some MSC $M \in \mathcal{L}(G)$. Formally, given a realizable cHMSC $G$ we want to compute the set $\text{Gen}(G)$ of *generators* of $G$ defined as follows. An *atomic* MSC $M$ belongs to $\text{Gen}(G)$ if $N = N_1 M N_2$ for some $N \in \mathcal{L}(G)$ and some MSCs $N_1, N_2$.

Clearly, for a realizable cHMSC to be equivalent to an HMSC, $\text{Gen}(G)$ needs to be finite. At the end of the section we show that if $\text{Gen}(G)$ is finite, then we can construct effectively an equivalent HMSC.

### 6.1 An Automata-Based Representation of Generators

We show now how to construct for a given realizable cHMSC $G$ a finite automaton $\mathcal{A}(G)$ that accepts only linearizations of $\text{Gen}(G)$ and such that for every $M \in \text{Gen}(G)$, at least one linearization of $M$ is accepted by $\mathcal{A}(G)$. The idea is to have a non-deterministic automaton $\mathcal{A}(G)$ that works as follows. Given an execution $z \in \mathcal{L}(G)$, the automaton guesses a factor $w$ of $z$ containing a generator $M \in \text{Gen}(G)$ and extracts the events of $M$ from $w$. Note that since we cannot choose the linearization $z$, such a generator $M$ won't be itself a factor of $z$.

The automaton $\mathcal{A}(G)$ must check two conditions in the definition of the generating set $\mathrm{Gen}(G)$. First, we guess for each event $e$ of the linearization $z$ above a type $k \in \{0, 1, 2\}$, telling whether $e$ belongs to $N_1$ ($k = 0$), to $M$ ($k = 1$) or to $N_2$ ($k = 2$). In addition, we check whether the guessed type is consistent, that is, that every send is of the same type as its associated receive. This will guarantee that $N_1, N_2$ and $M$ are all complete MSCs. The strategy is that $\mathcal{A}(G)$ will read the HMSC $G$ and write or not an event $e$ read according to its guess whether $e \in M$ or not.

The harder part is that $\mathcal{A}(G)$ must check that $M$ is atomic. Proposition 1 gives an algorithm for verifying that a given MSC $M$ is atomic, checking that the connection graph $CG(M)$ is strongly connected.

We have now to test whether an MSC $M$ is atomic using a finite automaton, that takes as input some arbitrary linearization of $M$. However, we cannot use directly the algorithm of [9], since the number of connected components is not bounded. In order to handle this, we restate the result of [9] as follows:

**Proposition 3** *Let $M$ be an MSC. Then $M$ is atomic if and only if for every pair of processes $p, q$, there is a path in $CG(M)$ from the last event of $p$ to the first event of $q$.*

We say that an event $e$ of $M$ *sees* another event $f$ if there exists a path from $e$ to $f$ in $CG(M)$.

The main idea behind the construction of $\mathcal{A}(G)$ is to keep some information for two kinds of events. First, $\mathcal{A}(G)$ needs to record the last event in $M$ (type $k = 1$) on each process. Let $\mathrm{last}_p$ denote the last event on process $p$ seen so far by $\mathcal{A}(G)$. Moreover, $\mathcal{A}(G)$ needs to take into account the unmatched send events in $M$ (type $k = 1$). Let $\mathcal{S}$ denote the set of these sends. The set $\mathcal{S}$ contains at most $b$ sends, where $b$ is the size of the cHMSC $G$.

Let $X = \{\mathrm{last}_p \mid 1 \leq p \leq \wp\} \cup \mathcal{S}$. Now, for each pair $(x, p) \in X \times \{1, \ldots, \wp\}$ we record an integer $T(x, p) \in \{0, 1, 2\}$ telling whether $x$ sees the first $p$-event in $M$ ($T(x, p) = 2$), or whether $x$ sees $\mathrm{last}_p$ but not the first $p$-event in $M$ ($T(x, p) = 1$), or whether it sees no $p$-event in $M$ at all ($T(x, p) = 0$). Actually, $\mathcal{A}(G)$ uses a "vision" function $T : x, p \in X \times \mathcal{P} \mapsto T(x, p)$, and a function $S : x \in X \mapsto S(x)$ where $S(x) \subseteq \mathcal{S}$ denotes the unmatched sends seen by $x$.

Clearly, computing the vision function of each event $\mathrm{last}_p$ suffices for deciding whether $CG(M)$ is strongly connected. This idea is used in the following construction.

**Theorem 3** *Let $G$ be a realizable cHMSC. Then we can construct effectively a finite automaton $\mathcal{A}(G)$ that accepts only linearizations of Gen($G$), such that for every $M \in$ Gen($G$) at least one linearization of $M$ is accepted by $\mathcal{A}(G)$.*

Using the automaton constructed in Theorem 3 we can test whether a given realizable cHMSC (or a pHMSC) is equivalent to an HMSC in polynomial space:

**Theorem 4** *Checking whether Gen($G$) is finite for a given realizable cHMSC $G$ (pHMSC $G$, resp.) can be done in PSPACE. Moreover, the problem is co-NP-hard.*

### 6.2 From Realizable cHMSCs to HMSCs

We have an algorithm for checking whether the MSC executions of a realizable cHMSC $G$ are finitely generated. We show in this section how to construct effectively an equivalent HMSC $H$, in case that the answer is positive. For simplicity, we will assume that states of $G$ are labeled by single events.

Let $\mathrm{Gen}(G)$ be the finite set of atoms of the realizable cHMSC $G$. We denote by *maxsize* the size of the largest atom of $\mathrm{Gen}(G)$ and by $b$ the bound on unmatched sends on paths of $G$.

We first describe intuitively the construction. The HMSC $H$ will have states labeled by the atomic MSCs in $\mathrm{Gen}(G)$. In addition, we will label each state with some information concerning paths of $G$ that can correspond to the sequence of atoms read so far in $H$. This additional information consists of a sequence of segments of a path of $G$ (i.e. a *sub-path*), that match this sequence of atoms. That is, each time we read an atom $A \in \mathrm{Gen}(G)$, we guess new segments of the path of $G$ that correspond to the MSC $A$. We keep track of path segments by recording only the first/last node of each segment and the processes occurring in the segment. Hence, all we need is that the number of segments is bounded (see the claim below).

We first define the HMSC $H$. Let Path be the set of sub-paths of $G$ consisting of at most $(b+1)\cdot\mathrm{maxsize}$ segments. The set of nodes of $H$ is $V = \mathrm{Gen}(G)\times\mathrm{Path}$. A node $(A,\rho) \in V$ is labeled by $A$. Moreover, there is an edge in $H$ from a node $(A,\rho)$ to $(A',\rho')$ iff $\rho \subsetneq \rho'$, and $\lambda(\rho') = \lambda(\rho) \circ A$. Here, we write $\rho \subseteq \rho'$ if $\rho$ is included in $\rho'$, i.e. each segment of $\rho$ is included in a segment of $\rho'$, and in the same order. The initial node is $(\emptyset,\epsilon)$, and the final nodes $(A,\rho)$ are those where $\rho$ is a one-segment, accepting path of $G$.

**Theorem 5** *Let $G$ be a cHMSC where Gen($G$) is finite. Let $n$ be the number of nodes of $G$, $e$ the number of events, maxsize the maximal size of MSCs in Gen($G$) and $b$ the maximal number of unmatched sends on initial paths of $G$. Then we can construct an equivalent HMSC $H$ from $G$ of size $O((n^{2b}\cdot 2^{\wp b}\cdot e)^{maxsize})$.*

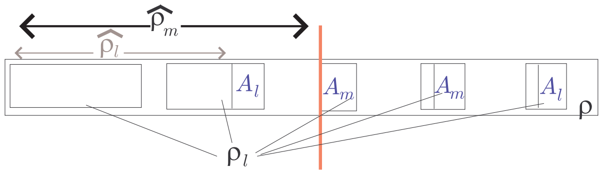
*Proof.* Let $M \in \mathcal{L}(H)$, then there exists an initial path of $G$ labeled by $M$. Conversely, let us consider an MSC $M = A_1 \cdots A_n$, where each $A_i$ belongs to $\mathrm{Gen}(G)$. This MSC labels an accepting path $\rho = v_1 \to \cdots \to v_k$ of $G$.

For simplicity, we extend the visual order of $M$ to the atoms $A_i$ by letting $A_i \leq A_j$ if there are some events $e$ in $A_i$, $f$ in $A_j$ with $e < f$. Now, we will assume w.l.o.g. that for every $i < j$ such that $A_i \not\leq A_j$ the first event of $A_i$ in $\rho$ comes before the first event of $A_j$ in $\rho$. That is, we choose an ordering of the atoms of $M$ according to their first occurrence in $\rho$.

**Claim:** Let $l \in \{1,\ldots,n\}$ and let $\rho_l$ be the sequence of segments of $\rho$ labeled by $A_1 \cdots A_l$. Then $\rho_l$ consists of at most $(b+1)\cdot\mathrm{maxsize}$ segments.

*proof of the claim:* We denote by $\widehat{\rho_j}$ the longest prefix of $\rho$ that contains no event of $A_j$. Let also $m$ be such that $\widehat{\rho_m}$ is the longest prefix $\widehat{\rho_j}$ with $j \leq l$.

The pMSC labeling $\widehat{\rho_m}$ has at most $b$ unmatched sends, among which $b'$ sends belong to $\rho_l$. Thus, $b'' = b - b'$ is the number of unmatched sends in $\widehat{\rho_m} \setminus \rho_l$ (the difference of two paths is obtained by deleting the nodes of the second path from the first one).




We claim first that there is no complete atomic MSC $A_p$ in $\widehat{\rho_m} \setminus \rho_l$. To see this, note first that such a complete MSC $A_p$ must satisfy $p > l$. However, it can satisfy neither ($A_p \not\leq A_m$ and $A_m \not\leq A_p$), by the choice of the ordering $A_1 \cdots A_n$, nor $A_m < A_p$, since $\widehat{\rho_m}$ has an empty intersection with $A_m$. Since each incomplete atom of $\widehat{\rho_m} \setminus \rho_l$ contributes with at least one unmatched send in $\widehat{\rho_m}$, we obtain that there are at most $b'' \cdot (\text{maxsize} - 1)$ events in $\widehat{\rho_m} \setminus \rho_l$, thus at most $b'' \cdot (\text{maxsize} - 1) + 1$ segments in $\rho_l \cap \widehat{\rho_m}$.

Moreover, by definition of $m$ there is just one new atom starting in $\rho_l \setminus \widehat{\rho_m}$, namely $A_m$. Hence there are at most $b' + 1$ different atoms in $\rho_l \setminus \widehat{\rho_m}$ ($b'$ that started already in $\widehat{\rho_m}$ plus $A_m$). This yields at most $(b' + 1) \cdot \text{maxsize}$ events in $\rho_l \setminus \widehat{\rho_m}$, hence at most $(b' + 1) \cdot \text{maxsize}$ segments. Therefore, we conclude that $\rho_l$ contains at most $(b' + b'' + 1) \cdot \text{maxsize} = (b + 1) \cdot \text{maxsize}$ segments.

Concerning the size of $H$, for each path segment it suffices to remember the first/last node, and the processes that occurred in the segment. This gives at most $(|G|^2 \cdot 2^{\wp})^{b \cdot \text{maxsize}} = 2^{2b(\log(|G|)+\wp)\text{maxsize}}$ paths consisting of less than $b \cdot \text{maxsize}$ segments. $\square$

*Example:* Let $G$ be a cHMSC with 4 states $Q, R, S, T$, labeled respectively by $s, s, r, r$ where $s$ is a send on process 1 corresponding to a receive $r$ on process 2. There are edges from $Q$ to $R$, $R$ to $S$, $S$ to $R$, and $S$ to $T$. Let $A$ be the atom consisting of the message $(s, r)$. The set of atoms of $G$ is $\{A\}$, hence each state of $H$ is labeled by $A$ (or $\emptyset$). We describe some of the resulting states of $H$:

- $s_1$ is the state $[Q, \{1\}]; [S, \{2\}]$, consisting of two segments, one being a path from $Q$ to $Q$ and the other from $S$ to $S$,
- $s_2$ is the state $[Q, \{1\}]; [T, \{2\}])$,
- $s_3$ is the state $([Q, S, \{1, 2\}]; [S, \{2\}])$ consisting of two segments, one being a path from $Q$ to $S$, the other the path from $S$ to $S$,
- $s_4$ is the (final) state $([Q, T, \{1, 2\}])$.

We have for instance edges from $(\epsilon, \emptyset)$ to $s_1$ and $s_2$. However, $s_2$ is a bad guess (that is, it will never reach the final state $s_4$). The reason is it cannot be extended by $A$: the segment $[T, \{2\}]$ forbids using the node $v_3$, since it already contains process 2. There are edges from $s_1$ to $s_3$, $s_3$ to $s_4$, $s_1$ to $s_4$ and a loop on $s_3$. The loop on $s_3$ corresponds to the guess of a scattered sub-path $[R, \{1\}]; [S, \{2\}]$ labeled by $A$, and to the guess $[R, \{1\}]$ making the connection between $([Q, S, \{1, 2\}]$ and $[S, \{2\}])$, while $[S, \{2\}]$ is a disjoint segment.

Since there are at most $2^{O(\text{maxsize})}$ atoms, $H$ is of exponential size. A priori, maxsize can be exponential in the size of the pHMSC, yielding an HMSC of doubly exponential size, but we believe this to be very unlikely. Actually, showing that maxsize is polynomial is as hard as showing that "Gen($G$) finite?" is NP-complete.

## 7 Model-Checking HMSCs against HMSCs

Validating HMSCs specifications allows detection of inconsistencies or undesired behaviors at early design stages. However, model-checking HMSCs specifications against properties specified by HMSCs is undecidable, in general (see e.g. [2, 13]). Several papers considered restrictions of HMSCs [2, 13, 8], for which model-checking becomes decidable. The first positive results were obtained for *bounded* HMSCs [2, 13], for which the set of MSC-linearizations is regular. A large family of HMSCs ensuring decidability of model-checking is given by the globally co-operative property [4]. An HMSC is globally cooperative if every loop is labeled by an MSC $M$ that cannot be written as $M = M_1 \circ M_2$, where $M_1, M_2$ are non-empty MSCs over disjoint sets of processes.

We consider in this section model-checking for pHMSCs against HMSC properties and we show two settings for which the problem is decidable, with the same complexity as for HMSCs.

The next theorem shows that model-checking remains decidable, if the pHMSC is arbitrary, but the HMSC property is globally cooperative:

**Theorem 6** *Let $G$ be a globally cooperative HMSC, $H$ an HMSC and $\pi_E(H)$ a projection of $H$. Then we can check in PSPACE whether $\mathcal{L}(\pi_E(H)) \cap \mathcal{L}(G) = \emptyset$, and in EXPSPACE whether $\mathcal{L}(\pi_E(H)) \subseteq \mathcal{L}(G)$.*

Even if the complexities we stated are rather high, note that in practice both the HMSC property and the reduced specification (the HMSC projected on a small part) can be reasonably small.

The second decidability result is based on the fact that the projection of a bounded HMSC preserves the boundedness.

**Theorem 7** *Let $G$ be a bounded HMSC, $H$ an HMSC and $\pi_E(G), \pi_{E'}(H)$ their respective projections. Then we can check in PSPACE whether $\mathcal{L}(\pi_E(G)) \cap \mathcal{L}(\pi_{E'}(H)) = \emptyset$, and in EXPSPACE whether $\mathcal{L}(\pi_{E'}(H)) \subseteq \mathcal{L}(\pi_E(G))$.*

The last result shows that we can compare two projections of HMSCs (e.g. in order to find common parts), as long as one of them is bounded, with the same complexity as for bounded HMSCs.

## 8 Conclusion

In this paper we defined projections of HMSCs and showed how to decide whether an HMSC projection can be represented as an HMSC. This notion of projection can then be used to perform model-checking when at least one of the HMSCs considered is either globally cooperative or bounded. Even when model-checking is not possible, HMSC projections may still be useful for a designer for extracting causality information from scenario descriptions. As already pointed out, the projection of an HMSC may result in a larger HMSC. However, this is just a worst-case estimation that is unlikely in practice. For example, hiding a complete instance would probably have a greater impact on the shape of the projection than a random choice of the hidden events. Moreover, the presence of unbounded crossings or crowns may reveal some properties of the system under study. For example, when a projection hides the communication medium, the presence of a crown can indicate the impossibility to save a consistent global snapshot of the system in some executions.

## References


1. R. Alur, G. Holzmann, and D. Peled. An Analyser for Message Sequence Charts. In *Proceedings of TACAS'96*, LNCS 1055, pages 35–48, 1996.
2. R. Alur and M. Yannakakis. Model Checking of Message Sequence Charts. In *Proceedings of CONCUR'99*, LNCS 1664, pages 114–129, 1999.
3. E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In *Proceedings of CAV'2000*, pages 154–169. LNCS 1855, 2000.
4. B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state HMSCs: Model-checking and realizability. In *Proceedings of ICALP'2002*, pages 657–668. LNCS 2380, 2002.
5. S. Govindaraju and D. Dill. Counterexample-guided choice of projections in approximate symbolic model checking. In *Proceedings of ICCAD'00*, 2000.
6. E. Gunter, A. Muscholl, and D. Peled. Compositional Message Sequence Charts. In *Proceedings of TACAS'01*, pages 496–511. LNCS 2031, 2001.
7. D. Harel and W. Damm. LSCs: breathing life into Message Sequence Charts. Technical Report CS98-09, Weizmann Institute, Avril 1998.
8. J. Henriksen, M. Mukund, K. Kumar, and P. Thiagarajan. On Message Sequence Graphs and finitely generated regular MSC languages. In *Proceedings of ICALP'99*. LNCS 1644, 1999.
9. L. Hélouët and P. Le Maigat. Decomposition of Message Sequence Charts. In *Proceedings of SAM2000(2nd conference on SDL and MSCs)*, Grenoble, Juin 2000.
10. ITU-TS. *ITU-TS Recommendation Z.120: Message Sequence Chart (MSC)*. ITU-TS, Geneva, September 1993.
11. P. Madhusudan. Reasoning about sequential and branching behaviours of Message Sequence Graphs. In *Proceedings of ICALP'01*, page 809. LNCS 2076, 2001.
12. A. Muscholl. Matching specifications for Message Sequence Charts. In *Proceedings of FoSSaCS'99*, LNCS 1578, pages 273–287, 1999.
13. A. Muscholl and D. Peled. Message Sequence Graphs and decision problems on Mazurkiewicz traces. In *Proceedings of MFCS'99*, LNCS 1672, 1999.
14. A. Muscholl, D. Peled, and Z. Su. Deciding properties for Message Sequence Charts. In *Proceedings of FoSSaCS'98*, LNCS 1378, page 226, 1998.
15. D. Peled. Specification and verification of Message Sequence Charts. In *Proceedings of FORTE'00*, pages 139–154. IFIP 183, 2000.
16. A. Pnueli. Abstraction, composition, symmetry, and a little deduction: The remedies to state explosion. In *Proceedings of CAV'2000*. LNCS 1855, 2000.