High-Level Message Sequence Charts (HMSC) projections (Full Version)

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 often a key issue in automatic verification, and is often performed through a projection on elements of importance with respect to a specific property. This paper proposes a definition of a projection for a scenario language called High-level Message Sequence charts (HMSC). We first show that projection of an HMSC can not always be represented by another HMSC. However, HMSC projections can still be represented by a bigger class of scenario languages, namely (realizable) cHMSCs. We propose a decision procedure to check if an HMSC projection can still be represented by an HMSC, and an algorithm to build it when possible. This result can be used to perform model-checking on an abstraction of the Model.

Keywords: Algorithm (Model-Checking), Scenario (MSC), Concurency, Abstraction.

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 [10] has received a lot of attention, both in the area of formal methods and in automatic verification [1, 8, 13]. MSCs can be considered as an abstract representation of communications in a system composed of asynchronous processes. They are usually used as requirements, documentations, abstract test cases, and so on. The usual approach for modeling the behavior of distributed systems is to describe them by means of parallel composition of communicating sequences. MSCs propose a new way of modeling behaviors based on the sequential composition of communication patterns. The main advantage of such a representation is to have an explicit description of the communication and the causalities appearing in the system. MSCs consist of basic diagrams (called bMSCs), that depict simple interactions between asynchronous instances, which are then composed with higher-level constructs (called HMSCs) that define sequential composition, choice and iteration. Even if MSCs seem to be a very simple formalism, their expressive power suffices for model checking to be in general undecidable ([2, 13]). By model-checking we mean validating HMSC specifications versus simple properties, that are either given in some logics or as HMSCs [11, 15]. Some weaker decidable comparison criteria have been proposed, such as matching with gaps [14, 12].

Abstraction often appears as a central issue for automatic verification of large systems [3, 5, 16]. With respect to this, MSCs are not an exception. Scenarios are supposed to remain rather concise, but in some cases they may have been designed with too many details, and important information concerning causalities might get lost. Abstraction for HMSCs can be performed in ignoring the differences between several nodes, as for automata. Such an abstraction can have more behavior than the intial system, thus we have an approximated specification. Abstraction can also be defined by projecting away (ignoring) some events or instances. The projected bMSCs are composed using the same HMSC. This abstraction of an HMSC is then still an HMSC and there is no theoritical difficulties for working with it. However, this notion of projection have side-effects: as messages are deleted, some causalities that were due to composition may disappear in the projected description. Hence, an abstraction can depict more behaviors than the initial system, and a counter-example for the abstraction may not be a counter-example for the original system.

The aim of this paper is to study an exact abstraction with no approximation. We will hide (project away) parts of the specification while preserving the causalities. This can have several benefits. First, it can provide a better comprehension of the interactions between particular instances (see Figure 4). Moreover, 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. Another reason for hiding actions in an MSC is to verify properties on a model that is hopefully smaller. Since we preserve causalities, partial-order properties are preserved in the reduced model. Finally, hiding becomes useful during the design process. Since it is usually an iterative process, hiding can help in finding common behaviors, and avoiding redundancy.

The main problem raised by taking the projection of certain elements of an MSC (i.e., hiding the other ones) while preserving causalities is the representation of the projected MSC. A projected MSC is not always an MSC, as hiding may produce events that represent at the same time message emissions and receptions. A more severe problem is that in general projected HMSCs (even bounded) cannot be represented by means of a finite set of generators. The main results of the paper is that we can always represents a projection of HMSC by a generalisation of HMSC called compositional HMSC. Moreover, we can effectively represent a projection of HMSC by a HMSC when it is possible.

This paper is organized as folows: section 2 introduces basic notions related to MSC, and section 3 presents HMSC projections. Then section 4 shows on an example (the RMTP2 protocol) the reasons that can prevent an HMSC projection to be represented as an HMSC. Section 5 establishes the effective equivalence between HMSC projections and an extension of HMSCs called realizable compositional HMSCs [6]. The main result of the paper is shown in section 6, and is that we can decide for both realizable compositional HMSCs and HMSC projections whether they can be represented as an HMSC in polynomial space. In the affirmative case, we can build effectively such an HMSC. Finally section 7 shows that model-checking directly HMSC projections is possible under some reasonable assumptions. Due to space limitations all proofs are ommited. For the full version see http://www.crans.org/˜genest/concur03_full_paper.ps

2 Preliminaries

Message Sequence Charts (MSC for short) is a scenario language standardized by the ITU ([10]). The specification formalism of MSCs is built over two levels. The lowest level is called basic MSC (bMSC, for short), and consists in simple diagrams depicting the activity and commnications in a distributed system. The entities participating in 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, a *bMSC* M is a partially ordered set (poset) described by a tuple $M = (E, \leq, A, \mathcal{P}, t, P, m)$ such that E is a finite set of events, $\leq \subseteq E \times E$ is a partial order on events 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 sending action is denoted by $p!q(m)$, meaning that p sends to q the message m. A receiving action is denoted by $q?p(m)$, meaning that q receives message m from p. The message relation m is consistent with the mapping t, i.e., if $(e, f) \in m$, then $t(e) = p!q(m)$ and $t(f) = q?p(m)$ for some p, q, m.

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 usual to assume that inter-process communication is FIFO: 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.

bMSCs are not expressive enough for specifying sets of scenarios and need operators such as choice or iteration to model interesting behaviors. Basic MSCs can then be composed with High-level Message Sequence Charts (HMSCs) [10], that are just transition systems with nodes labeled by bMSCs.

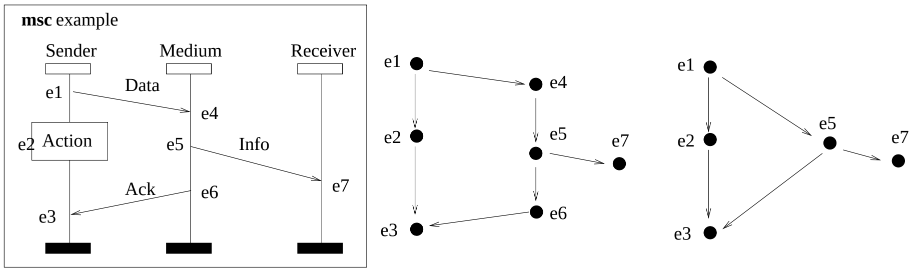


Fig. 1. a) bMSC example b) partial order associated c) projection

The *sequential composition* of two bMSCs M_1, M_2 with $M_i = (E_i, \leq_i, A_i, \mathcal{P}_i, t_i, P_i, m_i)$, is the bMSC $M_1 \circ M_2 = (E_1 \uplus E_2, \leq, A_1 \cup A_2, \mathcal{P}_1 \cup \mathcal{P}_2, t_1 \cup t_2, P_1 \cup P_2, m_1 \cup m_2)$, where $\leq$ is the transitive closure of $\leq_1 \cup \leq_2 \cup \{(e_1, e_2) \in E_1 \times E_2 \mid P(e_1) = P(e_2)\}$ defined on the disjoint union of events $E_1 \uplus E_2$. Intuitively, the sequential composition glues two bMSCs 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 ρ is an accepting path. The bMSC $\lambda(\rho)$ associated with a path ρ is the sequential composition of the bMSCs labeling the nodes, $\lambda(\rho) = \lambda(v_1) \circ \cdots \circ \lambda(v_k)$. An HMSC H defines a set of (finite) bMSCs $\mathcal{L}(H) = \{\lambda(\rho) \mid \rho \text{ is an accepting path of } H\}$. Figure 4a) 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 a bMSC 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 *isolated*. A cHMSC is an HMSC whose nodes are labeled by cMSCs. Sequential composition of cMSCs is defined as above, with the additionnal 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 is two sequential messages from 1 to 2.

A *compositional* HMSC H is an HMSC whose nodes are 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 if every accepting path is labeled by a bMSC.

The example of Figure 2-a shows two cMSCs $C1$ and $C2$. In $C1$, two occurrences of message Data are sent. In $C2$, two instances of message *Data* are received. The concatenation of $C1$ and $C2$ defined with the cHMSC of Figure 2-b is equivalent to the MSC of Figure Figure 2-c.

We end this section by some properties needed later in our algorithms. A bMSC M is called *atomic*, if it cannot be written as $M = M_1 \circ M_2$, where

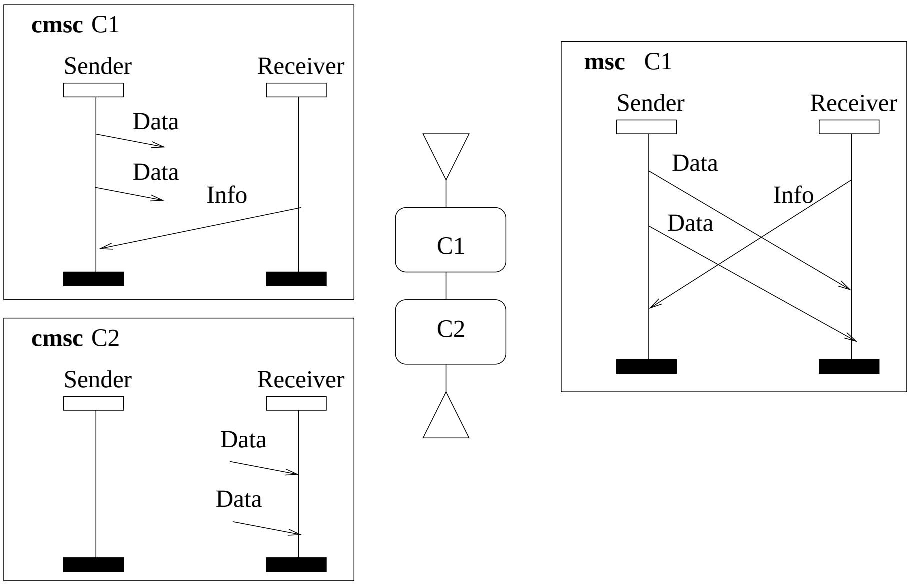


Fig. 2. a) two cMSCs b) cHMSC c) equivalent bMSC

M_1, M_2 are non-empty bMSCs. [9] describes an algorithm for testing whether a bMSC M is atomic. This algorithm tests (in linear time [17]) whether the following graph is strongly connected:

Definition 1 *Let $M = (E, \leq, A, \mathcal{P}, t, P, m)$ be a bMSC over set of events E. The connection graph $CG(M) = (E, \rightarrow_{CG})$ of M is defined by $(v_1, v_2) \in\rightarrow_{CG}$ 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] A bMSC M is atomic if and only if the connection graph $CG(M)$ is strongly connected.*

The left part of Figure 3-a shows a bMSC M that can be separated into four atoms and Figure 3-b shows the strongly connected components of CG(M).

3 Projections of MSCs

Consider a bMSC $M = (E, \leq, A, \mathcal{P}, t, P, m)$ 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

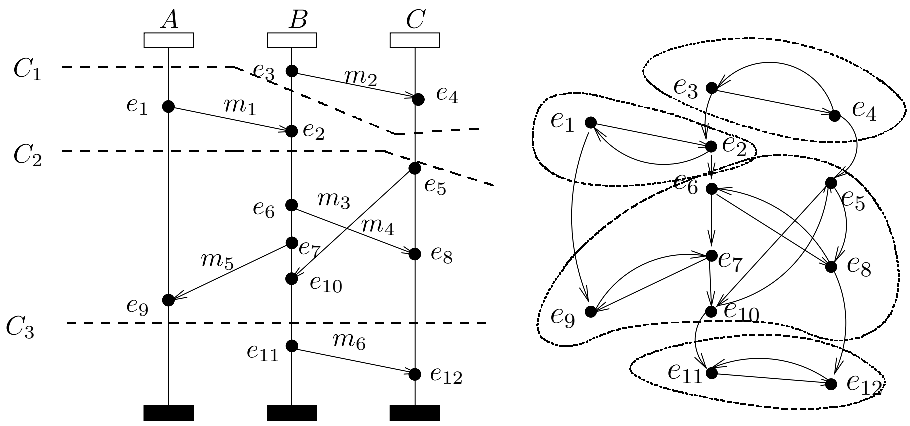


Fig. 3. a) Examples of atoms in an MSC b) connection graph and decomposition

E' will be called *unerased* events. In the same way as for bMSCs we depict the causal dependency between different processes by a causality (aka message) relation m'. We define $(e, f) \in m'$ for $e, f \in E'$ if $e \lessdot' f$ and $P(e) \neq P(f)$. That is, e and f are events located on different processes, $e < f$ are causally ordered in M and there is no intermediate unerased event $g \in E'$ with $e < g < f$. For simplicity, the projection of a bMSC will be called a pMSC.

Note that a pMSC is not necessarily a bMSC, since an event in the projection may represent the contraction of several actions in the initial bMSC (these events will be called *multitype 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, multitype events are not a real problem for modelling, as pMSCs are still partially ordered event sets.

We can define *atomic* pMSCs like *atomic* bMSCs: A pMSC M is atomic iff the connection graph $\mathrm{CG}(M)$ 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. In addition to the edges represented on the Hasse diagram of Figure 1-c the connection graph $CG(M')$ contains back edges $(e_5, e_1), (e_3, e_5)$ and (e_8, e_5), and is strongly connected. Moreover, if $M = M_1 \circ \cdots \circ M_n = M'_1 \circ \cdots \circ M'_m$ are two decompositions into atomic pMSCs, then $m = n$ and $M'_1, \ldots, M'_n$ can be obtained from $M_1, \ldots, M_n$ by exchanging adjacent pMSCs over disjoint sets of processes. Note that projecting does not preserve atomicity. In general, atoms of $\pi_E(M)$ can be larger than in M.

For notational simplicity we will call the projection of an HMSC a *pHMSC*. For an HMSC $H = (V, \rightarrow, v^0, F, \lambda)$, a projection $\pi_{E'}$ is defined as the projection of each bMSC obtained from an accepting path of H on all occurrences of events of E'. Let v_i be an node of H, and let us denote by E_i the events associated to the bMSC $\lambda(v_i)$. Let $E'' = \biguplus_{i \in 1..k} (E_i \cap E')$ be the set of occurrences of events of E'. Then, the set of partial orders 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 $H\}$.

4 pHMSCs are more expressive than HMSCs

A pHMSC is given by an HMSC, together with a projection function. This implicit representation has several drawbacks. First, since causal dependencies are implicitly given by the HMSC, the meaning of the projected scenarios is not easily understandable. Second, an implicit representation is not very convenient for algorithmic manipulations. Finally, by projecting an HMSC we usually want to obtain a smaller object, for which a more compact representation is desirable. Consequently, an immediate question appears when projecting an HMSC H to some pHMSC $H' = \pi_E(H)$. Is there any equivalent HMSC G, i.e., such that $\mathcal{L}(G) = \mathcal{L}(H')$? In other terms, can we find a finite set X of bMSCs (generators) such that any $M \in \mathcal{L}(\pi_E(H))$ is a composition of elements from X? Two problems may prevent us from finding a finite representation for $\mathcal{L}(\pi_E(H))$.

The first case encountered is called an *unbounded crossing*. Intuitively, an HMSC projection contains an unbounded crossing if there is a communication pattern that can be iterated an unlimited number of times between two events situated on different processes that are causally related. For example, the HMSC of Figure 4-a generates an unbounded crossing for a projection on the instances A and B. Figure 4-b shows the partial orders generated by the HMSC of Figure 4-a, and Figure 4-c shows the partial orders after the projection on A and B. In this picture, one can easily see that the event corresponding to sending message m and the event corresponding to receiving message o are causally related. Moreover, any atom containing these two events may also contain an arbitrary number of occurrences of message n.

Unbounded crossing is not the only configuration that may appear when the orders generated by an HMSC are projected, and which does not allow for an HMSC representation. The second case is called a *crown*. Let us illustrate the presence of a crown on an example. Consider the HMSC of Figure 4. This HMSC describes scenarios for data transmission and acknowledgement for a multicast protocol called RMTP2. The RMTP2 network is organised as a tree, propagating data packets from a data source in the network, and aggregating acknowledgements 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 acknowledgement message *Hack* to its parent. A receiver may also decide to send an acknowledgement after a certain delay *ThackMax*. The situation depicted by HMSC of Figure 4 shows communications between a node and two children. Child$_1$ always sends an acknowledgement to its parent upon data reception, while Child$_2$ always acknowledges data packets after the delay expiration. Furthermore, packets are never missed, but Child$_1$ can receive corrupted data, and retransmission and *Hack* packets may cross. The left part of Figure 4 shows

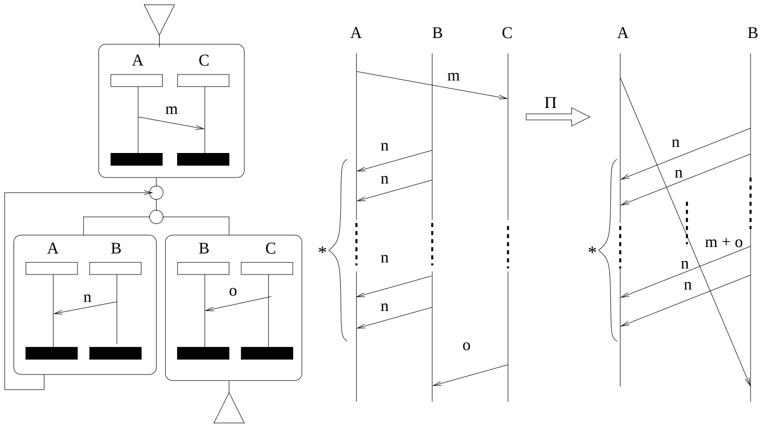


Fig. 4. a) HMSC generating an unbounded crossing b) MSC c) pMSC

the partial order associated with $(\mathit{Hack\_incomplete} \circ \mathit{Crossing})^{\omega}$. The right part of Figure 4 shows the same order after hiding the *Parent*, and all timer events. 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 order to represent pHMSCs in a more practical way, e.g. for understanding easily the scenarios they describe or for model-checking algorithms, we can ask whether a graphical representation can be used. We can compare them with a notation that is more expressive than HMSCs, such as *compositional HMSCs* (cHMSC) [6]. Since pHMSCs involve in general multitype events we actually need enriched versions of MSCs and cMSCs, by allowing multitype events. For simplicity we will assume in this section that the projections do not generate multitype events. However, our constructions can be easily adapted to this case.

From the examples in Section 4 we know that pHMSCs are more expressive than HMSCs, e.g. pHMSCs allow to specify an unbounded crossing of messages like in Figure 4. Actually, this behavior can be described using the cHMSC notation. The aim of this section is to compare the expressivity of pHMSCs and cHMSCs.

We first observe that cHMSCs are not weaker than 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 5 below gives a rough idea of what kind of behavior of cHMSCs cannot be expressed as the projection of an HMSC. Intuitively, the cHMSC in the left part of Figure 5 describes two or more isolated send events (denoted α), that are

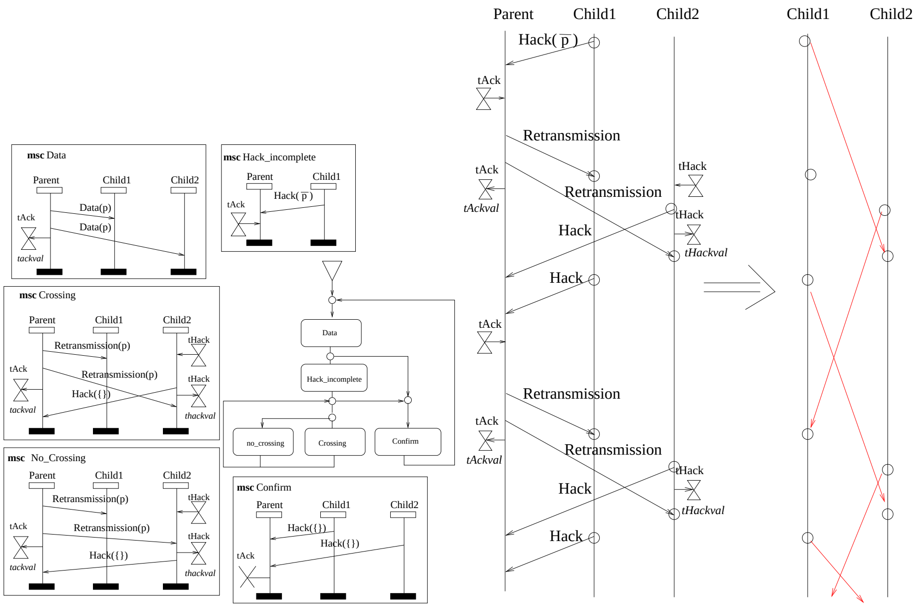


Fig. 5. A part of RMTP2 protocol and Crown generated by hiding instance Parent

matched after an arbitrary number of β messages. If we want to describe this using a pHMSC, then we need for each event α a fresh process, that disappears through projection (see middle part of Figure 5). More formally, for matching a sequence $e_1 < \ldots < e_n$ of sends on process A with a sequence $f_1 < \ldots < f_n$ of receives on process B, we need for each pair e_i, f_i a fresh process C_i (that disappears through projection). 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. Therefore, since the number of processes is fixed we cannot describe an arbitrary number of isolated send events α crossing the β messages by pHMSCs, whereas this can be done with a cHMSC.

5.1 pHMSCs compared to cHMSCs

We show now that we can transform each pHMSC (without multitype events) into an equivalent realizable cHMSC. We recall that a cHMSC is 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 *unerased* event.

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 new type of some events $e \in M'$ (send/receive/local, to which process etc). For example, in Figure 4 we have to guess that the first event on Child_1 will be a send to Child_2. In order

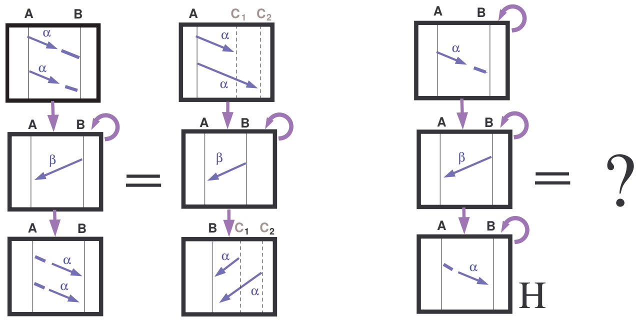


Fig. 6. how to do such a cHMSC with a pHMSC?

to verify whether the guess is correct, we need to keep track of the processes occurring in the future of e within M. Among these processes we need also to know which processes (called *dead* processes) are seen by an unerased event $e' > e$. Hence, let us define for each unerased event $e \in E'$:

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

An unerased event $e \in E'$ is called *unchecked* if $\mathrm{LiveF}(e) \neq \emptyset$. Intuitively, when an event e is created, it is unchecked. It becomes checked as soon as we have a proof that it can match no more event after M. 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 we stop the current path. Let $e < e'$ with e, e' not erased and $p \in F(e')$. Assume by contradiction that e is matched by some unerased event f after M with $P(f) = p$. We have that $e < e' < f$, hence f cannot be an immediate successor of e in the projection ie it does not match e, a contradiction.

The set of unchecked events of 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 ρ be an initial path of an HMSC H over $\wp$ processes, and M' the pMSC defined by projecting ρ. Then $|ToCheck(M')| \leq \wp^2$.*

Proof. Assume by contradiction that M' contains more than $\wp^2$ unchecked events. Then we can find two events $e < e'$ in E' with $P(e) = P(e')$ such that $\mathrm{F}(e) = \mathrm{F}(e')$, thus $\mathrm{LiveF}(e) = \emptyset$. The reason is that for any $e < e'$ with $P(e) = P(e')$, we have $\emptyset \neq \mathrm{F}(e) \subseteq \mathrm{F}(e') \subseteq \mathcal{P}$. Hence e cannot be unchecked. $\square$

We are now ready to show how to construct for each pHMSC H' with no multitype events an equivalent cHMSC G. The nodes of G are associated to initial paths $\rho = (v_0 \to v_1 \to \cdots \to v_k)$ of H. A priori, the number of paths is infinite. But with the bound on the unchecked events, we show that at most $2^{\wp^3}$ states are needed. For each path ρ we guess the type of unchecked events: we create a node $[v, U, to, \mathrm{LiveF}, \mathrm{DeadF}]$ for each mapping $to : U \to \mathcal{P}$, with the following semantics:

1. $v = v_k$ is the last node on ρ,
2. $U = \mathrm{ToCheck}(M')$ is the set of unchecked events of the cMSC M' labeling the projection of the path ρ,
3. LiveF (DeadF, resp.) is a mapping from U to the set $2^{\mathcal{P}}$, denoting the set of live (dead, resp.) processes in the future of the unchecked event $e \in U$,
4. If $to(e) \neq P(e)$, then e is supposed to be an isolated send to process $to(e)$. Otherwise, e is supposed to be a local event.

A node $x = [v, U, to, \mathrm{LiveF}, \mathrm{DeadF}]$ is labeled by a cMSC M' defined on the unerased events of the MSC labeling node v. It remains to define the type of events of M'. First, each unchecked $e \in U$ is either an isolated send to $to(e)$ (if $to(e) \neq P(e)$) or a local event, otherwise. We determine then all pairs of unerased events $(e, f) \in E' \times E'$ such that $P(e) \neq P(f)$ and $e \lessdot_{M'} f$. Such pairs are messages. Any event $f \in E'$ s.t. $e \lessdot f$ for some event $e \notin M'$ of ρ is an isolated receive from process $P(e)$. Remaining events are local events in M'.

We assume that U gives the ordering of unchecked events (according to the path ρ). We define a set of initial nodes x_0 associated with the path $p = (v_0)$ consisting of the initial state, only. The final states $x = [v, U, to, \mathrm{LiveF}, \mathrm{DeadF}]$ are associated with paths ending in a final state v of H and are required to satisfy $to(e) = P(e)$ for all e, that is all remaining unchecked events are local events.

It remains to define the transitions between nodes of the cHMSC G. Suppose that x, y are nodes of G associated with paths ρ and ρ', resp., where ρ' is obtained from ρ by extending it by one node, say v. Let also u be the last node on ρ. Thus, $x = [u, U, to, \mathrm{LiveF}, \mathrm{DeadF}]$ and $y = [v, V, to', \mathrm{LiveF}', \mathrm{DeadF}']$. We just need to determine which transitions $x \to y$ are allowed in G, with respect to the mapping to:

1. Let $e \in U$ be such that $to(e) = P(e)$. That is, we guessed that e has to be a local event. Then we require that no unerased event f in v matches e.
2. Let $e \in U$ be such that $to(e) = q \neq P(e)$. That is, e is supposed to be an (unmatched) send to process q. Then we require that no unerased event f in v with $P(f) \neq q$, matches e in H'. Additionally, we ask that $q \notin \mathrm{DeadF}'(e)$.

If one of these conditions is false, then the guess was wrong, and the state cannot access a final state. We can delete such states. Notice that an unchecked event $e \in U$ is matched in H' by an unerased event f of v if all conditions below are satisfied (where $\leq_v$ is the visual order of the MSC labeling node v):

- $g \leq_v f$ for some event g in v with $P(g) \in \mathrm{LiveF}(e)$, and
- $g \leq_v f$ for no event g in v with $P(g) \in \mathrm{DeadF}(e)$ (i.e. $e < e' < g \leq f$ for some unerased event e' in v), and

– $g \leq_v f$ for no unerased event g in v, with $P(g) \in F(e) = \text{DeadF}(e) \cup \text{LiveF}(e)$ (i.e. $e < g \leq f$).

We recall that realizable cHMSCs are cHMSCs where every accepting path is labeled by an MSC, i.e., there is no unmatched event. The previous construction yields:

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 $n2^{O(\wp^3)}$.*

Remark 1 *We can extend our construction in order to check that the given pHMSC has no multitype events. Moreover, our construction can be easily modified in the presence of multitype events, for obtaining an extended realizable cHMSC.*

Realizable cHMSCs have the property that on any initial path, the number of unmatched sends is linear in the size of the cHMSC:

Definition 2 *Let b be an integer. A cHMSC is called* b-bounded *if for each initial path ρ, each prefix of ρ is labeled by a cMSC with at most b unmatched sends.*

Proposition 2 *[6] A realizable cHMSC G is b-bounded, where b is the size (= total number of events) of G.*

We can notice that the cHMSC obtained in Theorem 1 is realizable. Moreover, it enjoys the property that the number of unmatched sends is polynomial in the size of H (see Lemma 1). We can restate Lemma 1 as follows:

Proposition 3 *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.*

We can now state the main result of this section:

Theorem 2 *The class of realizable cHMSCs coincides with the class of projections of HMSCs (that contain no multitype events).*

Proof. Let $\mathcal{P}$ be the set of processes occurring in a given realizable cHMSC G and let b be the size of G. Using Proposition 2 we know that every initial path of G is b-bounded. More precisely, since each accepting path of G is labeled by an MSC, every loop has as many isolated sends as isolated receives on a given channel. In addition, with each event e that is either a send or a receive from p to q we can associate an integer $b(e) \leq b$ such that on each path $v_0 \to v_1 \to \cdots \to v_n$ from the initial state to the node v_n containing e, there are $b(e)$ unmatched sends from p to q before e (excluding e).

We define an HMSC H over processes from $\mathcal{P} \cup \{(p, q, k) \mid p, q \in \mathcal{P}, 1 \leq k \leq b\}$, together with a projection $\pi(H)$ such that $\pi(H)$ is equivalent to G. The projection will erase all events of the auxiliary processes (p, q, k). The nodes of H will be in one-to-one correspondence with the nodes of G. Each cMSC M labeling a node of G is transformed into an MSC as follows. Each isolated send $e \in M$ from p to q is replaced by a message from p to the additional process $(p, q, b(e) + 1)$. Suppose now that $e \in M$ is an isolated receive from p to q. We replace e by a sequence of messages, from $(p, q, 1)$ to q, then from $(p, q, 2)$ to $(p, q, 1)$, and so on, up to one from $(p, q, b(e))$ to $(p, q, b(e) - 1)$. Note that in both cases, the isolated event e corresponds to the event on the original process (p, resp. q). Hence we just have to show that messages are the same. Let e be an unmatched send of G. After having received exactly $b(e)$ messages in G on q from p, the event $e \in H$ sees the process $(p, q, 1)$. Then when another receive occurs in G, there is a message in H from $(p, q, 1)$ to the receive f on q, hence $e \lessdot f \in \pi(H)$, which yields a message in the projection. □

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 $\mathrm{Gen}(G)$ of *generators* of G defined as follows. An *atomic* MSC M belongs to $\mathrm{Gen}(G)$ 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 it is necessary for $\mathrm{Gen}(G)$ to be finite. At the end of the section we show that if $\mathrm{Gen}(G)$ 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 $\mathrm{Gen}(G)$ and such that for every $M \in \mathrm{Gen}(G)$, 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 $M \in \mathrm{Gen}(G)$ and extracts the events of M from w.

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 $\mathrm{CG}(M)$ 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 4 *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 $\mathrm{CG}(M)$.

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 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. By Proposition 2, 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 $(x, p) \in X \times \{1, \dots, \wp\}$ we record an integer $T(x, p) \in \{0, 1, 2\}$ telling whether x sees the first event in M $(k = 1)$ on p $(T(x, p) = 2)$, or whether x sees last_p but not the first event in M on p $(T(x, p) = 1)$, or whether it sees no event in M on p $(T(x, p) = 0)$. 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 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 $\mathcal{A}(G)$.*

Proof. For sake of simplicity we will assume that the cHMSC G is given as an equivalent finite automaton $\mathcal{B} = (V, \to, I, F)$, with transitions labeled by events from the set E, $(\to \subseteq V \times E \times V)$, and $I, F \subseteq V$ as sets of initial and final states. The automaton recognizes at least one representative linearisation for each $M \in \mathcal{L}(G)$.This automaton can be obtained very easily by linearising the cMSCs labelling nodes of G one by one.

We define now the finite automaton $\mathcal{A}(G)$. Each state w of $\mathcal{A}(G)$ is of the form $[v, T, S, k, X]$, where $v \in V$ is a state of $\mathcal{B}$, $X = \{\mathrm{last}_p \mid 1 \leq p \leq \wp\} \cup \mathcal{S}$ is the set of at most $\wp + b$ events defined above, T, S, k are the vision function, unmatched send and type functions. Note that when x first sees y, we can update T', S' by a call $\mathrm{Update}(x, y)$, that sets $T'(x, p) = \max((T'(y, p), T'(x, p))$ for each p and $S'(x) = S'(x) \cup S'(y)$.

We define now the transitions of $\mathcal{A}(G)$ as follows. Assume that $v \xrightarrow{e} v'$ is a transition of $\mathcal{B}$ and let $p = P(e)$. We let $[v, T, S, k, X] \xrightarrow{\ell} [v', T', S', k', X']$, with T', S', k', X', obtained through the following algorithm:

1. Define X' using X and e ($\text{last}_p = e$ and update $\mathcal{S}$).
2. Guess a value $k'(\text{last}_p) \geq k(\text{last}_p)$. If $k'(\text{last}_p) = 1$, then $e \in M$, so the transition is labeled by $\ell = e$. Else $\ell = \epsilon$ and we stop.
3. For all $x \neq \text{last}_p$, set $T'(x) = T(x)$.
4. Set $T'(\text{last}_p, q) = 0$ for $q \neq p$, and $S'(\text{last}_p) = \emptyset$. Then, set $T'(\text{last}_p, p) = 2$ if $k'(\text{last}_p) = 0$ (i.e., e is the first event in M on process p). Else $T'(\text{last}_p, p) = 1$.
5. If e is a receive event, then let $s \in \mathcal{S}$ be the associated send.
   (a) Check that $k'(\text{last}_p) = k(s)$.
   (b) Update(e, s).
   (c) For each $x \in X$ such that either $T'(x, p) \neq 0$ or $s \in S'(x)$ (i.e., x sees either e or s), Update(x, e) .

Moreover, a state $[v, T, S, k, X]$ is initial if $v \in I$, $k(x) = T(x, p) = 0$ for all $x \in X$, $p \in \mathcal{P}$ and $\mathcal{S} = \emptyset$ (there is no unmatched send). A state $[v, T, S, k, X]$ is final if $\mathcal{S} = \emptyset$ (there are no unmatched sends) and $T(\text{last}_p, q) = 2$ for all $p, q \in \mathcal{P}$ (thus, the last event on p sees the first event on q for all p, q).

□

Proof. Let us show by induction over the number of steps that after one step of the algorithm, $T(x, i) = 1$ iff x can reach last_i and $T(x, i) = 2$ iff x can reach the first event of process i in CG and $s \in S(x)$ iff x can reach the unmatched send s in CG.

Assume that x can reach last_p in CG at step n of the algorithm. If the event x has just been created, then it can be on event of p and we are done. Else, if it is a send, since it was just created, the receive has not yet be seen, so the event sees nothing. So x must be a receive. The only edge that go from it leads to its associated send. So the send should reach in CG one event of p, (that is not x). So we can apply the induction hypothesis to the system of the step $n-1$ (without x) and conclude with rule 5 of the algorithm: $T'(x, p)$.

Else, event x was not just created. If x saw an event of p before, then it still sees one event of p after since all the rules never decreament a counter. Else it is because the just created event y is needed to go from x to any events of p. If y is a send, nothing happens. So y is a receive. Either x saw the send event s associated with y, and then y should be on p. By induction hypothesis, x saw s thus $s \in S(x)$. By the rule 5, after the last step, $T'(x, p) = max(T'(x, p), T'(y, p)) = 1$. Or x did not see s, and s saw an event of p. Moreover, at the end of the step, x will see y, that is it saw an event of $P(y)$. By induction hypothesis, we know that $T(x, P(y)) \neq 0$ and $T(s, p) \neq 0$, so by rule 5, $T(x, p) = max(T(x, p), max(T(s, p), T'(y, p)) = 1$, CQFD.

We can do the same for the two other cases (first event and unmatched send event).

Conversely, assume that $s \in S'(x)$, $P(s) = p$. If event x has just been created, it can be s and we are done. Else, it is a receive and the send event associated u'

satisfies $s \in S(u)$. Notice that $s \neq u$ since u will no more be an unmatched send at the end of the step and all its entries will be deleted. So if we apply induction hypothesis on previous step, since $s \in S(u)$, the event u sees s. So its associated receive, x, sees also s and we are done.

Else, event x was not just created. If x saw s before, then it still sees s since the event just seen cannot be the receive associated else s would not be in $S'(x)$. □

Remark 2 *We cannot apply this construction to non-realizable cHMSCs, since we do not have any bound on the number of unmatched sends.*

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.*

Proof. Note that the automaton $\mathcal{A}(G)$ accepting Gen(G) is exponential in the size of G, if G is a realizable cHMSC. Moreover, note that if G is the cHMSC obtained from a pHMSC $H' = \pi(H)$, then $\mathcal{A}(G)$ is still exponential in the size of H. The reason is that G is b-bounded, where b is polynomial in the size of H (Proposition 3). Therefore, checking whether $\mathcal{A}(G)$ accepts a finite language can be done in PSPACE.

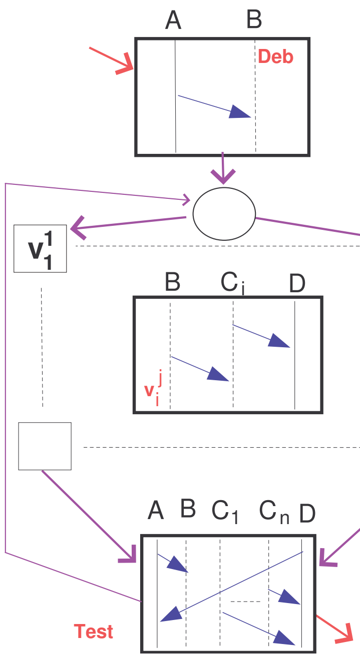
For the co-NP-hardness we give a reduction from the complement of the Hamiltonian path problem. A given graph $G = (V, E)$ with n vertices has no hamiltonian path if and only if every path of length n contains some node at least twice.

We build a pHMSC H' from the graph G as in the Figure 6.1, with v_i^j labeled by some pMSC M_i described below. For generating the paths of length n we use the usual trick: we have n copies of G, with n^2 nodes (v_i^j). The transition relation of H' is given as $v_i^j \to v_{i'}^{j'}$ iff $v_i \to_G v_{i'}$ and $j' = j + 1$. Additionally, we need an initial node Deb and a final node Test that can return to any node at the beginning of the graph: $v_i^j \to$ Test iff $j = n$ and Test $\to v_i^j$ iff $j = 1$.

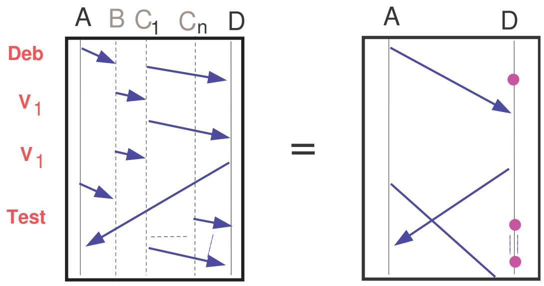
The figure also describes the pMSC labeling the nodes of H'. The projection erases all events of $B, (C_i)$, the only events remaining are those on A and D. The idea is to create an order between process A and D only if some node v_i is seen twice (here, v_1 is seen twice).

Hence, if there is an hamiltonian path in G then by iterating the corresponding loop in H' we obtain atoms of arbitrary large size (corresponding to a crown).

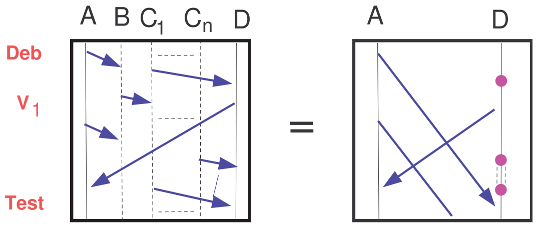
Else, if G has no hamiltonian path then the atoms of H' are either messages from A to D, or two crossing messages from A to D and from D to A. Thus Gen(H') is finite. □



A
B
Deb
v_1^1
B
C_j
D
v_i^j
v_n^n
A B C_1 C_n D
Test



A B C_1 C_n D
Deb
V 1
V 1
Test
=
A
D



A B C_1 C_n D
Deb
V 1
Test
=
A
D

6.2 Construction of HMSCs from realizable cHMSCs

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, according to Proposition 2.

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 (ie a *subpath*), 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. Since we keep track of path segments by recording only the first/last node of each segment (and the processes occurring in the segment), all we need to show is that the number of segments can be bounded.

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

Theorem 5 *Let G be a cHMSC where $Gen(G)$ is finite. Then we can construct an equivalent HMSC H from G of size $O(|G|^{(2+\wp)\cdot(b\cdot maxsize)})$.*

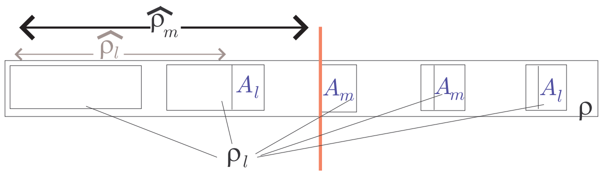
Proof. If M is recognized by H, then there exists a full, accepting path in G that recognizes M. Conversely, let us take an MSC $M = A_1 \cdots A_n$, where each A_i belongs to $\mathrm{Gen}(G)$. This MSC labels an accepting path $\rho = v_1 \to \cdots \to v_k$ of G.

For notational 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 ρ comes before the first event of A_j in ρ. That is, we choose an ordering of the atoms of M according to their first occurrence in ρ.

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

Hence $(A_1, \rho_1) \cdots (A_n, \rho_n)$ is an accepting path of H labeled by M, and the proof is done.

proof of the claim: We denote by $\widehat{\rho_j}$ the longest prefix of ρ 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 ρ_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 intesection 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 $b'' \cdot (maxsize - 1)$ events in $\widehat{\rho_m} \setminus \rho_l$, thus at most $b'' \cdot (maxsize - 1) + 1$ 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 $(b'+1) \cdot maxsize$ events in $\rho_l \setminus \widehat{\rho_m}$, hence at most $(b'+1) \cdot maxsize$ segments. Therefore, we conclude that ρ_l contains at most $(b' + b'' + 1) \cdot maxsize = (b+1) \cdot maxsize$ segments.

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

Example: Let G be a cHMSC with 4 states v_1, v_2, v_3, v_4, labeled respectively by s, s, r, r where s is a send on process 1 coresponding to a receive r on process 2. There are edges from v_1 to v_2, v_2 to v_3, v_3 to v_2, and v_3 to v_4. 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 $[v_1, \{1\}]; [v_3, \{2\}]$, consisting of two segments, one being a path from v_1 to v_1 and the other from v_3 to v_3,
– s_2 is the state $[v_1, \{1\}]; [v_4, \{2\}]$),
– s_3 is the state $([v_1, v_3, \{1, 2\}]; [v_3, \{2\}])$ consisting of two segments, one being a path from v_1 to v_3, the other the path from v_3 to v_3,
– s_4 is the (final) state $([v_1, v_4, \{1, 2\}])$.

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 $[v_4, \{2\}]$ 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 coresponds to the guess of a scattered subpath $[v_2, \{1\}]; [v_3, \{2\}]$ labeled by A, and to the guess $[v_2, \{1\}]$ making the connection between ($[v_1, v_3, \{1, 2\}]$ and $[v_3, \{2\}]$), while $[v_3, \{2\}]$ is a disjoint segment.

Since there are at most $2^{maxsize}$ 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 HMSC against HMSC

Validating HMSCs specifications allows detectection 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 are obtained for *bounded* HMSCs [2, 13], for which the set of MSC-linearizations is regular. The most general class ensuring decidability is given by globally cooperative HMSCs [4]. Roughly speaking, 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 $P(M_1) \cap P(M_2) = \emptyset$, i.e., M_1, M_2 are 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.

If we are lucky enough to build a globally cooperative HMSC from a given pHMSC, then we can do model-checking using the known algorithms [4] against any HMSC property. The next theorem shows that model-checking remains decidable, if the pHMSC is arbitrary, but the HMSC property is globally cooperative (which is a reasonable assumption).

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 $\mathcal{L}(\pi_E(H)) \subseteq \mathcal{L}(G)$.*

Proof. This result will follow from [4], since we can build an automaton $\mathcal{B}_G$ of exponential size recognizing any sequence $A_1 \cdots A_m$ of atomic MSCs such that $A_1 \circ \cdots \circ A_m \in \mathcal{L}(G)$. For $\pi(H)$ we build first an equivalent cHMSC H' as in theorem 1. To H' we apply the algorithm of section 6, but using the atoms of G instead of those of Gen(H') (that is possibly infinite). The size of the HMSC H'' we obtain is simply exponential in the size of H and G. Checking if the intersection of $\mathcal{B}_G$ and H'' is non-empty is equivalent to $\mathcal{L}(G) \cap \mathcal{L}(\pi(H)) \neq \emptyset$. Since this test is logaritmic in the size of $\mathcal{B}_G, H''$ this gives the PSPACE bound. The EXPSPACE bound for the inclusion is obtained with the same argument. □

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 $\mathcal{L}(\pi_{E'}(H)) \subseteq \mathcal{L}(\pi_E(G))$.*

Proof. We can apply the algorithm of [13, 2], getting an automaton $\mathcal{A}$ of exponential size that recognizes all linearisations of $\mathcal{L}(G)$. We have that $\pi_E(\mathcal{A})$ recognizes all linearisations of $\pi_E(\mathcal{L}(G))$, since the projection preserves the causalities. Then we just check the emptiness of the intersection of $\pi_{E'}(H)$ and $\pi_E(\mathcal{A})$. This is done in logarithmic space (w.r.t. the size of $\mathcal{A}$ and H), hence in PSPACE in the size of G, H. The same arguments apply to the inclusion $\mathcal{L}(\pi_{E'}(H)) \subseteq \mathcal{L}(\pi_E(G))$. □

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.

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 mesage 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. ounterexample-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 *ICCAD*, 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.
17. R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal of Computing*, 1(2), 1992.