Compositional Message Sequence Charts (CMSCs) are better to Implement than MSCs.

Blaise Genest

LIAFA, Université Paris VII, 2 place Jussieu, 75251 Paris, France
& Departement of Computer Science, Warwick, Coventry, CV4 7AL, UK

Abstract. Communicating Finite States Machines (CFMs) and Message Sequence Graphs (MSC-graphs for short) are two popular specification formalisms for communicating systems. MSC-graphs capture requirements (scenarios), hence they are the starting point of the design process. Implementing an MSC-graph means obtaining an equivalent *deadlock-free* CFM, since CFMs correspond to distributed message-passing algorithms. Several partial answers for the implementation have been proposed. E.g., local-choice MSC-graphs form a subclass of deadlock-free CFM: Testing equivalence with some local-choice MSC-graph is thus a partial answer to the implementation problem. Using Compositional MSCs, we propose a new algorithm which captures more implementable models than with MSCs. Furthermore, the size of the implementation is reduced by one exponential.

1 Introduction

Specifying the behavior of software systems in such a way that formal methods can be applied and validation tasks can be automated, is a challenging goal. While research has brought strong results and tools for simple systems, complex systems still lack powerful techniques. For instance, concurrent systems such as message passing systems are still hard to cope with.

Concurrent languages such as Harel's Live Sequence Charts [11], UML sequence diagrams, interworkings..., have seen a growing interest this last decade. Among them, the ITU visual notation of *Message Sequence Charts* (*MSCs*, [14]) has received a lot of attention, both in the area of formal methods and in automatic verification [2, 13, 19, 18, 20]. MSCs can be considered as an abstract representation of communications between asynchronous processes. They are used as requirements, documentations, abstract test cases, and so on. *MSC-graphs* propose a way of modeling set of behaviors, combining parallel composition (processes) with sequential composition (transition system). The main advantage of such a visual representation is to have a local, explicit description of the communication and the causalities appearing in the system. On the other hand, SDL (ITU norm Z100) brings another formalism, namely *Communicating Finite States Machines* (*CFM* for short) [5]. Being really close to distributed algorithms, CFMs are the ideal model for modelling parallel programs. The absence of deadlock is crucial for communication protocols, where any blocking execution means a failure in the system. That is, any concurrent system which has to be implemented must be turned into a deadlock-free CFM. Hence, we will consider only deadlock-free CFM implementations in this paper.

Our aim is to give a heuristic to implement MSC-graphs. That is, if a model passes the test, then it is implementable and we can provide an implementation. However, if it fails the test, it may be the case that it is implementable anyway. The important point is to understand which implementable systems are captured with this algorithm, the more the better. Our test captures every model which is equivalent to some *local-choice CMSC-graph*. Every local-choice CMSC-graph is implementable, for its control is local (as for CFM), in contrast with the usual global control of MSC-graphs.

Implementation of MSC-graphs is a non trivial task, which has yet no definitive answer. The first implementation test, which was proposed by [1], captures a subclass of MSC-graphs which are equivalent to some deadlock-free CFM without adding control data to messages. This test covers only a subclass of the very restricted regular MSC-graphs. It was further extended to capture a subclass of globally-cooperative MSC-graphs [16], with the same EXPSPACE-complete complexity. With this same restriction of disallowing additional data, [12] characterizes the subclass of local-choice MSC-graph which is implementable. Since data parts are usually abstracted away in an MSC-graph specification, this restriction prevents many useful models from being implementable. For instance, as soon as we add data to messages, any local-choice MSC-graph is implementable [9]. The first paper to consider additional data was [13], giving the exact expressivity of a subclass of (not deadlock-free) CFMs in terms of MSC-graphs. [4] characterizes the expressivity of deadlock-free CFMs in terms of MSC-graphs, but no complete algorithm is provided. At last, an internal report [7] gives a PSPACE algorithm to test implementation into local-choice MSC-graph (thus into deadlock-free CFM), yielding an implementation of doubly exponential size.

In this paper, we extend the results of [7] in order to improve the complexity of the test to co-NP, yielding an implementation of single exponential size. To achieve this goal, we use local-choice Compositional MSC-graphs (CMSC-graphs). The implementation of this class follows easily from [9]. Using compositional MSCs instead of MSCs allows to capture more formalisms. CMSC-graphs were introduced by [10] to get rid of the finite generation restriction of MSC-graphs. Later, safe CMSC-graphs were shown to be model-checkable against MSO [18], temporal logic [8], and globally cooperative CMSC-graphs [6]. The important property used by these algorithms is that the events of any generated MSC can be scheduled using bounded communication channels, for a fixed bound (see section 3). Since nodes of a compositional MSC-graph need not be labeled by complete MSCs (there can be unmatched sends and receives), the time consuming test of [7] becomes irrelevant: Not only we show that we can still test whether a CMSC-graph is equivalent to some local-choice CMSC-graph (with a new algorithm), but the complexity is better than for local-choice MSC-graphs (co-NP-complete vs PSPACE), improving the implementation size by one exponent, and thus making the test practical.

Related work: A new formalism, Triggered MSCs [22], was designed from the ground for the implementation purpose. It makes implementation easier than for MSC-graphs, but model-checking has not been studied yet. Also, Live Sequence Charts [11] use a different semantics to obtain implementability.

2 Message Sequence Charts (MSCs)

Message Sequence Charts (MSC for short) is a scenario language standardized by the ITU ([14]). 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 can also be represented.

Definition 1 *[10] A* compositional *MSC (CMSC) is a tuple $M = \langle \mathcal{P}, E, \mathcal{C}, t, m, < \rangle$ where:*

- *$\mathcal{P}$ is a finite set of processes,*
- *E_p is a finite set of events on process p, with $E = \bigcup_{p \in \mathcal{P}} E_p$*
- *$\mathcal{C}$ is a finite set of names for messages and local actions,*
- *$t : E \to \mathcal{T} = \{p!q(a), p?q(a), p(a) \mid p \neq q \in \mathcal{P}, a \in \mathcal{C}\}$ labels an event with its* type*: either a send $p!q(a)$ of message a on process p to q, a receive $p?q(a)$ on p from q, or a local event $p(a)$. We partition $E = S \dot\bigcup R \dot\bigcup L$ into sends, receives and local events.*
- *$m : S \to R$ is a partial and injective function matching a send to its corresponding receive. If $m(s) = r$, then $t(s) = p!q(a)$ and $t(r) = p?q(a)$ for some $p, q \in \mathcal{P}, a \in \mathcal{C}$.*
- *$< \subseteq E \times E$ is an acyclic relation between events consisting of:*
  - a total order on E_p, for every process $p \in \mathcal{P}$, and
  - $s < r$, whenever $m(s) = r$.

An MSC is a CMSC where the message function m is a total function that is one-to-one.

The event labeling t implicitly defines the process $P(e)$ for each event e: $P(e) = p$ if $t(e) \in \{p!q(a), p?q(a), p(a)\}$ for some $q \in \mathcal{P}, a \in \mathcal{C}$. We denote any pair $(p, q) \in \mathcal{P}^2$ of distinct processes as a channel. We assume that channels are FIFO, i.e., there is no overtaking on messages sent on the same channel.

The relation $<$ is called the *visual* order of the CMSC, since it corresponds to its graphical representation. It is comprised of the process ordering and the message ordering. Since $<$ is required to be acyclic, its reflexive-transitive closure $<^*$ is a partial order on the set E of events, which we will denote by $\leq$. An extension of $\leq$ to a total order is called a *linearization* of M. We consider labeled linearizations $t(e_1) \cdots t(e_n)$, with $e_1 \cdots e_n$ a linearization on events. One can understand any linearization as some particular execution of the CMSC. Notice that because of the FIFO condition, one can retrieve an MSC from any of its linearizations.

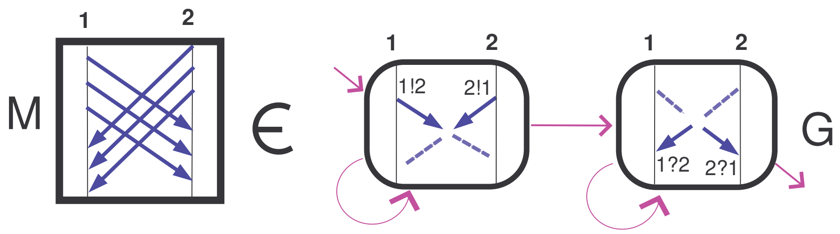

Fig. 1. The left part of the figure depicts an MSC scenario M.
— The two squares on the right are CMSCs involving actions $1!2, 2!1, 1?2, 2?1$.

Definition 2 *[17] We say that a linearization $t(e_1)\cdots t(e_n)$ is* b-bounded *if, for each channel (p,q), the difference between the number of sends $p!q$ and the number of receives $q?p$ in any prefix $t(e_1)\cdots t(e_i)$ is at most b, for any i. We say that an MSC M is* existentially *(respectively* universally*)* b-bounded *if some (resp. every) linearization of M is b-bounded .*

MSCs specify only finite behaviors. For describing sets of behaviors, we use MSC-graphs, which are the basic fragment of the High-level MSCs of the norm [14] (the norm allows hierarchy that we do not take into account here) that are just transition systems with nodes labeled by MSCs. They were extended to Compositional MSC-graphs (CMSC-graphs), where nodes are labeled by CMSCs [10].

Definition 3 *A* CMSC-graph *is a labeled transition system $G = (V, \to, v^0, F, \lambda)$ with set of nodes V, transition relation $\to \in V \times V$, initial node v^0 and set of final nodes F. Each node v is labeled by a CMSC $\lambda(v)$. An accepting path of G is defined as a sequence of transitions $\rho = (v_1 \to v_2 \cdots \to v_k)$ with $v_1 = v^0$ and $v_k \in F$.*

The composition of two CMSCs is the set of CMSCs defined by gluing together the processes axis, and extending the messages functions in any way such that the FIFO condition is preserved.

Definition 4 *A composition of CMSCs $M_1, \cdots, M_n$, where $M_i = \langle \mathcal{P}, E_i, A_i, t_i, m_i, <_i \rangle$ is a CMSC $M = \langle \mathcal{P}, E = \bigcup_i E_i, \mathcal{A} = \bigcup \mathcal{A}_i, t = \bigcup t_i, m, < \rangle$ such that:*

- *The message function m extends each m_i and it is required that m preserves the FIFO restriction on matched events. That is if $m(s) = r$ and $m(s') = r'$ are two messages from p to q and $s <_p s'$ then $r <_q r'$.*
- *The visual order $<$ is the union of $<_i$ and the set of (e, f) with $m(e) = f$ or $P(e) = P(f)$ and $e \in E_i, f \in E_j$, $j > i$.*

Notice that a sequence of given CMSCs $M_1, \ldots, M_n$ can admit several compositions. However, there can be at most one such composition that is an MSC, since the k-th send from p to q of an MSC is matched by the k-th receive on q from p. If it exists, we denote by $M_1 \cdots M_n$ the MSC which is a composition of $M_1, \cdots, M_n$. For instance, there exists only one MSC which is a composition of CMSCs seen along the path that loops three times around each node of figure 1. This MSC is depicted in the left part of figure 1. Since we will consider only the MSCs generated by a CMSC-graph, we are only interested in this unique composition, if it exists.

Definition 5 *Let G be a CMSC-graph. The language of G is $\mathcal{L}(G) = \{\lambda(v_1) \cdots \lambda(v_k) \in \mathbb{MSC} \mid v_1 \cdots v_k$ is an accepting path of $G\}$, where $\mathbb{MSC}$ is the set of all MSCs.*

An MSC-graph is a CMSC-graph whose nodes are labeled by MSCs. Let G be an MSC-graph. The MSC-graph G is *finitely-MSC-generated*, that is any MSC generated by G is the composition of MSCs labeling the nodes of G. Hence, any $M \in \mathcal{L}(G)$ is existentially b-bounded, where b is the size of the largest MSC in the set L of MSCs labeling the nodes of G. We call G an *$\exists$-b-bounded* MSC-graph.

The right part of figure 1 depicts a CMSC-graph which is not existentially bounded, since iterating n times both loops gives an MSC which is not universally $n-1$ bounded. In figure 2, we denote s for the send from host to function, and r for the receive on function from host. Then iterating n times the leftmost loop in figure 2 yields an MSC having the 1-bounded linearization $(sr)^{n+1}$, and having also the linearization $s^{n+1}r^{n+1}$ which is not $n-1$-bounded. That is, this MSC-graph is existentially 1-bounded, but it is not universally bounded.

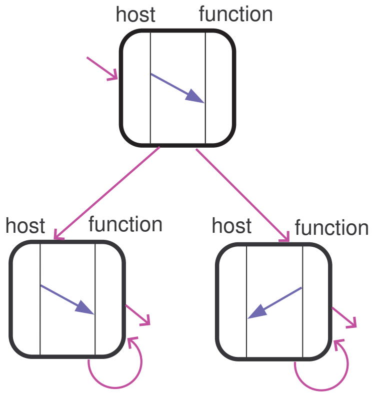

Fig. 2. MSC-graph depicting the isochronous transactions of usb 1.1

The size $|M|$ of a (C)MSC M is its number of events. The size $|G|$ of a (C)MSC-graph G is the sum of the sizes $|M|$ of the CMSCs labeling its nodes. The size of $\mathcal{P}$ is the number $\wp$ of processes.

A *Communicating finite-state machine* (CFM) $\mathcal{A} = (\mathcal{A}_p)_{p \in \mathcal{P}}$ [5] consists of finite automata $\mathcal{A}_p$ associated with processes $p \in \mathcal{P}$, that communicate over unbounded, error-free, FIFO channels. The content of a channel is a word over a finite alphabet $\mathcal{C}$. With each pair $(p, q) \in \mathcal{P}^2$ of distinct processes we associate a channel $C_{p,q}$. Each $\mathcal{A}_p$ is described by a tuple $\mathcal{A}_p = (S_p, A_p, \rightarrow_p, F_p)$ consisting of a set of local states S_p, a set of actions A_p, a set of local final states F_p and a transition relation $\rightarrow_p \subseteq S_p \times A_p \times S_p$. The computation begins in an initial state $s^0 \in \prod_{p \in \mathcal{P}} S_p$. The actions of $\mathcal{A}_p$ are either local actions or sending/receiving a message. We use the same notations as for MSCs. Sending message $p!q(a)$ means that a is appended to the channel $C_{p,q}$. Receiving message $p?q(a)$ means that a must be the first message in $C_{q,p}$, which will be then removed from $C_{q,p}$. A local action a on process p is denoted by $l_p(a)$. A run of a CFM is a linearization x such that the projection of x on process p is a run of A_p, for all p. We denote a run of the CFM as *successful*, if each process p can finish the run in some final state of F_p, and all channel are empty. The set of successful runs generated by $\mathcal{A}$ is denoted $L(\mathcal{A})$. It is easy to notice that if x is an accepting run of the MSC M, then any linearization of M is also a successful run. We will denote by $\mathcal{L}(\mathcal{A})$ the MSC-language of $\mathcal{A}$, that is the set of MSCs whose linearizations are successful runs. Moreover, we say that a CFM is *deadlock-free* if every run can be extended to a successful run.

3 Existential Bound on Channels

CMSC-graphs are more expressive than CFMs [10], and thus most non trivial problems for CMSC-graphs are undecidable. The solution applied here to recover decidability is to consider representative linearizations, as what was done first for MSC-graphs by [20] and for CMSC-graphs by [18]. More precisely, if G is $\exists$-b-bounded, then every MSC $M \in \mathcal{L}(G)$ has a linearization in the set $\mathrm{Lin}^b(G)$ of b-bounded linearizations of $\mathcal{L}(G)$. We call the set $\mathrm{Lin}^b(G)$ a set of representatives for $\mathcal{L}(G)$, since any MSC of $\mathcal{L}(G)$ can be retrived from a linearization of $\mathrm{Lin}^b(G)$. To ensure an existential bound on channels, *safe CMSC-graphs* (called *realizable CMSC-graph* by [10], and simply CMSC-graphs by [18]) were defined.

Definition 6 *A CMSC-graph G is safe if every sequence of CMSCs labeling an accepting path of G admits one composition that is an MSC.*

Recall that some paths of a (non safe) CMSC-graph may not admit a composition that is an MSC (e.g. because number of sends and receives from p to q are not equal). Notice that safe CMSC-graphs contain the class of MSC-graphs. For instance, figure 2 depicts a safe CMSC-graph. Being safe implies that each loop of the CMSC-graph G is labeled by the same number of sends and receives from p to q, for each pair of processes p, q. This is the key argument for the syntactical characterization of safe CMSC-graphs that can be checked in polynomial time [10]. For instance, the CMSC-graph depicted on the right part of figure 1 is not safe since loops are labeled with a different number of sends and receives.

Let K_G be the automaton obtained from G by replacing every node v of G by a sequence of $|\lambda(v)|$ transitions of the automaton, labeled by some linearization of $\lambda(v)$. The language $L(K_G)$ of this automaton is a set of representatives of $\mathcal{L}(G)$. Since each loop of the CMSC-graph G is labeled by the same number of sends and receives from p to q, for each pair of processes p, q, every $x \in L(K_G)$

is b-bounded, with $b \leq |G|$. Since this bound is important for G, we denote the b for which $L(K_G)$ is universally b-bounded as b_G. Hence, G is existentially b_G-bounded.

We will use the class of globally-cooperative CMSC-graphs as a central class of our implementation algorithm. The reason is that this class ensures the existence of a regular set of representatives, namely $\mathrm{Lin}^{b_G}(G)$.

Definition 7 *The communication graph of a CMSC M is a directed graph whose vertices are the processes involved in M, and there is an edge between vertices p, q iff M contains both a send $p!q$ from p to q and a receive $q?p$ on q from p (these send and receive may not define the same message).*

For instance, the communication graph of the MSC made of one message from process 3 to 2 and one message from process 1 to 2 is weakly connected. We recall that a loop in a CMSC-graph is a path starting and ending in the same node (we do not require that the loop is simple, that is, a loop can meet several times the same node).

Definition 8 *A CMSC-graph is loop-connected if every loop is labeled by a CMSC whose communication graph is weakly connected.*

*A CMSC-graph is globally-cooperative (*gc-CMSC-graph *for short) if it is safe and loop-connected.*

In particular, if there is a loop labeled by two groups of processes without communication between the two groups, then G is not loop-connected, hence not globally-cooperative.

Proposition 1 *[6] The set $Lin^b(G)$ of b-bounded linearizations of every globally-cooperative CMSC-graph G is regular, for all $b \geq b_G$.*

4 Implementation by CFMs

It is easy to see that not every globally-cooperative CMSC-graph is implementable by a deadlock-free CFM (actually, they are always implementable by a CFM with possible deadlocks [6]). Since any specification should be implementable, we need a test for implementability.

Definition 9 *A CMSC-graph G is implementable without additional data iff there exists some deadlock-free CFM $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(G)$.*

There is an EXPSPACE-complete algorithm to test whether a globally-cooperative MSC-graph is implementable without additional data [1, 16]. Anyway, there are two drawbacks in such an approach. First, the algorithm is obviously time-consuming. Second, implementing directly an MSC-graph is too extreme, since some easily implementable MSC-graphs are said not to be, as the globally-cooperative MSC-graph of figure 2. The reason is that the data written in the first message is abstracted away, hence both host and function can choose to send the second message, yielding a scenario that is not possible in the system, thus a deadlock. The solution already used in [9, 13] is to allow data to be added to messages. For instance, we would add in the first message a bit to indicate which process (host or function) must send. A data projection function simply projects away the additional data from messages.

Definition 10 *A CMSC-graph G is implementable (with additional data) iff there exists some data projection Proj and some deadlock-free CFM $\mathcal{A}$ with* $Proj(\mathcal{L}(A)) = \mathcal{L}(G)$.

The problem is that we have no algorithm to test this implementability. Moreover, even if such an algorithm would exist, it would probably be too time consuming. We propose then an alternative approach to the problem, trying to go through a class which is easily implementable with additional data. The reason for non-implementability of an MSC-graph is the global control, whereas the choices in a CFM must be done locally. The idea is then to define MSC-graphs that have only local choices, that is, any node is controlled by a single process [3, 12].

Definition 11 *Let G be a CMSC-graph. Then G is a* local-choice *CMSC-graph if*

- *G is safe*
- *For each transition $v \to w$, node w has a unique minimal event* $\min(w)$. *Moreover, the minimal process of w, denoted $pmin(w)$, appears in v.*
- *There exists a process p_0 such that every initial node of G has a unique minimal event on p_0.*

A local-choice MSC-graph is a local-choice CMSC-graph which is an MSC-graph.

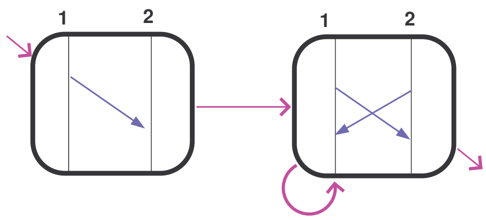

Fig. 3. A globally-cooperative MSC-graph universally bounded but not local-choice

Example 1. The MSC-graph in figure 2 is local-choice. The MSC-graph in figure 3 is not local-choice, since the looping node has two minimal events.

The next proposition follows easily from [9].

Proposition 2 *Any local-choice CMSC-graph is implementable. Moreover, the size of the CFM obtained is linear in the size of the local-choice CMSC-graph.*

The local-choice restriction appears to be a heuristics for implementation. That is, if a CMSC-graph is local-choice or equivalent to some local-choice CMSC-graph, then it is implementable (without deadlock). However, if it is not equivalent to a local-choice CMSC-graph, then this does not mean that it is not implementable.

4.1 A Concrete Protocol: USB

The protocol *USB* (Universal Serial Bus) describes several communication modes between two communicating processes, a master (called host), and a slave (called function) in the norm [21]. Every command is given by host. That is, the first message of each mode is from host to function, and contains the command (mode chosen, actions to perform, etc.). Three kinds of interactions can be done, Isochronous, Bulk and Setup.

The isochronous mode is described by the local-choice MSC-graph in figure 2. The first message tells function that host has chosen the isochronous mode, and whether host must send or receive information. Setup mode is a slight variation of the isochronous mode.

Bulk transfer looks like the alternated bit protocol. Every message received should be acknowledged with the parity of the message, such that the sender can be sure that his message was indeed received. In order to bound the channel, a limit for send events in transit is imposed. We represent a part of the Bulk protocol in the upper part of figure 4.

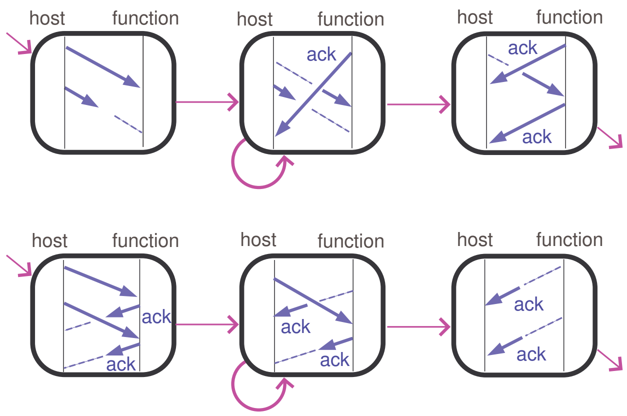

Fig. 4. Equivalent CMSC-graphs specifying the Bulk transactions of usb 1.1.

The CMSC-graph in the upper part of figure 4 is not local-choice (the looping node has two minimal events). However, we can transform this CMSC-graph into the equivalent local-choice CMSC-graph depicted in the lower part of figure 4. We want to give an algorithm to build such an equivalent local-choice CMSC-graph, whenever it is possible.

5 Implementation Algorithms

5.1 Characterizing local-choice MSC-graphs

A crucial notion related to local-choice are triangles. We call a CMSC T a *triangle* if it has a unique minimal event $min(T)$ for the visual order. A triangle T is called an *MSC-triangle* if it is an MSC. Let T_n be the set of triangles of size bounded by n. We define a generic CMSC-graph H_n^T: for each triangle $T \in T_n$, it has a node v_T labeled by T. There is a transition $v_T \to v_{T'}$ whenever $P(\min(T')) \in P(T)$. We define in the same line the generic local-choice MSC-graph H_n^M on MSC-triangles of size at most n.

The next proposition shows that we must consider only globally-cooperative CMSC-graphs for our implementability test:

Proposition 3 *Let G be a safe CMSC-graph that is not globally-cooperative. Then G is not equivalent to any local-choice CMSC-graph.*

Theorem 1 *A globally-cooperative CMSC-graph G is equivalent to some local-choice CMSC-graph iff it exists some n with $\mathcal{L}(G) \subseteq \mathcal{L}(H_n^T)$. If it is the case, then one can obtain some local-choice CMSC-graph equivalent to G, of size exponential in $|G|$ and n.*

Sketch of Proof. If $\mathcal{L}(G) \subseteq \mathcal{L}(H)$ for some local-choice CMSC-graph H, then $\mathcal{L}(G) \subseteq \mathcal{L}(H_n^T)$ with $n = |H|$.

Conversely, if $\mathcal{L}(G) \subseteq \mathcal{L}(H_n^T)$, then we compute an automaton $\mathcal{A}$ accepting $Lin^{b_G+(\wp+b_G\wp^2)n}(G)$ using proposition 1, of size exponential in n and $|G|$. Making the product between $\mathcal{A}$ and H_n^T, we obtain a CMSC-graph H of size $|\mathcal{A}||H_n^T|$ that is local-choice. To see that it is equivalent to G, we have to show that for any decomposition of $M \in \mathcal{L}(G)$ into a sequence of triangles $T_1 \cdots T_m$ of size at most n, there exist linearizations $x_1, \cdots, x_m$ of $T_1, \cdots, T_m$ such that $x_1 \cdots x_m$ is $b_G + (\wp + b_G\wp^2)n$-bounded. By contradiction, assume that for some channel (p, q), the linearization $x_1 \cdots x_{k-1}$ has b_G unmatched sends (we denote by s_0 the first unmatched send), $x_k \cdots x_l$ has $(\wp + b_G\wp^2)n + 1$ unmatched sends, and x_{l+1} contains r_0, the receive associated with s_0. Hence, there are at least $(\wp + b_G\wp^2) + 1$ triangles containing the unmatched sends of $x_k \cdots x_l$ from p to q. Since $M \in \mathcal{L}(G)$, there exists some b_G-bounded linearization x equivalent to $x_1 \cdots x_{l+1}$. In x, the receive r_0 must occur before all unmatched sends in $x_k \cdots x_l$ from p to q. So the past of r_0 (i.e., all events e with $e \leq r_0$) occurs in x before the unmatched sends in $x_k \cdots x_l$. Notice that the past of r_0 contains the minimal event of each triangle in $T_k, \ldots, T_l$ that has at least one unmatched send. It is now easy to check that the past of r_0 restricted to any of these triangles either eliminates some process from the past restricted to later triangles, or it contains an unmatched send. Since there are at most $\wp$ triangles of the first kind, there must be at least $b_G\wp^2 + 1$ triangles of the second kind, hence at least $b_G\wp^2 + 1$ unmatched sends in x before the unmatched sends of $x_k \cdots x_l$. So there is at least one channel with $b_G + 1$ unmatched sends, which contradicts the b_G-boundedness of x.

$\square$

We can state theorem 1 similarly for local-choice MSC-graphs, by replacing H_n^T by H_n^M. Theorem 1 will give an algorithm for testing whether G is equivalent to some local-choice CMSC-graph as soon as we limit the value of n for which we must test $\mathcal{L}(G) \subseteq \mathcal{L}(H_n^T)$.

We show now some structural properties that must be satisfied by the CMSC-graphs we are interested in. We call two **MSCs** R, S a *couple in parallel for* G if $P(R) \cap P(S) = \emptyset$ and there exist CMSCs L, N with $LRSN \in \mathcal{L}(G)$.

Proposition 4 *Let G be a local-choice CMSC-graph. Let R, S be a couple in parallel in G associated with the CMSCs L, N. Then either $|R| \leq 2\wp|G|$, or $|S| \leq 2\wp|G|$.*

We give another property that concerns only local-choice MSC-graphs, and not CMSC-graphs.

Let M be an MSC and e be an event of M. We call e a *peak* of M if its future $\text{Future}(e) = \{f \in M \mid e \leq f\}$ for the visual order of M is an MSC (that is, if it contains some send or receive, it should also contain the associated event). In a local-choice **MSC**-graph, every event that starts a node is a peak, which is not always the case in a local-choice CMSC-graph. Let G be a CMSC-graph. We say that an MSC-triangle M is an *MSC-triangle without G-peak* if it exists an MSC-triangle L and an MSC N with $LMN \in \mathcal{L}(G)$ and LMN has a unique peak within M (which is min_M). It is worth noting that LMN can have peaks other than min_M, as soon as these peaks are not within M. Moreover, LM can have several peaks within M, but they will not be peaks anymore for LMN.

Proposition 5 *Let G be a local-choice MSC-graph. Let M be an MSC-triangle without G-peak. Then $|M| \leq 2\wp|G|$.*

Using these notions, we can characterize the class of safe CMSC-graphs which are equivalent to any local-choice MSC-graph:

Theorem 2 *Let G be a safe CMSC-graph. Then G is equivalent to some local-choice MSC-graph iff there exists some integer n such that:*

1. *G is globally cooperative.*
2. *Each $M \in \mathcal{L}(G)$ is a triangle.*
3. *Each MSC couple M, N in parallel for G satisfies $|M| \leq n$, or $|N| \leq n$.*
4. *Each MSC-triangle M without G-peaks satisfies $|M| \leq n$.*

5.2 A Tractable Test Algorithm

The first test of theorem 2 is co-NP-complete [19]. The second test is NLOGSPACE [7]. The third test is co-NP [7]. The fourth test is PSPACE [7]. If the third and fourth tests are satisfied, then we have a value for n, such that $\mathcal{L}(G) \subseteq \mathcal{L}(H_{\wp n}^M)$, and then we can compute an equivalent local-choice MSC-graph using theorem 1. The problem is that the fourth test gives an exponential value to n (while the third gives a polynomial value to n), making the implementation potentially doubly exponential. However, the fourth test makes no sense for local-choice CMSC-graph, for which we can do better.

Example 2. The globally-cooperative MSC-graph in figure 3 is not equivalent to any local-choice MSC-graph, but it is not hard to show the equivalence with a local-choice CMSC-graph.

We turn now to the test whether a given safe CMSC-graph is equivalent to some local-choice CMSC-graph. We characterize triangles that cannot belong to $\mathcal{L}(H_n^T)$, that is which cannot be decomposed in a sequence of triangles of size at most n. Let $T = T_1T_2$ be a decomposition of such a triangle into two triangles. We call the minimal events $\min(T) = \min(T_1) = e$ and $\min(T_2) = f$. In a CMSC, there are at most two immediate successors g, h of e (the event g on the same process as e, and the receive h of e if e is a send). Obviously, either $f \geq g$ or $f \geq h$. That is, if we want to minimize the size of T_1, an optimal choice is to take either $f = g$, or $f = h$[1]. The triangle T_2 is defined as the set of events $\mathrm{Future}(f)$ in the future of f, and T_1 is the set of events that are not in $\mathrm{Future}(f)$, that is $F(f) = \mathrm{Future}(e) \setminus \mathrm{Future}(f) = T_1$. That is, if $|F(g)| > n$ and $|F(h)| > n$, then T is not decomposable into a sequence of triangles of size at most n. Furthermore, if T labels a path of a safe CMSC-graph G, and if $F(g)$ and $F(h)$ are large enough, then we can find a loop of G in $F(f)$ and one in $F(g)$ that we can iterate such that both $F(g)$ and $F(h)$ become as large as we want. That is, $\mathcal{L}(G) \not\subseteq \mathrm{Ł}(H_n^T)$ for any n.

Iterating one of these loops should not delete any event in $F(h)$ or in $F(g)$ because of a new dependency. To do so, we need to define the Ω-type, which is related to the existential bound b_G associated with G. The Ω-type of an event e is its type $t \in \{p!q, p?q\}$, plus the number modulo b_G of events of the same type that have happened before e, that is, $\Omega = \mathcal{T} \times \{0, \cdots, b_G - 1\}$. We denote by $type(X)$ the set of Ω-types of events in X.

Lemma 1. *Let MBN be an MSC that has two minimal events g, h, and assume that $type(Future(g) \cap M) = type(Future(g) \cap MB)$ and $type(Future(h) \cap M) = type(Future(h) \cap MB)$. We denote by $Future', F'$ and m' the functions corresponding to $Future, F$ and m with respect to $MBBN$. Assume that M is existentially-b_G-bounded.*

Then $F(g) \subseteq F'(g)$ and $F(h) \subseteq F'(h)$.

Proof. Let $f \in \{g, h\}$. Assume by contradiction that $\mathrm{Future}'(f) \cap F(f) \neq \emptyset$. We denote $d_1 \prec d_2 \prec \cdots \prec d_m$ a causality chain in $MBBN$ with $d_1 = f, d_m \in F(f)$, and where $d_i \prec d_{i+1}$ if $m'(d_i) = d_{i+1}$ or if $d_i <_p d_{i+1}$ for some process p.

We will show that $d_m \in \mathrm{Future}(f)$, a contradiction with $d_m \in F(f)$. Assume that there is an i with d_i in the first B and d_{i+1} in the second B. We will delete the first B of $MBBN$ to obtain MBN. In MBN, we still have $d_{i+1} < d_m$. Since B conserves the Ω-types of $\mathrm{Future}(f)$, we have a $d_i' \in \mathrm{Future}(f) \cap M$, of same Ω-type as d_i. If $d_i <_p d_{i+1}$, then within MBN, we also have $d_i' <_p d_{i+1}$. Hence $d_m \in \mathrm{Future}(f)$. Else, we have $m(d_i) = d_{i+1}$ in $MBBN$. Let d be the first event of B of same $\mathcal{T}$-type as d_i (that is, the second component of its Ω-type can

[1] Actually, we take the only immediate successors of h instead of h since we want that the minimal process of a node belongs to any predecessor node.

differs from d_i). We have $d'_i <_p d \leq_p d_i$ for some p. Hence $d \in \mathrm{Future}(f) \cap MB$ and there exists some $d' \in \mathrm{Future}(f) \cap M$, of same Ω-type as d. Hence, we have at least b_G sends of same $\mathcal{T}$-type as d_i in $[d', d[$, that is in $\mathrm{Future}(f) \cap M$.

Since M is existentially b_G-bounded, it has at most b_G unmatched sends: if we delete the first B, there exists some $d'' \in [d', d[\subseteq \mathrm{Future}(f) \cap M$ with $m(d'') = d_{i+1}$. Hence $d_m \in \mathrm{Future}(f)$.

It remains to consider the case where there is one of the two occurrences of B that contains no $(d_i)_{i \leq m}$. We will then delete this occurrence of B to obtain MBN. We consider the new ordering relation in MBN. If $d_i <_p d_{i+1}$ in $MBBN$, then this is also true in MBN. If $m'(d_i) = d_{i+1}$, assume that $m(d_i) \neq d_{i+1}$. Else, $d_m \in \mathrm{Future}(f)$. We have some d_i before the deleted B, and d_{i+1} after the deleted B. In the deleted B, there exists a send d of same $\mathcal{T}$-type as d_i. We can apply the same arguments than above to prove that $d_m \in \mathrm{Future}(f)$. □

Let us recall that H_n^T is a CMSC-graph whose nodes are triangles of size at most n.

Proposition 6 *A globally-cooperative CMSC-graph G is equivalent to a local-choice CMSC-graph if and only if $\mathcal{L}(G) \subseteq \mathcal{L}(H_{b_0}^T)$, with $b_0 = 4b_G\wp^2|G| + 1$. This test can be done in co-NP.*

Proof. Assume that $\mathcal{L}(G) \not\subseteq \mathcal{L}(H_{b_0}^T)$. It means that there exists an MSC $M \in \mathcal{L}(G)$ and a send $e \in M$ such that for all immediate $<$-successors $f \in \{g, h\}$ of e, we have $|F(f)| > b_0$. Else, we could decompose inductively any triangle $M \in \mathcal{L}(G)$ into triangles of size at most b_0. This test can be performed in co-NP.

We show now how to increase the size of $F(g)$ without decreasing the size of $F(h)$. By symmetry, we will do the same for augmenting $F(h)$. The MSC M labels a path of G. Since there are b_0 events in $F(g)$, there are at least $4b_G\wp^2 + 1$ occurrences of the same event $e_g \in F(g)$. Hence we can decompose M into a sequence $M = BB_1 \cdots B_n B'$ with B_i labeling a loop of G. Moreover, B_i begins and ends by e_g, and $n = 4b_G\wp^2 + 1$.

Among the loops $B_1 \cdots B_n$, at most $2b_G\wp^2$ can change the Ω-types of $\mathrm{Future}(g)$. More formally, let $Type_i(g)$ be the set of Ω-types of events in $(BB_1 \cdots B_i) \cap \mathrm{Future}(g)$. There are at most $2b_G\wp^2$ loops B_i with $Type_{i-1}(f) \neq Type_i(f)$, since $Type_i(f)$ is an increasing sequence of sets of size at most $2b_G\wp^2$. In the same line, there are at most $2b_G\wp^2$ loops that can change the Ω-types of $\mathrm{Future}(h)$. That is, there is at least one loop, say B_k, that changes neither the Ω-type of $\mathrm{Future}(g)$, nor those of $\mathrm{Future}(h)$. We can then iterate the loop B_k without deleting any event from $F(g)$ or $F(h)$, applying the lemma 1.

Since B_k contains $e_g \in F(g)$ and does not change the Ω-types of $\mathrm{Future}(g)$, iterating the loop B_k makes $F(g)$ strictly grow.

In the same line, we can decompose M with respect to $F(h)$. Hence, we can iterate a loop that makes $F(h)$ strictly grow without shrinking $F(g)$. Since G is safe, we obtain an MSC of $\mathcal{L}(G)$ by iterating these two loops. Hence, we obtain for all k an MSC $M_k \in \mathcal{L}(G) \setminus \mathcal{L}(H_k^T)$. That is, G cannot be equivalent to any local-choice CMSC-graph.

□

Using theorem 1 and the previous proposition, we obtain:

Theorem 3 *Let G be a safe CMSC-graph. Then one can decide in co-NP whether G is equivalent to some local-choice CMSC-graph. If the answer is positive, then an equivalent local-choice CMSC-graph can be built of size exponential in $|G|$.*

Here is an example for which we need an exponential-blowup for coming from a globally-cooperative CMSC-graph to a local-choice CMSC-graph. Since there is a loop on host, we need to put the n local events on the same node. Since we have 2 choices for each local events, it yields 2^n nodes in any equivalent local-choice CMSC-graphs.

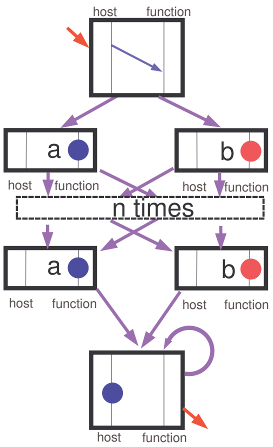

Fig. 5. globally-cooperative CMSC-graph hard to turn into a local-choice CMSC-graph

Conclusion

We presented an algorithm testing implementability of a scenario-based specification, CMSC-graphs, into local-choice CMSC-graphs, which is a strict subclass of deadlock-free CFMs. This test seems practical since it is co-NP and gives an implementation of size exponentially larger than the specification in the worst case.

This test is an improvement in both expressivity and complexity of the internal report [7], even when the model is given as an MSC-graph. That is, compositionality appears only as a technical step in our construction, and needs not to be known by the user.

There are two restrictions of local-choice MSC-graphs for turning them into CFMs. The first one is the need of numerous peaks, and the second one is a restriction on the number of events that are pairwise concurrent. We succeeded in getting rid of the first restriction using compositional MSC-graphs. A further work will consist in finding specification formalisms that allow more parallelism than local-choice CMSC-graphs.

Acknowledgments I would like to thank Anca Muscholl for fruitful discussions.

References

1. R. Alur, K. Etessami, and M. Yannakakis. Realizability and verification of MSC graphs. In *ICALP'01*, LNCS 2076, pp.797-808, 2001.
2. R. Alur and M. Yannakakis. Model checking of message sequence charts. In *CONCUR'99*, LNCS 1664, pp.114-129, 1999.
3. H. Ben-Abdallah and S. Leue. Syntactic detection of process divergence and non-local choice in MSCs. In *TACAS'97*, LNCS 1217, pp.259-274, 1997.
4. Nicolas Baudru and Rémi Morin. Safe implementability of regular message sequence chart specifications. In *(SNPD'03)*, pp 210–217. ACIS, 2003.
5. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):pp.323-342, 1983.
6. Blaise Genest, Dietrich Kuske, and Anca Muscholl. A Kleene theorem for a class of communicating automata with effective algorithms, to appear in *DLT*, LNCS, 2004.
7. B. Genest and A. Muscholl. The structure of local choice in HMSC Internal report LIAFA, 2003 Available at http://www.crans.org/~genest/GM03.ps.
8. B. Genest, M. Minea, A. Muscholl, and D. Peled. Specifying and verifying partial order properties using template MSCs. In *FoSSaCS'04*, LNCS 2987, pp.195-210, 2004.
9. B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state High-level MSCs: Model-checking and realizability. In *ICALP'02*, LNCS 2380, pp.657-668, 2002.
10. E. Gunter, A. Muscholl, and D. Peled. Compositional Message Sequence Charts. In International Journal on Software Tools for Technology Transfer (STTT), Volume 5, Springer, 2003.
11. D. Harel and R. Marelly. Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine. Springer 2003.
12. Loïc Hélouët and Claude Jard. Conditions for synthesis of communicating automata from HMSCs. In *5th FMICS'00*, 2000.
13. J. G. Henriksen, M. Mukund, K. Narayan Kumar, M. Sohoni and P. Thiagarajan. A Theory of Regular MSC Languages To appear In *IC*, 2004, available at http://www.comp.nus.edu.sg/~thiagu/icregmsc.pdf.
14. ITU-TS recommendation Z.120, Message Sequence Charts, Geneva, 1999.
15. D. Kuske. Regular sets of infinite message sequence charts. In *Information and Computation 187*, Academic Press, pp.80-109, 2003.
16. M. Lohrey. Safe realizability of High-level Message Sequence Charts. In *CONCUR'02*, LNCS 2421, pp.177-192, 2002.
17. M. Lohrey and A. Muscholl. Bounded MSC communication In *Information and Computation 189*, Academic Press, pp.135-263, 2004.
18. P. Madhusudan and B. Meenakshi. Beyond Message Sequence Graphs In FSTTCS'01, LNCS 2245, pp.256-267, 2001.
19. A. Muscholl and D. Peled. Message Sequence Graphs and decision problems on Mazurkiewicz traces. In *MFCS'99*, LNCS 1672, pp.81-91, 1999.
20. D. Peled. Specification and verification of Message Sequence Charts. In *FORTE/PSTV'00*, pp.139-154, 2000.
21. USB 1.1 specification, available at http://www.usb.org/developers/docs/usbspec.zip
22. Bikram Sengupta and Rance Cleaveland. Triggered Message Sequence Charts. In *SIGSOFT 2002/FSE-10*. ACM Press, 2002.