Beyond Message Sequence Graphs

P. Madhusudan[1] and B. Meenakshi[2]

[1] Chennai Mathematical Institute, Chennai, India.
madhu@imsc.ernet.in
[2] The Institute of Mathematical Sciences, Chennai, India.
bmeena@imsc.ernet.in

Abstract. We study the model-checking problem for classes of message sequence charts (MSCs) defined by two extensions of message sequence graphs (MSGs). These classes subsume the class of regular MSC languages. We show that the model checking problem for these extended message sequence graphs against monadic second-order specifications is decidable. Moreover, we present two ways to model-check the extended classes — one extends the proof for MSGs while the other extends the proof for regular MSC languages.

1 Introduction

Message sequence charts (MSCs) is an ITU-standardized notation widely used to capture system requirements in the early stages of design of communication protocols [ITU97,RGG96]. The clear graphical layout of an MSC describes how the components of a distributed system communicate by exchanging messages. In its simplest form, an MSC depicts the exchange of messages between processes of a distributed system and corresponds to a single partially-ordered execution of the system. A standard way to specify a collection of scenarios is by using *message sequence graphs* (MSGs) (also known as high-level MSCs). MSGs allow MSCs to be combined using the operations of choice, concatenation and repetition.

In [AY99], the authors study the (linearization) model-checking problem for message sequence graphs. One is given an MSG and a regular set of sequences of actions as the specification and asked whether for every MSC represented by the MSG, all the linearizations of the MSC are included in the specification. This problem is shown to be undecidable, and the authors go on to define a restricted class of MSGs for which the problem is decidable.

The model-checking problem of MSGs against monadic second-order (MSO) formulas, which are interpreted directly on MSCs and not on its linearizations, was studied in [Mad01]. It was shown that the model-checking problem was decidable without any restriction on the MSGs. Independently, Peled has shown that model-checking a fragment of TLC (a temporal logic on partial-orders) is solvable for unrestricted MSGs [Pel00]. These results suggest that structural logics may be more amenable to verification than those based on linearizations.

In [HMKT00b], the authors study another class of MSCs called *regular MSC-languages*. It turns out that using a proof in [HMKT00b], one can show that the model-checking problem for regular MSC languages against MSO specifications is decidable as well. However, the class of MSC-languages defined by MSGs and the class of regular MSC-languages are incomparable — there are MSC-languages that are definable in one but not in the other [HMKT00a].

Thus, though the model-checking problem for MSO formulas is decidable for MSC-languages presented as MSGs as well as for regular MSC languages, the proof methodologies seem very different. While the former works by viewing concatenated MSCs as meta-strings where each letter represents an MSC, the latter works using the set of all linearizations of the MSCs.

We unify the proof mechanisms in the above works by considering a larger class of MSCs through the formalism of *compositional Message Sequence Graphs* (CMSGs) introduced in [GMP01]. These are similar to MSGs, except that one is allowed to have *unmatched* send-events (send-events whose corresponding receive-events do not occur in the current atomic MSC) while forming an MSC, which will be matched with other receive-events in other MSCs. In [GMP01] it was shown that this is a strictly more expressive formalism than that of an MSG, and that it allows modelling of interesting scenarios such as the behaviour of the alternating-bit protocol.

In terms of linearizations, a regular MSC language is one such that *all* the linearizations of all MSCs in it is a regular set of sequences [HMKT00b]. We can relax this condition, and represent an MSC-language by a subset L of linearizations of MSCs in the MSC-language such that L has *at least one* linearization for each MSC in the language. When L is regular, we say that the MSC-language has a *regular representative linearization*. It then turns out that the class of CMSG definable languages is precisely the class of MSC-languages that have a regular representative linearization. Consequently, this class of languages subsumes the class of MSG-definable languages as well as regular MSC-languages.

We show that the model-checking problem for CMSGs against MSO specifications is decidable. More importantly, we identify two ways of solving this problem — one extending the method used to model-check MSGs in [Mad01] and the other technique implicit in [HMKT99] to model-check regular MSC-languages.

We then go on to define and study a formalism more general and flexible than a CMSG, called *extended CMSGs* (XCMSGs). In this setting, apart from unmatched send-events, one is also allowed to have unmatched receive-events when traversing a path in the MSG. We show that the notion analogous to linearizations in this setting is that of *semi-linearizations* — an ordering of events which respects the local total orders of the MSC but *does not* necessarily respect the causal order. We show then theorems analogous to the above results — equivalence between MSC-languages defined by XCMSGs and those that have regular representative semi-linearizations and how the two proofs of decidability of the model-checking problem extend to this class as well.

We believe our results will help understand the nuances involved in model-checking classes of MSC-languages, which borders on the line of undecidability. On the theoretical side, checking MSO properties of infinite graphs and infinite classes of graphs is an interesting area of research [Tho97]. Our proofs using linearizations and semi-linearizations seem to be suggest a new technique to tackle this problem for certain classes of graphs.

Due to space constraints, we present only the gist of proofs — more details and related results (like extensions to infinite MSCs, etc.) can be found in [MM01].

2 Preliminaries

Let us fix a finite set of processes $\mathcal{P}$ for the rest of the paper and let p, q, r range over $\mathcal{P}$. Let us also fix a finite set of messages Γ_m and a finite set of local actions Γ_l. Let $\Sigma_p = \{(p!q, a), (p?q, a), (p, l) \mid p, q \in \mathcal{P}, p \neq q, a \in \Gamma_m, l \in \Gamma_l\}$ denote the set of actions that p participates in. The action $(p!q, a)$ should be read as "p sends the message a to q" while the action $(p?q, a)$ means "p receives the message a from q". (p, l) represents p doing a local action l. Let $\Sigma = \bigcup_{p \in \mathcal{P}} \Sigma_p$ denote the set of all actions.

Definition 1. *A* message sequence chart *(MSC) over $\mathcal{P}$ is a tuple $m = (E, \{\leq_p\}_{p \in \mathcal{P}}, \lambda, \eta)$ where*

- *E is a finite set of events.*
- *$\lambda : E \to \Sigma$ is the labelling function which identifies for each event an action. Let $E_p = \{e \in E \mid \lambda(e) \in \Sigma_p\}$ denote the set of events of E which p participates in. Also, let $S_E = \{e \in E \mid \lambda(e) = (p!q, a)$ for some $p, q \in \mathcal{P}, a \in \Gamma_m\}$ denote the set of send-events and $R_E = \{e \in E \mid \lambda(e) = (p?q, a)$ for some $p, q \in \mathcal{P}, a \in \Gamma_m\}$ denote the set of receive-events of E.*
- *$\eta : S_E \to R_E$ is the matching function which associates with each send-event, its corresponding receive-event. We require η to be a bijection and for every $e, e' \in E$, if $\eta(e) = e'$, then $\lambda(e) = (p!q, a)$ and $\lambda(e') = (q?p, a)$ for some $p, q \in \mathcal{P}, a \in \Gamma_m$.*
- *$\leq_p$ is a total order on E_p for each $p \in \mathcal{P}$.*
- *Let $\widehat{\leq} = (\bigcup_{p \in \mathcal{P}} \leq_p) \cup \{(e, e') \mid e, e' \in E$ and $\eta(e) = e'\}$. Let $\leq = (\widehat{\leq})^*$ be the transitive closure of $\widehat{\leq}$. Then $\leq$ denotes the* causal ordering *of events in the MSC and we require it to be a partial order on E.*
- (non-degeneracy) *m is non-degenerate ([AEY00]) in the sense that two identical messages sent by a process are not received in the reverse order: i.e. if there are events e_1, e_2, e'_1, e'_2 such that $\lambda(e_1) = \lambda(e_2) = (p!q, a)$, $\eta(e_1) = e'_1$, $\eta(e_2) = e'_2$ and $e_1 <_p e_2$, then it must be the case that $e'_1 <_q e'_2$.* □

An *event-linearization* of an MSC m is a linear order on E which extends $\leq$, the causal order of m. We represent event-linearizations as sequences over E: a sequence $e_1, \ldots, e_k$ represents the event-linearization $\sqsubseteq$ given by $e_i \sqsubseteq e_j$ iff $i \leq j$.

A *linearization* of an MSC m is any possible sequence of actions which the MSC describes. Formally, *lin*(m), the set of linearizations of m, is the set of sequences $\lambda(e_1), \ldots, \lambda(e_k)$ where $e_1, \ldots, e_k$ is an event-linearization of m. Linearizations are represented as words in Σ^*.

Note that while an MSC defines a nonempty set of linearizations, one can associate a unique MSC (up to isomorphism) with a given linearization by defining an appropriate set of events and matching the sends and receives using the non-degeneracy condition for an MSC. For a linearization $w \in \Sigma^*$, we can define the MSC corresponding to w as $(E, \{\leq_p\}_{p \in \mathcal{P}}, \lambda, \eta)$, where E is the set of all non-epsilon prefixes of w, $\lambda(x.r) = r$, where $x.r \in E$, $x \in \Sigma^*$, $r \in \Sigma$. $x.r \leq_p y.r'$ iff $x.r$ is a prefix of $y.r'$ and $r, r' \in \Sigma_p$. $\eta(x.r) = y.r'$ iff $r = (p!q, a)$, $r' = (q?p, a)$ and the number of occurrences of r in x is equal to the number of occurrences of r' in y.

For any $w \in \Sigma^*$, if for every $p, q \in \mathcal{P}$, $a \in \Gamma_m$, and every prefix y of w, the number of occurrences of $(p!q, a)$ in y is at least the number of occurrences of $(q?p, a)$ in y, and the number of occurrences of $(p!q, a)$ in w is the same as the number of occurrences of $(q?p, a)$ in w, then one can associate an MSC m for which w is a linearization. We call such words *well-formed* words. For such a $w \in \Sigma^*$, let $msc(w)$ denote the MSC corresponding to w.

We can now represent a collection of MSCs through a set of *representative linearizations*. Let $L \subseteq \Sigma^*$ be a set of well-formed words. Then L represents the class of MSCs $msc(L) = \{msc(w) \mid w \in L\}$. Note that we do not require L to have *all* the linearizations of the MSCs it represents (i.e. L need not be equal to $lin(msc(L))$), as required for *regular MSC languages* [HMKT00b]. We say that L is a representative linearization of a collection of MSCs if L represents this collection (up to isomorphism).

Another way to define a collection of MSCs is through the notion of a *Message Sequence Graph* (MSG). This is a mechanism whereby one takes a finite set of MSCs and defines regular ways of combining them using choice, concatenation and repetition. In this paper, we will consider a more general notion of MSGs, called *compositional* MSGs, or CMSGs.

Let us first define a compositional MSC (CMSC). A CMSC is basically an MSC which, in addition to the normal messages in an MSC, can also have unmatched send and receive events — these will be matched up later using corresponding unmatched receive and send events respectively, when the CMSC is composed with another.

Definition 2. *A* compositional message sequence chart *(CMSC) over $\mathcal{P}$ is a tuple $\hat{m} = (E, \{\leq_p\}_{p \in \mathcal{P}}, \lambda, \eta)$ where E, λ and $\leq_p$ are as in an MSC and $\eta : S_E \rightharpoonup R_E$ is a* partial *matching function which associates with some send-events, their corresponding receive-events. We require η to be injective and for every $e, e' \in E$, if $\eta(e) = e'$, then $\lambda(e) = (p!q, a)$ and $\lambda(e') = (q?p, a)$ for some $p, q \in \mathcal{P}, a \in \Gamma_m$. Using the events on which η is defined, we can define the causal order, as in an MSC. We require this to be a partial order. We also require $\hat{m}$ to be non-degenerate (with respect to the messages matched by η).* ☐

We say that $e \in S_E$ is an *unmatched send-event* if η is not defined on e and $e \in R_E$ is an *unmatched receive-event* if $e \notin \eta(S_E)$. For a CMSC $\hat{m}$ and $(p!q, a) \in$

Σ, let $\#^{unm}_{(p!q,a)}(\hat{m}) = |\{e \in E \mid \lambda(e) = (p!q,a) \text{ and } \eta(e) \text{ is not defined }\}|$ denote the number of unmatched send-events of the type $(p!q, a)$.

Concatenation of two CMSCs is done by (asynchronously) concatenating the events of each process, and by matching up some unmatched send-events of the first CMSC with corresponding unmatched receive-events of the second.

Definition 3. *Let $\hat{m}_i = (E_i, \{\leq^i_p\}_{p \in \mathcal{P}}, \lambda_i, \eta_i)$, $i = 1, 2$ be two CMSCs, with $E_1 \cap E_2 = \emptyset$. Also, let $\hat{m}_1$ have no unmatched receives. Let $\hat{m} = (E, \{\leq_p\}_{p \in \mathcal{P}}, \lambda, \eta)$ where $E = E_1 \cup E_2$; $\forall e \in E_1, \lambda(e) = \lambda_1(e)$ and $\forall e \in E_2, \lambda(e) = \lambda_2(e)$; $\leq_p = \leq^1_p \cup \leq^2_p \cup \{(e_1, e_2) \mid e_1 \in E_1 \cap E_p, e_2 \in E_2 \cap E_p\}$ and η is defined as follows:*

(C1) *If $e \in S_{E_i}$ and $\eta_i(e)$ is defined, where $i \in \{1, 2\}$, then $\eta(e) = \eta_i(e)$.*
(C2) *Let $e \in S_{E_1}$, η_1 be undefined on e, and $\lambda(e) = (p!q, a)$. If there is an event $e' \in R_{E_2}$ such that η_2^{-1} is undefined on e' (i.e. $e' \notin \eta_2(S_{E_2})$), $\lambda(e') = (q?p, a)$ and $|\{f \in E \mid f \leq_p e, \lambda(f) = (p!q, a)\}| = |\{f' \in E \mid f' \leq_p e', \lambda(f') = (q?p, a)\}|$, then we set $\eta(e) = e'$. Clearly, if such an e' exists, it is unique.*
(C3) *η is undefined on all other events of S_E.*

If $\hat{m}$ is non-degenerate, then it is easy to see that $\hat{m}$ is a CMSC, and we define the concatenation *of $\hat{m}_1$ and $\hat{m}_2$, denoted by $\hat{m}_1 \cdot \hat{m}_2$, to be $\hat{m}$. Otherwise, concatenation is not defined.* □

When we write a series of concatenations $\hat{m}_1 \cdot \hat{m}_2 \cdot \hat{m}_3 \cdot \ldots$ we always mean a left-to-right application, i.e. the term $((\hat{m}_1 \cdot \hat{m}_2) \cdot \hat{m}_3) \ldots$. Note that concatenation is sensitive to the order in which it is made.

Let us now fix some notation for describing automata. Let A be a finite alphabet. A deterministic finite automaton (DFA) over A is a tuple $\mathcal{A} = (S, s_{in}, \delta, F)$ where S is a finite set of states, $s_{in} \in S$ is the initial state, $\delta : S \times A \to S$ is the transition function and $F \subseteq S$ is the set of final states. δ can be extended to $\delta' : S \times A^* \to S$ by defining $\delta'(s, \epsilon) = s$ and $\delta'(s, x.d) = \delta(\delta'(s, x), d)$ where $x \in A^*$ and $d \in A$. We say that a word $x \in A^*$ is accepted by $\mathcal{A}$ if $\delta'(s_{in}, x) \in F$. The language of $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of words in A^* accepted by $\mathcal{A}$.

We are now ready to define CMSGs. Let us fix a finite set of CMSCs M — we will call these *atomic* CMSCs. We say that a sequence of CMSCs $\hat{m}_1, \hat{m}_2, \ldots \hat{m}_n$ from M is *well-defined* iff the concatenation of the CMSCs in this order is defined. We say that it is *complete* if the concatenated CMSC has no unmatched send or receive events — i.e. it is an MSC.

Let Π be a finite alphabet and $h : \Pi \to M$ be a bijection. Given a word $x = d_0 \ldots d_n \in \Pi^*$, if $h(d_0), \ldots h(d_n)$ is well-defined, then x defines the CMSC $cmsc(x) = h(d_0) \cdot h(d_1) \ldots h(d_n)$. We say that x *is well-defined* if $h(d_0), \ldots, h(d_n)$ is well-defined and say x *is complete* if $cmsc(x)$ is an MSC.

Definition 4. *A* compositional message sequence graph *(CMSG) is a tuple $G = (\Pi, M, h, \mathcal{A})$ where Π is a finite alphabet, M is a finite set of CMSCs, $h : \Pi \to M$ is a bijection, and $\mathcal{A}$ is a DFA over Π. We require that every $x \in L(\mathcal{A})$ is well-defined and complete. The CMSG represents the set of MSCs $msc(G) = \{cmsc(x) \mid x \in L(\mathcal{A})\}$.* □

For example, consider the CMSG in Figure 1. It depicts a producer-consumer example where p sends messages continuously to c. At some point, c sends the "abort" signal to p requesting it to stop sending messages but this message takes an arbitrarily long time to reach p. The figure shows a typical behaviour and a CMSG which represents such scenarios.

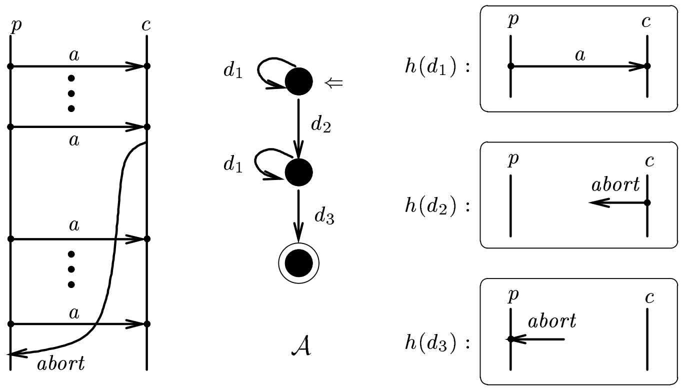

Fig. 1. A compositional message sequence graph

It turns out that given a structure $G = (\Pi, M, h, \mathcal{A})$, we can decide if it is a CMSG.

One way to simplify CMSGs is to partition Γ_m into two sets Γ_m^1 and Γ_m^2 and always use Γ_m^1-events for unmatched sends and receives, while using Γ_m^2 for matched sends and receives, in all the atomic CMSCs (as done in [GMP01]). Such a condition would ensure that when concatenating CMSCs, the resulting CMSC will never be degenerate. Also, one can show that any CMSG can be converted to an equivalent one over such restricted atomic CMSCs.

MSGs are just like CMSGs except that the atomic CMSCs are in fact MSCs. The class of CMSGs clearly extend that of MSGs — in fact, they are strictly more powerful. It can be shown that the behaviour illustrated in Figure 1 cannot be represented by MSGs nor is a regular MSC-language (as defined in [HMKT00b]).

We use monadic second order logic (MSO) to describe properties of MSCs. This logic is defined as follows: We have at our disposal a countable number of first-order variables $\{x, y, z, \ldots\}$ and second-order variables $\{X, Y, Z, \ldots\}$ — the former will be interpreted over events while the latter over subsets of events of an MSC. The atomic formulas are of the kind $(x \to y)$ (meaning that x is a send-event whose corresponding receive-event is y), $(x \leq_p y)$ (meaning x and y are events of p and x is before y in the total order of process p), $Q_r(x)$, where $r \in \Sigma$ (meaning the label of the event x is r), and $(x \in X)$. Other formulas are formed using the boolean connectives $\vee$ and $\neg$ and using quantification over first-order and second-order variables.

This logic is a powerful structural logic and can express many interesting properties of MSCs. It can be shown, for example, that the temporal logic TLC over MSCs considered in [Pel00] can be embedded into MSO.

Let φ be a formula and $m = (E, \{\leq_p\}_{p\in\mathcal{P}}, \lambda, \eta)$ be an MSC. An interpretation of a set of first-order and second-order variables V is a function I that assigns to each first-order variable an event in E, and assigns a set of events of E to each second-order variable. Semantics of when m satisfies φ under an interpretation I is defined in the obvious way and leads to the definition of when m satisfies a sentence φ.

3 CMSGs and regular representative linearizations

We now show that the class of MSC-languages represented by CMSGs is the same as that definable using a regular set of representative linearizations.

Lemma 1. *Let $\mathcal{L}$ be a collection of MSCs over $\mathcal{P}$. There exists a CMSG $G = (\Pi, M, h, \mathcal{A})$ that represents $\mathcal{L}$ iff there exists a regular set $L \subseteq \Sigma^*$ of well-formed words which is a representative linearization of $\mathcal{L}$.*

Proof: Let $G = (\Pi, M, h, \mathcal{A})$ be a CMSG such that $msc(G) = \mathcal{L}$. For every $\hat{m} \in M$, let us fix one linearization of $\hat{m}$, $\widehat{lin}(\hat{m})$. Let $L = \{\widehat{lin}(h(d_1)) \ldots \widehat{lin}(h(d_n)) \mid d_1 \ldots d_n \in L(\mathcal{A})\}$. It is easy to see that L is a regular representative linearization of $\mathcal{L}$. The other direction follows from [GMP01]. Let $L \subseteq \Sigma^*$ be a representative linearization of $\mathcal{L}$, and L be a regular language accepted by a DFA $\mathcal{B} = (S, s_{in}, \delta, F)$ over Σ. For every $r \in \Sigma_q$, let M have a CMSC with a *single event* r labelled r. One can now use $\mathcal{B}$ itself to define a CMSG which has L as a representative linearization. ☐

Let us now turn to the model checking problem for CMSGs: Given a CMSG G (or a regular representative linearization L), and a MSO sentence φ, do all the MSCs represented by G satisfy φ? We show that this problem is decidable. We exhibit two proofs of this theorem, one which extends the proof of decidability of model-checking MSGs [Mad01] and the other which extends the proof of decidability of model-checking for regular MSC-languages [HMKT00b].

A CMSC $m = (E, \{\leq_p\}_{p\in\mathcal{P}}, \lambda, \eta)$ is said to be b-memory bounded, where $b \in \mathbb{N}$, if the number of unmatched sends of any type is at most b, i.e., $\#^{unm}_{(p!q,a)}(m) \leq b$. A sequence of CMSCs $m_1, m_2, \ldots, m_k$ is b-memory bounded if for all its prefixes $m_1, m_2, \ldots, m_i$, the CMSC $m_1 \cdot m_2 \ldots \cdot m_i$ is b-memory bounded, where $i \leq k$. A CMSG $G = (\Pi, M, h, \mathcal{A})$ is b-memory bounded if for every $d_1 d_2 \ldots d_n \in L(\mathcal{A})$, the sequence of CMSCs $h(d_1), h(d_2), \ldots, h(d_n)$ is b-memory bounded. The fact that a CMSG defines a collection of MSCs forces it to be b-memory bounded:

Lemma 2. *For every CMSG G, there exists $b \in \mathbb{N}$, which is computable, such that G is b-memory bounded.*

Proof: Let $G = (\Pi, M, h, \mathcal{A})$ be a CMSG and let $\mathcal{A} = (S, s_{in}, \delta, F)$. For any state $s \in S$ which is reachable from the initial state and from which the final state can be reached, one can show that all CMSCs defined on a path from the initial state to s must have the same number of unmatched receives (from the fact that all accepting paths define MSCs). It then follows that if b is the maximum number of unmatched receives at any such state, then G is b-memory bounded. $\square$

The following lemma will be used to show that the model checking problem for CMSGs is decidable.

Lemma 3. *Let Π be a finite alphabet, M be a set of CMSCs and $h : \Pi \to M$ be a bijection. Let φ be an MSO sentence and $b \in \mathbb{N}$. Then the collection of words $w = d_1 \dots d_n \in \Pi^*$ such that $h(d_1), \dots h(d_n)$ is well-defined, complete and b-memory bounded, and $cmsc(w) \models \varphi$, is a regular subset of Π^*.*

Proof: The proof follows the corresponding proof for MSCs in [Mad01]. Let φ be an MSO formula with free second-order variables $V_\varphi = \{X_1, \dots X_k\}$ (assume for now that we have only second-order variables). Then let $\Pi_\varphi = \{(d, I) \mid d \in \Pi, I : V_\varphi \to 2^E,$ where E is the set of events of $h(d)\}$. Each letter in Π_φ encodes along with a letter $d \in \Pi$, an interpretation of the free variables over the events of the CMSC corresponding to d. The idea is to construct an automaton $\mathcal{A}_\varphi$ which will accept a word $(d_1, I_1) \dots (d_n, I_n)$ iff the sequence $h(d_1), \dots h(d_n)$ is well-defined, complete and b-memory bounded, and the MSC $m = cmsc(d_1 \dots d_n)$ under the combined interpretation defined by $I_1, \dots I_n$ on the events of m, satisfies φ. This is done inductively on the structure of the formula.

First, the set of all words $d_1 \dots d_n \in \Pi^*$ such that $h(d_1), \dots h(d_n)$ is a well-defined and complete b-memory bounded sequence, is regular. We run an automaton accepting this in parallel with the automaton we construct.

The atomic formula $x \to y$ is the hardest to handle. The automaton will check the interpretation indeed maps it to singleton sets — assume it does. The automaton now has to check if y is the corresponding receive-event of x. Since the sequence we are reading is b-memory bounded, we will know the number of sends of the kind $(p!q, a)$ which are pending when x happens — let this be k (k is bounded by the sum of b and the maximum number of events in an atomic CMSC). The event y is the corresponding receive-event of x, if the number of receive-events of the kind $(q?p, a)$ between x and y is k. For each atomic CMSC, we know a priori the number of receive-events of this kind it has. Since the number of atomic CMSCs between the atomic CMSCs which x and y belong to that have at least one receive-event of this kind is bounded, we can construct with some bookkeeping an automaton which checks this property.

Other atomic formulas are easy to handle; formulas obtained by disjunction, negation and existential quantification can be handled by using the fact that automata are closed under union, complement and projection respectively. $\square$

Theorem 1. *The model checking problem for CMSGs against MSO-formulas is decidable.*

Proof: Let $G = (\Pi, M, h, \mathcal{A})$ be a CMSG and φ be an MSO-formula. From Lemma 2, it follows that there exists $b \in \mathbb{N}$ (which can be computed) such that

G is b-memory bounded. Construct using Lemma 3, an automaton $\mathcal{A}_{\varphi}^{b}$ over Π^* that accepts a word $w = d_1 \ldots d_n$ iff $h(d_1), \ldots h(d_n)$ is well-defined, complete and b-memory bounded, and $cmsc(w) \models \varphi$. Clearly, all MSCs in $msc(G)$ satisfy φ iff $L(\mathcal{A}) \subseteq L(\mathcal{A}_{\varphi}^{b})$, which can be checked. $\square$

Let us now turn to linearizations and use them to solve the model checking problem. Let $w \in \Sigma^*$ be a well-formed word and $b \in \mathbb{N}$. w is said to be *b-bounded* if for every prefix x of w, the difference between the number of send-events in x of the type $(p!q, a)$ and the number of receive-events of the same type is at most b. A language is said to be b-bounded if each word in it is b-bounded.

Lemma 4. *Given an MSO-formula φ and $b \in \mathbb{N}$, the set of all well-formed words $w \in \Sigma^*$ such that w is b-bounded and $msc(w) \models \varphi$ is regular.*

Proof: From φ we construct, using the technique in [HMKT99, Lemma 4.1], an MSO formula φ_{str}^{b} over finite words in Σ^* such that for $w \in \Sigma^*$, $w \models \varphi_{str}^{b}$ iff w well-formed, b-bounded and $msc(w) \models \varphi$. The required conclusion follows since the strings described by MSO formulae on words form a regular set [Tho90]. The formula φ_{str}^{b} is constructed so that it matches the send and receive events of the MSC corresponding to the word, and dynamically interprets φ using this matching and the local total orders. $\square$

We can now give another proof of the result that model checking CMSGs against MSO specifications is decidable using linearizations.

Alternative proof of Theorem 1: In view of Lemma 1, let the CMSG be presented as a regular representative linearization L via a DFA $\mathcal{B}$ accepting L. Since any regular language of well-formed words is b-bounded (see [HMKT00b]), where b can be computed, we can find a b such that L is b-bounded. Construct using Lemma 4, an automaton $\mathcal{B}'$ that accepts all b-bounded words which represent MSCs that satisfy φ. Clearly, all the MSCs represented by L satisfy φ iff $L(\mathcal{B}) \subseteq L(\mathcal{B}')$, which is decidable. $\square$

Another important (albeit simple) consequence of dealing with representative linearizations is that it allows us to show decidability of the *linearization model checking problem* [AY99] for *trace-closed* or linearization-closed specifications. The linearization model checking problem for CMSGs is: Given a CMSG G and a regular language L over Σ, is $lin(msc(G)) \subseteq L$? This problem is known to be undecidable even for MSGs [AY99]. However, if L is linearization-closed, we can effectively solve the problem: Using Lemma 1, construct L' which is a regular representative linearization of $msc(G)$. It is now easy to verify that all linearizations of MSCs represented by G belong to L iff $L' \subseteq L$.

Also, the above procedure works in time polynomial in the sizes of the given CMSG and the specification automaton. The ease of the above model-checking problem raises the hope for defining temporal logics whose models are trace-closed (akin to LTrL in the setup of Mazurkiewicz traces [TW97]), as these logics may yield a fast verification procedure. The logic m-LTL introduced in [MR00] and the logic TLC for MSCs [Pel00] can be seen as starting points for this endeavour.

4 Extended CMSGs

In the definition of concatenation of two CMSCs $\hat{m}_1$ and $\hat{m}_2$ (Definition 3), we required that $\hat{m}_1$ have no unmatched receives. We can drop this requirement and define concatenation so that it matches unmatched receive-events of $\hat{m}_1$ with corresponding unmatched send-events in $\hat{m}_2$ (if any). This relaxed definition is the same as that of Definition 3 except that condition (C2) is replaced by:

(C2′) Let $e \in S_E$, η_1 and η_2 be undefined on e, and $\lambda(e) = (p!q, a)$. If there is an event $e' \in R_E$ such that η_1 and η_2 are undefined on e', $\lambda(e') = (q?p, a)$ and $|\{f \in E \mid f \leq_p e, \lambda(f) = (p!q, a)\}| = |\{f' \in E \mid f' \leq_p e', \lambda(f') = (q?p, a)\}|$, then we set $\eta(e) = e'$.

Again, this concatenation is defined only if the resulting structure is a CMSC (i.e. the resulting causal order is a partial-order and is non-degenerate).

We can now define *extended CMSGs* (XCMSG) which are exactly like CMSGs (see Definition 4) except that the MSCs represented by them are defined using the relaxed notion of concatenation above. As in a CMSG, we require that the accepting paths of the XCMSG define MSCs.

Again, given a structure $G = (\Pi, M, h, \mathcal{A})$, one can decide if it is a XCMSG. Also, the class of MSCs definable in this fashion is strictly larger than the class defined by CMSGs. For example, Figure 2 illustrates an XCMSG and a typical MSC represented by it. The figure describes a scenario where three processes p, q and r exchange messages. The channel from p to r is a slow data channel while the channels from p to q and q to r are fast. Consequently, the messages b followed by c overtake an arbitrary number of messages a from p to r. It is not difficult to see that there is no equivalent CMSG representation for it.

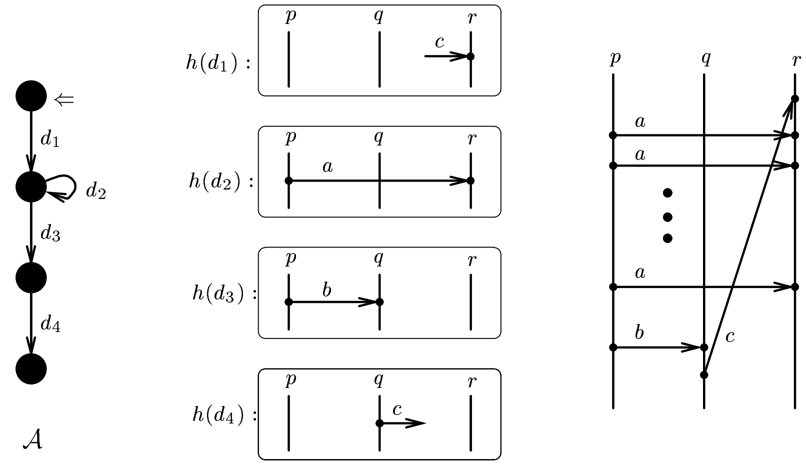

Fig. 2. An extended CMSG

However, XCMSGs are amenable to model checking against MSO specifications. Analogous to the definition of b-memory boundedness for CMSCs, one can define the notion of (b', b)-*memory bounded* CMSC, where $b, b' \in \mathbb{N}$. A CMSC m is (b', b)-memory bounded if the number of unmatched sends of any type is at most b (i.e., it is b-memory bounded) and the number of unmatched receives of any type is bounded by b'. We can extend this notion to sequences of CMSCs and to XCMSGs. It then turns out that any XCMSG is (b', b)-memory bounded for some $b, b' \in \mathbb{N}$ and we have a result analogous to Lemma 3:

Lemma 5. *Let Π be a finite alphabet, M be a set of CMSCs and $h : \Pi \to M$ be a bijection. Let φ be an MSO formula and $b, b' \in \mathbb{N}$. Then the collection of words $w = d_1 \ldots d_n \in \Pi^*$ such that $h(d_1), \ldots h(d_n)$ is well-defined, complete and (b', b)-memory bounded, and $cmsc(w) \models \varphi$, is a regular subset of Π^*.* $\square$

We then have

Theorem 2. *The model checking problem for XCMSGs against MSO specifications is decidable.* $\square$

The example in Figure 2 also illustrates the fact that there is no regular representative linearization of the MSCs represented by the given XCMSG. It turns out that the notion analogous to linearizations in this setting is that of a *semi-linearization.* For an MSC $m = (E, \{\leq_p\}_{\{p \in \mathcal{P}\}}, \lambda, \eta)$, an *event semi-linearization* of m is a linear order on E which extends the orders $\leq_p$ for each $p \in \mathcal{P}$. Note that the order *need not* extend the causal ordering $\leq$ — a receive-event could be ordered before its corresponding send-event. The semi-linearizations of an MSC m is the set of sequences $\lambda(e_1), \ldots, \lambda(e_k)$ where $e_1, \ldots, e_k$ is an event semi-linearization of m.

Again, while an MSC defines a non-empty set of semi-linearizations, a semi-linearization w corresponds to a unique MSC, denoted by $msc(w)$. For a word $w \in \Sigma^*$, we say w is *semi well-formed* if there is an MSC for which w is a semi-linearization We can now represent a collection of MSCs through a set of representative semi-linearizations. The following is analogous to Lemma 1.

Lemma 6. *Let $\mathcal{L}$ be a collection of MSCs over $\mathcal{P}$. Then, there exists an XCMSG $G = (\Pi, M, h, \mathcal{A})$ which represents $\mathcal{L}$ iff there exists a regular set $L \subseteq \Sigma^*$ of semi well-formed words which is a representative semi-linearization of $\mathcal{L}$.*

Proof: Given an XCMSG G, we can fix some linearization of each atomic CMSC and realize a set of representative semi-linearizations as a homomorphic image of the language accepted by $\mathcal{A}$, where each letter is mapped to the fixed linearization of the CMSC corresponding to it. Since we finish all the events of a CMSC before moving to another CMSC, it is clear that the resulting words respect the local partial orders of the individual CMSCs. Hence this set is semi well-formed. The proof of the converse is similar to that of Lemma 1. $\square$

Also, a result analogous to Lemma 4 goes through smoothly. A semi well-formed word $w \in \Sigma^*$ is said to be (b', b)-*bounded* if for every prefix x of w, the number of unmatched send-events in x of any type is at most b and the number of unmatched receive-events in x of any type is at most b'. A language is said to be (b', b)-bounded if each word in it is (b', b)-bounded.

Lemma 7. *Given an MSO sentence φ and $b, b' \in \mathbb{N}$, the set of all semi well-formed words $w \in \Sigma^*$ such that w is (b', b)-bounded and $msc(w) \models \varphi$ is regular.*

We can also show that any regular language of semi well-formed words is (b', b)-bounded for some $b, b' \in \mathbb{N}$. Then, we can use Lemma 7 to give an alternative proof of Theorem 2.

References

[AEY00] R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence charts. In *Proc. 22nd Intl. Conf. on Software Engg.*, pages 304–313, 2000.

[AY99] R. Alur and M. Yannakakis. Model checking of message sequence charts. In *Proc. CONCUR '99*, vol. 1664 of *LNCS*. Springer-Verlag, 1999.

[GMP01] E. Gunter, A. Muscholl, and D. Peled. Compositional message sequence charts. In *Proc. TACAS'01*, vol. 2031 of *LNCS*, pages 496–511. Springer-Verlag, 2001.

[HMKT99] J.G. Henriksen, M. Mukund, Narayan Kumar, and P.S. Thiagarajan. Towards a theory of regular MSC languages. *BRICS Report RS-99-52, Department of Computer Science, Aarhus University, Denmark*, 1999.

[HMKT00a] J.G. Henriksen, M. Mukund, Narayan Kumar, and P.S. Thiagarajan. On message sequence graphs and finitely generated regular MSC languages. In *Proc. ICALP '00*, vol. 1853 of *LNCS*. Springer-Verlag, 2000.

[HMKT00b] J.G. Henriksen, M. Mukund, Narayan Kumar, and P.S. Thiagarajan. Regular collections of Message Sequence Charts. In *Proc. MFCS '00*, vol. 1893 of *LNCS*. Springer-Verlag, 2000.

[ITU97] ITU-TS Recommendation Z.120. Message sequence chart (MSC). *ITU-TS*, 1997.

[Mad01] P. Madhusudan. Reasoning about sequential and branching behaviours of message sequence graphs. In *Proc. ICALP '01*, vol. 2076 of *LNCS*. Springer-Verlag, 2001.

[MM01] P. Madhusudan, B. Meenakshi. Beyond Message Sequence Graphs. *Institute of Mathematical Sciences Technical Report*, IMSc./2001/09/51. Available at `http://www.imsc.ernet.in/~madhu` or `/~bmeena`.

[MR00] B. Meenakshi and R. Ramanujam. Reasoning about message passing in finite state environments. In *Proc. ICALP '00*, vol. 1853 of *LNCS*. Springer-Verlag, 2000.

[Pel00] D. Peled. Specification and verification of message sequence charts. In *Proc. IFIP FORTE/PSTV*, 2000.

[RGG96] E. Rudolph, P. Graubmann, and J. Grabowski. Tutorial on message sequence charts. In *Computer Networks and ISDN Systems—SDL and MSC, Vol. 28*, 1996.

[Tho90] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 165–191, 1990.

[Tho97] W. Thomas. Languages, automata, and logic. *Handbook of Formal Language Theory*, III:389–455, 1997.

[TW97] P.S. Thiagarajan and I. Walukiewicz. An expressively complete linear time temporal logic for Mazurkiewicz traces. In *Proc. 12th IEEE Conf. on Logic in Computer Science*. IEEE Computer Society, 1997.