Reasoning about Sequential and Branching Behaviours of Message Sequence Graphs

P. Madhusudan

Institute of Mathematical Sciences, C.I.T. Campus, Taramani, Chennai, India.
madhu@imsc.ernet.in

Abstract. We study the model-checking problem of message-sequence graphs (MSGs). In the sequential setting, we consider the set of message-sequence charts (MSCs) represented by an MSG and tackle specifications given in monadic second-order logic. We show that this problem, without any restrictions on the MSGs, is decidable. We then turn to branching behaviours of MSGs, define a notion of an unfolding of an MSG, and show that the model-checking problem on unfoldings is also decidable. Our results are stronger and imply that, over an appropriate universe, satisfiability and synthesis of MSCs and MSGs, respectively, are decidable.

1 Introduction

Message sequence charts (MSC) are a popular visual formalism used to describe design requirements of concurrent message-passing systems [16, 10]. They are used to describe behaviours of systems in early models of system design. An MSC describes a single partially-ordered execution of the system and a collection of these diagrams formalize the set of possible scenarios exhibited by a model. These diagrams are usually used in the design of distributed telecommunication protocols and are also known as timing-sequence diagrams or message-flow diagrams [16].

Message sequence graphs (MSGs), also known as MSC-graphs, are a convenient mechanism to specify an infinite collection of MSCs. These graphs basically have a finite set of atomic MSCs and the MSG defines some (regular) ways to combine them using concatenation, choice, and repetition. As a model of concurrent systems they are interesting as they exhibit a sequential composition of parallel behaviours, as opposed to most other models which are parallel compositions of sequential machines.

Since MSCs and MSGs have a formal semantics, they are amenable to analysis. The issue of *model-checking* MSCs and MSCs defined by an MSG has been an area of active study in recent years [1, 11, 14, 13, 2, 9]. Automated verification of MSGs could enable engineers to check whether the design requirements have been met and whether it describes the set of desirable (or undesirable) behaviours correctly.

Alur and Yannakakis [2] give a detailed account of the model-checking problem for MSGs and the corresponding complexity involved. They consider the problem of deciding whether the *linearizations* of events of all MSCs defined by an MSG satisfy some linear-time property. As they show, the problem is undecidable even if the specification of good behaviours is given by a deterministic finite automaton. Consequently, in [2] the authors consider a syntactically identifiable sub-class of MSGs called *bounded MSGs*, for which they prove that the linearizations form a regular set of sequences, and thereby solve the model-checking problem for this class.

In [7, 9], the authors define a notion of *regular* MSC-languages which are recognizable by distributed communicating finite-state devices and study their properties. The global device is forced to be finite-state by constraining channels to have a bounded capacity. However, this class of languages are not contained in, nor contain, the class of languages defined by MSGs [8]. It is also shown that bounded MSGs define only regular MSC-languages and hence it follows from their results that one can decide specifications written in MSO on such bounded MSGs.

In this paper, the main point of departure from previous work is to completely ignore linearizations of events of an MSC and instead work directly on the graphical (visual) description of the charts. The main result we show is that the problem of deciding whether the set of all MSCs defined by an MSG satisfy a property given as a MSO formula is decidable. The solution does not assume *any* restriction on the atomic MSCs— they can exhibit non-FIFO behaviour, the messages can have labels, processes can have local actions, etc. Also, the MSGs are not restricted in any way— they can be unbounded in the sense of [2].

In the second half of our paper, we show how to handle branching-time behaviours of MSGs. We define the notion of an *unfolding* of an MSG which we believe is useful to reason about its behaviours, and then show that this infinite graph has a decidable monadic theory. This then enables the verification of properties such as asking whether every (finite) MSC defined by a given MSG is always extendable to another defined by the MSG, which satisfies a property expressed in MSO.

Our results are in fact are stronger. Let us fix any finite set of atomic finite MSCs and consider the universe of all MSCs generated by this set using concatenation. Then we show that a restricted form of satisfiability— whether there is an MSC which is formed using the atomic MSCs which satisfies a given formula— is decidable. We also show that the problem of synthesis of MSGs— whether there is an MSG whose unfolding satisfies a given specification— is decidable.

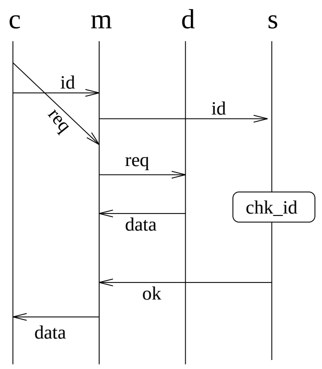
Our results and the simplicity of the concepts in our proofs suggest that *structural* logics are perhaps more amenable to reasoning about MSCs than logics which specify properties of the linearizations of an MSC. Logics such as monadic second-order logic, or an adaptation of the logic LTrL [19], or *localised logics* such as mLTL [15] can all be seen as structural logics which when defined over MSCs appropriately will be decidable.

Turning to the theoretical interest of our results, the study of infinite graphs which have a decidable monadic theory is an interesting area of research [5, 20, 18]. Our results exhibit a class of infinite graphs, which arise naturally in the study of concurrent systems, that do have a decidable monadic theory. A careful reader will note that our proofs do not assume much about MSCs and the results can be extended to the setting where we work with any set of atomic finite n-boundaried graphs. Also, our results are stronger as we show that we can work over an infinite universe of infinite graphs and the monadic second-order formulas describe only "regular" subsets of this class. We can hence answer questions like whether there is a graph in such a class which satisfies a property given in MSO.

In the next section we introduce our problem setting and deal with the sequential behaviours of MSGs in Section 3. In Section 4 we define the notion of an unfolding of an MSG and show how we can reason about it. Due to space restrictions we provide only the main ideas— more details can be found in [12].

2 Preliminaries

A message sequence chart (MSC) is a diagram which depicts a single partially-ordered execution of a distributed message-passing system. For example, consider the MSC depicted on the left-hand side of Figure 1. It consists of four processes— a customer c, a manager m, a database server d and a security checker s. The MSC depicts a particular scenario where the customer requests for data with his identity. Though the customer sends the request first followed by the id, the messages get delivered in reverse order. After receiving the id, the manager sends it to the security checker for verification. It also sends the request to the database. After receiving messages from them, the manager serves the customer with the data. Note that the security process has a local action where it checks the id. The MSC represents one partially-ordered behaviour of the way the messages were sent and received— some events are ordered while others, like m receiving the request and s receiving the id, are not.



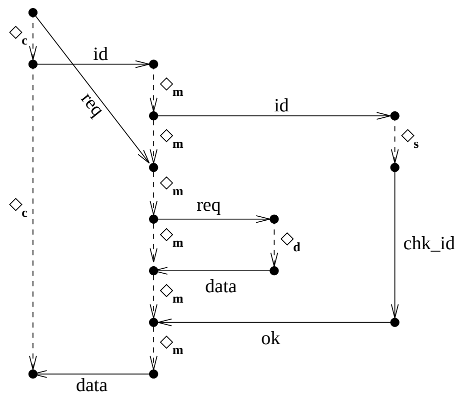


Fig. 1. An MSC and its associated graph

We work with MSCs by considering them as labelled directed graphs. For example, Figure 1 illustrates an MSC and the corresponding graph representation for it. The graph representing the MSC has vertices at points where the events take place, and edges denote messages as well as local actions. A vertex with an outgoing message edge is to be viewed as the action of sending a message, while the vertex at the other end is seen as the point where the message is received. The vertices of each process p are connected in the natural way according to the local linear flow of time, and these edges are labelled by a special symbol $\diamond_p$ (denoted by dotted lines).

Let us now define MSCs formally. Let us fix a finite set of processes $\mathcal{P}$ and let p, q, r range over $\mathcal{P}$. Let us also fix a finite set of local actions Γ_l and a set of messages Γ_m. For each $p \in \mathcal{P}$, let us also have a *next-state symbol* $\diamond_p$ and let $\Gamma_n = \{\diamond_p \mid p \in \mathcal{P}\}$. We fix all these for the rest of the paper and assume that all these sets are pairwise disjoint. Let $\Gamma = \Gamma_l \cup \Gamma_m \cup \Gamma_n$. In the above example, $\mathcal{P} = \{c, m, d, s\}$, $\Gamma_m = \{id, req, data, ok\}$, $\Gamma_l = \{chk\_id\}$, $\Gamma_n = \{\diamond_c, \diamond_m, \diamond_d, \diamond_s\}$.

Let us fix some notations on directed graphs. A directed graph G over a finite label-set L is a tuple (V, E) where V is a set of vertices and $E \subseteq V \times L \times V$. By $v \xrightarrow{l} v'$ we mean $(v, l, v') \in E$. For a set $L' \subseteq L$ and $v \in V$, we denote by $out_{L'}(v)$ the set of all vertices such that there is an L'-labelled edge from v to it— i.e. $out_{L'}(v) = \{v' \in V \mid \exists l \in L' : (v, l, v') \in E\}$. Similarly, $in_{L'}(v)$ denotes the set of vertices which have an L'-labelled edge to v.

For a linear-order $\leq$ on a finite set S, let $\lessdot$ denote the *covering relation for* $\leq$: for any $x, y \in S$, $x \lessdot y$ iff ($x \leq y$ and $\forall z \in S$, if $x \leq z \leq y$, then $z = x$ or $z = y$). Also, for any $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, \ldots, n\}$.

Definition 1. *A message sequence chart (MSC) is a tuple* $(\{V_p\}_{p \in \mathcal{P}}, E)$ *where, if* $V = \bigcup_{p \in \mathcal{P}} V_p$*, then:*

- *V is finite*
- $E \subseteq [\bigcup_{p \in \mathcal{P}} (V_p \times (\Gamma_l \cup \Gamma_n) \times V_p)] \ \cup \ [\bigcup_{p,q \in \mathcal{P}, p \neq q} (V_p \times \Gamma_m \times V_q)]$
- *For each $p \in \mathcal{P}$, there is a linear-order $\leq_p$ on V_p such that $\forall v, v' \in V_p$: ($v \lessdot v'$ iff $\exists l \in \Gamma_l \cup \Gamma_n : v \xrightarrow{l} v'$).*
- *$out_{\Gamma_l \cup \Gamma_n}(v)$ is a singleton set.*
- *For each $v \in V$, one of the three sets $out_{\Gamma_l}(v)$, $out_{\Gamma_m}(v)$ and $in_{\Gamma_m}(v)$ is a singleton set and the other two are empty.*
- *$G = (V, E)$ is acyclic.* □

The second condition demands that the edges within a process be labelled by local actions or next-state symbols while edges between different processes are labelled using the message alphabet. The next two conditions demand that the events of each local process are totally ordered and are connected in this order using exactly one edge. We also demand that at each point, exactly one of the following happen: a message is received, a message is sent, or there is a local action. We finally require the events to be partially ordered.

Though we do not require it, for technical ease let us assume that each V_p is nonempty. For an MSC m and a process $p \in \mathcal{P}$, the linear order on V_p satisfying the condition above is clearly unique, and we denote this by $\leq_p$. Let the minimum vertex of V_p in m under this ordering be denoted by first_p^m and the maximum vertex by last_p^m. We identify an MSC with its equivalence class with respect to isomorphism— i.e. two graphs will represent the same MSC if they are isomorphic.

Concatenation of two MSCs is done by (asynchronously) concatenating the individual process evolutions of the MSCs. For example, Figure 2 illustrates the concatenation of two MSCs. Formally,

Definition 2. *Let $m_1 = (\{U_p\}_{p\in\mathcal{P}}, E_1)$ and $m_2 = (\{V_p\}_{p\in\mathcal{P}}, E_2)$. Let U_p and V_p be disjoint sets (if they are not, relabel vertices to make them disjoint). The concatenation of m_1 and m_2, denoted by $m_1 \cdot m_2$ is the MSC $(\{W_p\}_{p\in\mathcal{P}}, E)$ where $W_p = U_p \cup V_p$ and $E = E_1 \cup E_2 \cup \{(\mathit{last}_p^{m_1}, \diamond_p, \mathit{first}_p^{m_2}) \mid p \in \mathcal{P}\}$.* □

Note that concatenation is associative.

We now fix notations for talking about finite-automata. A deterministic finite-state automaton (DFA) $\mathcal{A}$ is a structure $(\Sigma, Q, q_{in}, \delta, F)$, where Σ is a finite alphabet, Q is a finite set of states, $q_{in} \in Q$ is the initial state, $\delta : Q \times \Sigma \to Q$ is the transition function and $F \subseteq Q$ is the set of final states. The transition function δ can be extended to a function $\delta' : Q \times \Sigma^* \to Q$ by defining $\delta'(q, a) = \delta(q, a)$ and $\delta'(q, x \cdot a) = \delta(\delta'(q, x), a)$. We say that $\mathcal{A}$ accepts a word $x \in \Sigma^*$ if $\delta'(q_{in}, x) \in F$. The language of $\mathcal{A}$, denoted by $L(\mathcal{A})$ is the set of strings it accepts.

Message sequence graphs are a convenient mechanism to define a collection of MSCs. We use a slightly different but equivalent terminology than usually found in the literature.

Definition 3. *A message sequence graph (MSG) is a structure $\mathcal{M} = (\Sigma, \mathcal{A}, M, h)$ where Σ is a finite alphabet, $\mathcal{A}$ is a finite automaton on Σ, M is a finite collection of MSCs over $\mathcal{P}$ and $h : \Sigma \to M$ is a bijection that identifies an MSC in M for each symbol in Σ.* □

We specify properties of MSCs as well as other directed graphs using *monadic second-order logic* (MSO). Let Π be any finite alphabet. Then the monadic second-order logic over directed graphs with edge labels from Π is defined as follows: let there be an infinite number of first-order variables $\{x, y, z, \ldots\}$ and second-order variables $\{X, Y, Z, \ldots\}$. The set of MSO formulas is the smallest set containing:

- The atomic formulas of the kind $x \in X$ and $x \xrightarrow{l} y$ where $l \in \Pi$.
- The formulas $\varphi \vee \psi$, $\neg\varphi$, $\exists x \varphi(x)$ and $\exists X \varphi(X)$, where φ and ψ are MSO formulas.
  $\varphi(x)$ ($\varphi(X)$) denotes a formula with free variable x (X).

Let $G = (V, E)$ be a (finite or infinite) graph where $E \subseteq V \times \Pi \times V$ and let φ be a formula. An *interpretation* for φ is a function I which assigns to each free first-order variable of φ a vertex in V and assigns to each free second-order variable of φ a subset of V. The semantics of a formula being true under an interpretation I is the usual one. A sentence is a formula φ without free variables. For a sentence φ and a graph G, we say G *satisfies* φ if φ is true in G and we denote this by $G \models \varphi$.

In the proofs, we work with a different set of formulas that is as expressive, but which has no formulas with first-order variables (as done in [18]). This restricted syntax has atomic formulas $X \subseteq Y$, *Singleton*(X) and $X \xrightarrow{l} Y$. Other formulas are $\varphi \vee \psi$, $\neg \varphi$ and $\exists X \varphi(X)$. It is easy to see that this syntax is exactly as expressive as the original one.

3 Sequential behaviours of MSGs

Let us fix an MSG $\mathcal{M} = (\Sigma, \mathcal{A}, M, h)$ for the rest of this section. An MSC m is generated by $\mathcal{M}$ if there is a string $x \in \Sigma^*$ accepted by $\mathcal{A}$ such that the concatenation of the MSCs in x is m.

Formally, for $x = a_0 a_1 \ldots a_n \in \Sigma^*$, let $msc_{M,h}(x) = h(a_0) \cdot h(a_1) \ldots h(a_n)$. When M and h are clear from context, we denote $msc_{M,h}(x)$ by $msc(x)$. The *MSC-language of* $\mathcal{M}$ is $\mathcal{L}(\mathcal{M}) = \{msc(x) \mid x \in L(\mathcal{A})\}$. We say that $\mathcal{M}$ generates m if $m \in \mathcal{L}(\mathcal{M})$.

The sequential model-checking problem for MSGs is then the following: Given an MSO sentence φ over directed graphs labelled by Γ and an MSG $\mathcal{M}$, do all the MSCs generated by $\mathcal{M}$ satisfy φ?

MSO is an expressive mechanism to state the required structural properties of MSCs. For example, we can state in the context of the example in Figure 1 that if the manager sends the data, then there must be a corresponding confirmation of id from the security checker. It can also express a variety of safety and liveness conditions (like "if p sends a request, then it eventually gets an acknowledgement", "if the messages are delivered in FIFO fashion, then a property φ holds", etc.).

The main lemma we use to prove that the above problem is decidable is to show that for any property φ, the strings $x \in \Sigma^*$ such that $msc_{M,h}(x) \models \varphi$ form a regular subset of Σ^*. The proof follows the structure of Thomas' adaptation of the proof by Büchi [3] and Elgot [6] that classes of languages defined by monadic second-order formulas over finite strings correspond to the class of regular languages (see [17, 18]). This proof works by inductively associating with every formula $\varphi(X_1, \ldots, X_k)$ an automaton $\mathcal{A}_\varphi$ over words that encode interpretations of the variables over the structure and accept the word iff the structure, under the interpretation, satisfies φ.

Let φ be a MSO formula over MSCs. The idea is to augment the alphabet Σ to get a new alphabet which has letters of the form (a, I), where $a \in \Sigma$. Let x be a finite string over this alphabet, say $x = (a_0, I_0), \ldots (a_n, I_n)$. The MSC we are working on now is $m = msc_{M,h}(a_0 \ldots a_n)$. The second component of the alphabet will define an interpretation of the free variables of φ over the MSC m. The idea is that I_j will encode an interpretation of the variables over the MSC $h(a_j)$, for each $j \in \{0, \ldots, n\}$. Together, the I_j's will define a complete interpretation of the variables over the graph m.

We then follow the automata-theoretic approach to show inductively that for every formula φ, there is an automaton over a suitable alphabet which accepts a word iff the MSC defined by the word, under the interpretation of the free variables of φ over this MSC, satisfies the formula φ.

Now, let φ have k free (second-order) variables. For a letter $a \in \Sigma$, we require a way to encode all interpretations of the k variables over the MSC $h(a)$. Let $h(a)$ have r vertices. Then a 0-1 matrix Z with k rows and r columns can represent any such interpretation: the j^{th} vertex of $h(a)$ (according to some fixed ordering) belongs to the i^{th} variable iff the entry in Z on the i^{th} row and j^{th} column is 1.

Let us fix some notations for matrices. For any $k, r \in \mathbb{N}$, let $S_{k\times r}$ denote the set of all 0-1 matrices with k-rows and r-columns. For $Z \in S_{k\times r}$, we denote by $Z[i,j]$ ($i \in [k]$, $j \in [r]$), the element at the i^{th} row and j^{th} column in Z.

Lemma 1. *Let φ be an MSO property on directed graphs with edge labels from Γ. Let M be a finite set of MSCs, Σ be a finite alphabet and let $h : \Sigma \to M$ be a bijection. Let $L_\varphi^{\Sigma,M,h} = \{x \in \Sigma^* \mid msc_{M,h}(x) \models \varphi\}$. Then $L_\varphi^{\Sigma,M,h}$ is a regular subset of Σ^*. Moreover, one can effectively construct a finite-state automaton accepting $L_\varphi^{\Sigma,M,h}$.*

Proof: For each MSC $m = (\{V_p\}_{p\in\mathcal{P}}, E) \in M$, let $s(m)$ denote the number of vertices in m, i.e. $|V|$, where $V = \bigcup_{p\in\mathcal{P}} V_p$. Also fix some ordering on V.

For any $k \in \mathbb{N}$, let $\Sigma_k = \{(a, Z) \mid a \in \Sigma, Z \in S_{k\times s(h(a))}\}$. A letter $(a, Z) \in \Sigma_k$ defines an interpretation of k variables on subsets of vertices of $h(a)$: a vertex u of the MSC $h(a)$ is in X_i iff u is the j^{th} element of m and $Z[i,j] = 1$.

A string $x = (a_0, Z_0)\ldots(a_n, Z_n) \in \Sigma_k^*$ encodes an interpretation of the k variables on the MSC $h(a_0)\cdot h(a_1)\ldots h(a_n)$— this is given by taking the union of the individual interpretations. The claim we prove now is:

Claim For each MSO formula $\varphi(X_1,\ldots X_k)$ there is a deterministic finite-state automaton $\mathcal{A}_\varphi$ over the alphabet Σ_k which accepts a string x iff the MSC defined by x, under the interpretation defined by x, satisfies φ.

Proof of Claim We show the claim by induction on the structure of φ. It is easy to see that the atomic formulas $X \subseteq Y$ and $Singleton(X)$ can be checked by an automaton. For the atomic formula $X \xrightarrow{l} Y$ where $l \in \Gamma$, the automaton first checks whether the interpretations for X and Y are singleton sets and rejects the word if it is not. Let X be interpreted as $\{u\}$ and Y as $\{v\}$. If u and v belong to the same atomic MSC $h(a)$ for some (a, Z) in the input sequence, then clearly $u \xrightarrow{l} v$ iff $u \xrightarrow{l} v$ in $h(a)$. The automaton can decide this by a table-lookup and accept or reject the word accordingly.

We are left with the case where u and v belong to different atomic MSCs. If $l \in \Gamma_l \cup \Gamma_m$, then the automaton rejects. Otherwise let $l = \Diamond_p$. It checks now whether u and v belong to consecutive atomic MSCs m, m' in the input, and whether u is $last_p^m$ and v is $first_p^{m'}$. If so, it accepts the word and otherwise rejects it.

The other formulas are easy to tackle— disjunction, negation and existential quantification are handled by using the fact that DFAs are effectively closed under union, complement and projection. **End of claim**

From the Claim, it is clear that for any sentence φ, we can construct an automaton over Σ^* such that the automaton accepts a word x iff $msc_{M,h}(x) \models \varphi$. $\square$

We can now solve the model-checking problem for MSGs:

Theorem 1. *Given an MSG $\mathcal{M}$ and an MSO formula φ over MSCs, the problem of checking whether all MSCs generated by $\mathcal{M}$ satisfy φ is decidable.*

Proof: Let the MSG be $\mathcal{M} = (\Sigma, \mathcal{A}, M, h)$. Construct using Lemma 1, an automaton accepting $L_{\neg\varphi}^{\Sigma,M,h}$. This language consists of all sequences $x \in \Sigma^*$ such that $msc(x) \not\models \varphi$. Take its intersection with $\mathcal{A}$. Clearly, the intersection is empty iff all the MSCs generated by $\mathcal{M}$ satisfy φ. $\square$

Another consequence of Lemma 1 is that we can solve a related satisfiability problem:

Theorem 2. *Let M be a finite set of MSCs and φ be an MSO formula over MSCs. Then the problem of finding whether there is an MSC m, which can be expressed as a concatenation of MSCs in M, such that m satisfies φ, is decidable.*

Proof: Take an alphabet Σ, with $|\Sigma| = |M|$, take a 1-1 function $h : \Sigma \to M$, and construct, using Lemma 1, an automaton accepting the language $L_{\varphi}^{\Sigma,M,h}$. Clearly, there is an MSC m formed using MSCs in M which satisfies φ iff $L_{\varphi}^{\Sigma,M,h} \neq \emptyset$. $\square$

Observe that all the above results can be adapted to infinite concatenations as well. We can easily extend Definition 2 to define infinite concatenations of finite atomic MSCs in the natural way. We can extend the notion of an MSG such that the automaton provided accepts regular ω-languages over Σ. Then Lemma 1 smoothly extends to MSO formulas over infinite MSCs generated by the finite collection of finite atomic MSCs in the MSG. All we need to do is to work over *nondeterministic Büchi automata* [4, 17, 18] and use the property that the languages these automata accept are effectively closed under union, complement and projection. The analogous model-checking and satisfiability theorems for infinite MSCs will hold as well. Such an extension will facilitate formulating liveness properties of the system.

4 Branching behaviours of MSGs

In this section, our main goal is to define the unfolding of an MSG and show that its MSO theory is decidable. However, as in the previous section, we prove a stronger theorem which says that the set of all unfoldings which satisfy a formula is regular. Let us prove this lemma before we tackle MSGs.

Fix $D \in \mathbb{N}$ and a finite alphabet Σ. A *Σ-labelled D-ary tree* is a tuple (T, ρ) where T is the graph $G = (V_T, E_T)$ where $V_T = [D]^*$, $E_T = \{(x, x \cdot d) \mid x \in V_T, d \in [D]\}$ and $\rho : [D]^+ \to \Sigma$. (Note that the root is not labelled). Let M be a finite collection of (finite) MSCs and $h : \Sigma \to M$ be a bijection.

Definition 4 (Wrap of an MSC). *Let $a \in \Sigma$, $m = h(a) = (\{V_p\}_{p \in \mathcal{P}}, E)$. Then the* wrap of m *is the graph $G_w = (V_w, E_w)$ where $V_w = (\bigcup_{p \in \mathcal{P}} V_p) \cup \{f_a, l_a\}$ and $E_w = E \cup \{(f_a, *_a, \mathit{first}_p^m), (\mathit{last}_p^m, *_a, l_a) \mid p \in \mathcal{P}\}$.* $\square$

Note that the wrap of an MSC is just the MSC with two extra vertices, one of which is connected to the first nodes of the MSC and the last nodes of the MSC are connected to the other. Now we associate with any Σ-labelled D-ary tree (T, ρ), the graph obtained by replacing each vertex u of T by the wrap of $h(\rho(u))$.

Definition 5. *Let (T, ρ) be a Σ-labelled D-ary tree with $T = ([D]^*, E_T)$. Then the graph associated with (T, ρ) is the graph $G = (V, E)$ where $V = \{\langle u, v\rangle \mid u \in [D]^+$ and v is a vertex in $wrap(h(\rho(u)))\} \cup \{r\}$ and*
$$E = \{(\langle u, v\rangle, l, \langle u, v'\rangle) \mid v \xrightarrow{l} v' \text{ in } wrap(h(\rho(u)))\} \cup$$
$$\{(\langle u, l_a\rangle, *, \langle u', f_b\rangle) \mid u \to u' \text{ in } T \text{ and } \rho(u) = a, \rho(u') = b\} \cup$$
$$\{(r, *, \langle i, f_a\rangle) \mid i \in [D], \rho(i) = a\}.$$
□

Let $\Pi = \Gamma \cup \{*_a \mid a \in \Sigma\} \cup \{*\}$. We now work with MSO formulas over graphs labelled over Π. Consider such a formula φ. We show now that the set of Σ-labelled D-ary trees T such that the graph represented by T satisfies φ, is a regular tree-language (of Σ-labelled infinite trees).

The idea behind the proof is exactly the same as in the sequential setting. We extend the alphabet so that each symbol can now encode an interpretation of variables over the (wrap of the) MSC at the node. Then, we can show that with every formula $\varphi(X_1, \ldots, X_k)$, one can associate a *parity tree automaton* which accepts a tree iff the graph associated with the tree along with the interpretation of $X_1, \ldots, X_k$ coded in it, satisfies φ. (We do not define parity tree automata as we will not give proofs here— see [17, 18] for details.) We then have:

Lemma 2. *Let φ be an MSO property on directed-graphs with edge-labels from Π. Let $D \in \mathbb{N}$, M be a finite set of MSCs, Σ be a finite alphabet and let $h : \Sigma \to M$ be a bijection. Let $\mathcal{L}_\varphi^{\Sigma,M,h}$ be the set of all Σ-labelled D-ary trees (T, ρ) such that the graph associated with it satisfies φ. Then $\mathcal{L}_\varphi^{\Sigma,M,h}$ is a regular tree-language. Moreover, one can effectively construct a finite-state parity tree-automaton accepting $\mathcal{L}_\varphi^{\Sigma,M,h}$.* □

We do not give the proof here as the ideas are exactly the same as in the sequential case and the rest is only detail.

Let us now turn to MSGs. An MSG in this section will be a tuple $\mathcal{M} = (\Sigma, TS, M, h)$ where Σ, M and h are as before and where TS is a (finite) transition system (Q, q_{in}, δ) where Q is a finite set of states, $q_{in} \in Q$ is the initial state and $\delta \subseteq Q \times \Sigma \times Q$. Let us define the successor function as $succ(q) = \{(a, q') \mid (q, a, q') \in \delta\}$ for each $q \in Q$. For technical simplicity, let us assume that the number of successors for each state is the same— let $\forall q \in Q : |succ(q)| = D$, for some $D \in \mathbb{N}$. Let us also order the successors of each state and let $succ(q)[i]$ denote the i^{th} successor of q, where $i \in [D]$.

Fix an MSG $\mathcal{M} = (\Sigma, TS, M, h)$ for the rest of this section. The $(\Sigma \times Q)$-*unfolding* of $\mathcal{M}$ is the $(\Sigma \times Q)$-labelled D-ary tree (T, ρ) where $T = (V_T, E_T)$ and ρ is the unique labelling which satisfies:

- $\rho(i) = succ(q_{in})[i]$, for every $i \in [D]$
- If $\rho(x) = q$, then $\rho(x \cdot i) = succ(q)[i]$, for every $i \in [D], x \in V_T$.

The Σ-unfolding of $\mathcal{M}$ is obtained by projecting the labels of the above tree to the first component. Now the *unfolding of the MSG M* is the graph associated with the Σ-unfolding of $\mathcal{M}$. [1]

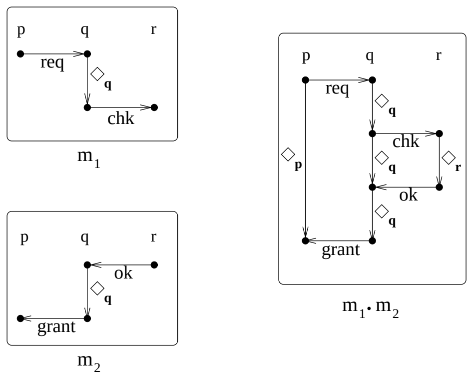


Fig. 2. Concatenation of MSCs

For example, for $\Sigma = \{a, b, c\}$ with $h(a) = m_1$ and $h(b) = m_2$, where m_1 and m_2 are the MSCs depicted in Figure 2, Figure 3 illustrates a transition system, part of its Σ-unfolding and the corresponding unfolding of the MSG.

Note that the notion of an unfolding does *not* imply that we have synchronous concatenation of MSCs. Indeed, as in the previous section, we can in the specification dynamically interpret the edges required for asynchronous concatenation and MSO formulas can then reason about the MSCs thus generated. The reason we use the above definition is to enable us to clearly describe the points at which the branching of behaviours takes place.

We can now show that the branching-time model-checking problem for MSGs is decidable:

Theorem 3. *Let $\mathcal{M}$ be an MSG and let φ be a MSO property on graphs labelled over Π. Then the problem of checking whether the unfolding of $\mathcal{M}$ satisfies φ is decidable.*

Proof: First construct, using Lemma 2, a parity tree-automaton $\mathcal{A}_\varphi$ that accepts exactly the Σ-labelled D-ary trees such that the associated graph satisfies φ. Now, TS can be converted to a tree automaton $\mathcal{A}_\mathcal{M}$ accepting the language consisting of a single tree which is the Σ-unfolding of $\mathcal{M}$. Now, clearly, there is a tree accepted by both $\mathcal{A}_\varphi$ and $\mathcal{A}_\mathcal{M}$ iff the unfolding of $\mathcal{M}$ satisfies φ. $\square$

We can now show that various branching-time properties of MSGs, like one saying that every finite MSC generated by the MSG is extendable to another generated by the MSG which satisfies a property φ on MSCs, is decidable. We can also prove an associated *synthesis* problem, which is an analogue of Theorem 2:

[1] Note that the unfolding of the MSG is independent of the way the successors of states of TS were ordered.

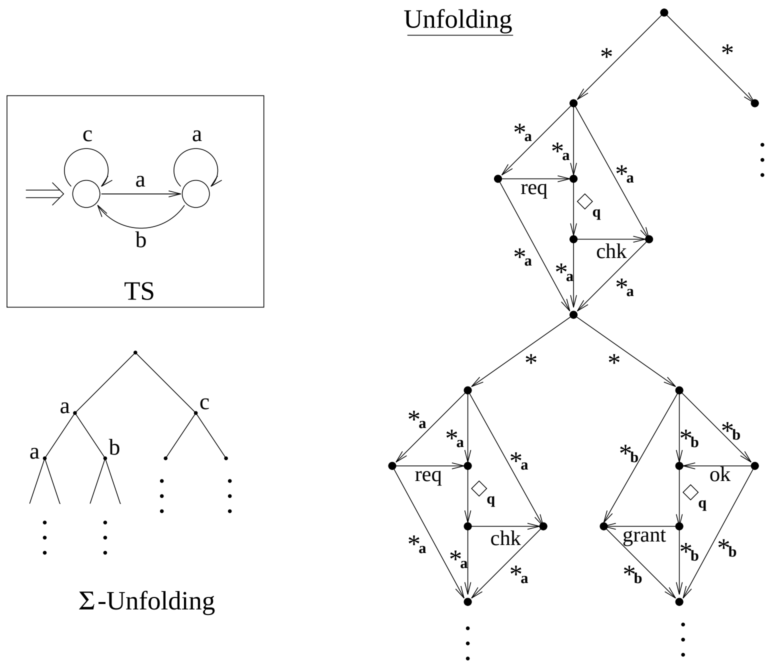


Fig. 3. A transition system, the Σ-unfolding and the unfolding

Theorem 4. *Let M be a finite set of MSCs, Σ be a finite set, $h : \Sigma \to M$ be a bijection, $D \in \mathbb{N}$ and φ be an MSO formula over Π. Then the problem of finding whether there is an MSG $\mathcal{M} = (\Sigma, TS, M, h)$ of branching degree at most D such that the unfolding of $\mathcal{M}$ satisfies φ, is decidable.*[2]

Proof: Construct, using Lemma 2, a tree automaton accepting the language $\mathcal{L}_{\varphi}^{\Sigma,M,h}$. Clearly, there is an unfolding formed using MSCs in M and respecting h which satisfies φ iff $\mathcal{L}_{\varphi}^{\Sigma,M,h} \neq \emptyset$. If the language is nonempty, we can use a regular run accepted by the automaton and construct a finite-state MSG whose unfolding satisfies φ. $\square$

This theorem can be used to synthesize MSGs which satisfy a property φ. The new edges added in the wrap constructions allow the formula to express properties of nondeterminism which can for example arise due to the fact that the environment is uncontrollable.

Acknowledgement: I would like to thank Wolfgang Thomas, who introduced me to decidable MSO theories of graphs beyond trees, and Meenakshi, for introducing me to MSCs and helping me understand the related literature.

[2] We need not even require D to be given— however the proof of this is out of the scope of this paper.

References

1. R. Alur, G.J. Holzmann, and D. Peled. An analyzer for message sequence charts. In *Software Concepts and Tools*, volume 17(2), pages 70–77, 1996.
2. R. Alur and M. Yannakakis. Model checking of message sequence charts. In *Proc. CONCUR '99*, volume 1664 of *LNCS*. Springer-Verlag, 1999.
3. J. R. Büchi. Weak second-order arithmetic and finite automata. In *Z. Math. Logik Grundl. Math.*, volume 6, pages 66–92, 1960.
4. J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Internat. Congr. Logic, Method and Philos. Sci. 1960*, pages 1–12, Stanford, 1962. Stanford University Press.
5. D. Caucal. On infinite transition graphs having a decidable monadic theory. In *Proc. 23rd ICALP*, volume 1099 of *LNCS*, pages 194–205. Springer-Verlag, 1996.
6. C.C. Elgot. Decision problems of finite automata and related arithmetics. In *Trans. Amer. Math. Soc.*, volume 98, pages 21–52, 1961.
7. 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.
8. 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*, volume 1853 of *LNCS*. Springer-Verlag, 2000.
9. J.G. Henriksen, M. Mukund, Narayan Kumar, and P.S. Thiagarajan. Regular collections of Message Sequence Charts. In *Proc. MFCS '00*, LNCS. Springer-Verlag, 2000.
10. ITU-TS Recommendation Z.120. Message sequence chart (MSC). *ITU-TS*, 1997.
11. V. Levin and D. Peled. Verification of message sequence charts via template matching. In *Proc. TAPSOFT '97*, volume 1214 of *LNCS*, pages 652–666. Springer-Verlag, 1997.
12. P. Madhusudan. Reasoning about Sequential and Branching properties of Message Sequence Graphs Technical Report IMSC/2001/04/22, *Institute of Mathematical Sciences*, Chennai, India, 2001.
13. A. Muscholl and D. Peled. Message sequence charts and decision problems on Mazurkiewicz traces. In *Proc. MFCS '99*, volume 1672 of *LNCS*, pages 81–91. Springer-Verlag, 1999.
14. A. Muscholl, D. Peled, and Z. Su. Deciding properties of message sequence charts. In *Proc. FOSSACS '98*, volume 1378 of *LNCS*, pages 226–242. Springer-Verlag, 1998.
15. B. Meenakshi and R. Ramanujam. Reasoning about message passing in finite state environments. In *Proc. ICALP '00*, LNCS. Springer-Verlag, 2000.
16. E. Rudolph, P. Graubmann, and J. Grabowski. Tutorial on message sequence charts. In *Computer Networks and ISDN Systems—SDL and MSC, Volume 28*, 1996.
17. W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 165–191, 1990.
18. W. Thomas. Languages, automata, and logic. *Handbook of Formal Language Theory*, III:389–455, 1997.
19. 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*, LNCS. Springer-Verlag, 1997.
20. I. Walukiewicz. Monadic second order logic on tree-like structures. In *Proc. STACS '96*, volume 1046 of *LNCS*. Springer-Verlag, 1996.