Global and Local Testing from Message Sequence Charts

Delphine Longuet
Univ Paris-Sud, LRI UMR8623, Orsay, F-91405
delphine.longuet@lri.fr

ABSTRACT

Message Sequence Charts are a widely used formalism for describing scenarios a communicating system must be able to perform. We study in this paper different formal frameworks for testing from MSCs. We first consider a setting where all the processes of the system can be controlled and observed globally. Then we study a setting where the system is tested from the point of view of each process individually, observations remaining local or being gathered at the end of each test. In each setting, we define a conformance relation based on global or local observations, for which we build an exhaustive test set. Moreover, we gather the conditions making local testing as powerful as global testing.

1. INTRODUCTION

During the design of a communicating system, a common practice is to visually describe the architecture of the system in terms of components, together with the desired interactions between them. Description languages like SDL [13] or UML [18] commonly use the Message Sequence Charts standard [14] to formalize these interactions. A Message Sequence Chart (MSC) shows the different processes of the system as vertical lines interpreted as time axes and communications between these processes as horizontal or downward-sloping arrows between these lines. Collections of MSCs are used to specify the scenarios the system must be able to perform. A standard way to generate sets of MSCs is to use Hierarchical MSCs (HMSCs) or equivalently Messages Sequence Graphs (MSGs) [14]. An MSG is a finite directed graph whose nodes are labelled by MSCs. Each path of an MSG from an initial to a final node defines an MSC by concatenating the MSCs labelling the nodes along the path.

The aim of this paper is to propose formal frameworks for testing from MSG specifications. The easiest way to test a distributed system is to have a global control and observation of it, meaning that one can trigger the sending of a message from a process or choose to treat a pending message for another, whenever the sending or the receipt is possible for the given process. But the nature of the system under test does not always allow a global control. In this case, one would like to test the system from the point of view of each component individually, and that it would be sufficient to decide conformance from a global point of view. We study here different ways of globally and locally testing a distributed system by defining global and local conformance relations, for which we build exhaustive test sets. Moreover, we set the conditions under which local testing is equivalent to global testing. The latter results are based on [4], where the decidability of these conditions is studied, and which inspired this work.

A formal framework for global testing from a set of MSCs has been proposed in [7]. Two conformance relations are defined, depending on the chosen semantics of the MSC specification, and a test generation algorithm is presented, but no exhaustivity result is provided. The closest work to ours are [8, 6]. Even if they are not all based on MSC specifications, they address the problem of defining local conformance relations and studying ways to build tests for these relations. For instance, the notions of independent and cooperative refinement in [6] are similar to our local conformance relations.

We first recall definitions of MSCs and MSGs in Sec. 2, and the general formal testing framework in Sec. 3. We propose in Sec. 4 a first conformance relation in the global setting and give an exhaustive test set for it. Then we study in Sec. 5 two conformance relations in the local setting, depending on the control and observation being purely local or distributed on a subset of processes. We give conditions for the equivalence to the global conformance relation and build exhaustive sets of local or distributed tests. Proofs may be found in the long version of this paper available online.[1]

2. PRELIMINARIES

2.1 Message Sequence Charts

We fix a finite set Proc of process names and a finite set Msg of message types. For any $p, q \in \text{Proc}$ with $p \neq q$ and any $m \in \text{Msg}$, we denote by $p!q(m)$ the sending by p of the message m to q, and by $p?q(m)$ the receipt by p of the message m from q. For $p \in \text{Proc}$, we define $\Sigma_p = \{p!q(m), p?q(m) \mid q \in \text{Proc}, m \in \text{Msg}\}$ the set of communications involving p. We define $\Sigma = \bigcup_{p \in \text{Proc}} \Sigma_p$. We denote by $\Sigma^!$ the set $\{p!q(m) \mid p, q \in \text{Proc}, m \in \text{Msg}\}$ of sendings and by $\Sigma^?$ the set $\{p?q(m) \mid p, q \in \text{Proc}, m \in \text{Msg}\}$ of receipts.

A *poset* over an alphabet Σ is a triple $(E, \leq, \lambda)$ where $(E, \leq)$ is a partially ordered set, and $\lambda : E \to \Sigma$ is a labelling mapping. A poset can be seen as an abstraction of the behaviour of a concurrent system: an element $e \in E$ is called an *event* and its label $\lambda(e)$ is the corresponding communication action performed by the system. The partial order $\leq$ describes the causal dependence between the events, independence of events meaning that they can be executed

[1] `www.lri.fr/~longuet/publications.html`

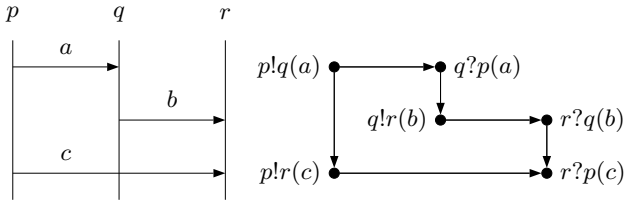


Figure 1: An MSC and its partial order representation (Hasse diagram).

concurrently: two elements e and e' are concurrent if $e \not\leq e'$ and $e' \not\leq e$.

For $p \in$ Proc, we define $E_p = \{e \in E \mid \lambda(e) \in \Sigma_p\}$ the set of events occurring on p and for $e \in E$, we define $E_a = \{e \in E \mid \lambda(e) = a\}$ the set of events labelled by the communication a. We define $E^! = \{e \in E \mid \lambda(e) \in \Sigma^!\}$ the set of sending events and $E^? = \{e \in E \mid \lambda(e) \in \Sigma^?\}$ the set of receipt events.

An *MSC*[2] over Proc is a poset $(E, \leq, \lambda)$ over Σ such that: for each $p \in$ Proc, the relation $\leq_p = \leq \cap (E_p \times E_p)$ is a total order; there is a bijective mapping $c : E^! \to E^?$ matching corresponding sendings and receipts; and the partial order $\leq$ is the reflexive and transitive closure of $\bigcup_{p \in \text{Proc}} \leq_p \cup \{(e, c(e)) \mid e \in E^!\}$.

Following the recommendation Z.120 [14], we allow the overtaking of different messages sent by a process p to a process q, but forbid the reordering of identical messages from p to q. This condition is called *non-degeneracy*. Formally, there must not be two send events $e, e' \in E^!$ such that $\lambda(e) = \lambda(e')$, $e < e'$ and $c(e') < c(e)$. Since these messages are indistinguishable, it does not make sense to explicitly specify their reception to be reversed. However, if the underlying architecture is FIFO, the order in which any two messages sent by a process p are received by a process q must not be reversed. Formally, there must not be two send events $e = p!q(a)$ and $e' = p!q(b)$ such that $e < e'$ and $c(e') < c(e)$. In the following, we will make no assumption on the underlying architecture and then only consider the non-degeneracy condition, but we will mention when the FIFO condition is needed.

A *linearization* of an MSC $M = (E, \leq, \lambda)$ is an execution of the communication actions of Σ compatible with the partial order $\leq$. Formally, a linearization of M is a word $a_1 \dots a_{|E|}$ over Σ such that there exists a total order $e_1 \dots e_{|E|}$ of the events of E with $\lambda(e_i) = a_i$ for all i, $1 \leq i \leq |E|$ and $e_i \leq e_j$ implies $i \leq j$ for all i, j, $1 \leq i, j \leq |E|$. For instance, $p!q(a)\ q?p(a)\ p!r(c)\ q!r(b)\ r?q(b)\ r?p(c)$ is a linearization of the MSC of Fig. 1. We denote by $\text{Lin}(M)$ the set of all the linearizations of M (also called the language of M).

For $w \in \Sigma^*$, we denote by $\text{Pref}(w)$ the set of prefixes of w: $\text{Pref}(w) = \{w_1 \in \Sigma^* \mid \exists w_2 \in \Sigma^*, w_1.w_2 = w\}$. For $L \subseteq \Sigma^*$, we denote by $\text{Pref}(L)$ the prefix closure of L.

2.2 Message Sequence Graphs

A natural way to specify a communicating distributed system is by a (possibly infinite) collection of scenarios the system should be able to perform. These scenarios can be generated by combining basic scenarios in a high-level description called a high-level MSC (HMSC) [16]. Without losing expressiveness, we consider only a subclass of HMSC called Message Sequence Graphs (MSG). A message sequence graph is a finite directed graph with initial and terminal nodes, where each node is labelled by an MSC. MSCs along a path in the graph are composed using asynchronous concatenation, defined as follows.

Let $M = (E, \leq, \lambda)$ and $M' = (E', \leq', \lambda')$ be two MSCs such that E and E' are disjoint. The asynchronous concatenation of M and M' is the MSC $M \circ M' = (E'', \leq'', \lambda'')$ where $E'' = E \cup E'$, $\lambda''(e) = \lambda(e)$ if $e \in E$ and $\lambda''(e) = \lambda'(e)$ if $e \in E'$, and $\leq'' = (\leq \cup \leq' \cup \bigcup_{p \in \text{Proc}} E_p \times E'_p)^*$.

An *MSG* is a structure $\mathcal{G} = (Q, \mathcal{M}, \to, I, F, \Phi)$ where Q is a finite set of nodes, $I \subseteq Q$ is the set of initial nodes, $F \subseteq Q$ is the set of final nodes, $\to \subseteq Q \times Q$ is the transition relation, $\mathcal{M}$ is a set of MSCs with disjoint sets of events and $\Phi : Q \to \mathcal{M}$ labels each node with an MSC.

We give in Fig. 2(a) an MSG for a simple connection protocol. The protocol p allows the higher layer h to connect to the lower layer l. The lower layer can refuse the connection, which is reinitialized, or accepts the connection and allows h to send data until h decides to end the connection.

A path π through an MSG is a sequence $q_0 \to q_1 \to \dots \to q_n$ where $(q_i, q_{i+1}) \in \to$ for all i, $1 \leq i < n$. The MSC generated by π is the asynchronous concatenation of the MSCs labelling nodes along the path: $M(\pi) = M_0 \circ M_1 \circ \dots \circ M_n$ where $M_i = \Phi(q_i)$ for all i, $1 \leq i \leq n$. The path π is a run if $q_0 \in I$ and $q_n \in F$. We denote by $M(\mathcal{G})$ the set of MSCs generated by runs through $\mathcal{G}$: $M(\mathcal{G}) = \{M(\pi) \mid \pi \text{ is a run through } \mathcal{G}\}$. The language defined by $\mathcal{G}$ is the union of the languages of all the MSCs generated by runs through $\mathcal{G}$: $\text{Lin}(\mathcal{G}) = \{\text{Lin}(M) \mid M \in M(\mathcal{G})\}$.

MSGs describing regular languages (i.e. such that $\text{Lin}(\mathcal{G})$ is regular over Σ) are known to have good properties. For instance, any MSG $\mathcal{G}$ describing a regular language is bounded: along any linearization $\ell \in \text{Lin}(\mathcal{G})$, there exists a constant B such that no channel ever contains more than B messages. An example of a non-regular MSG is given Fig. 2(b): every word in the language described by this MSG must contain as many $p!q(a)$ as $q?p(a)$, so it is clearly not regular. It is undecidable whether a given MSG describes a regular language [11]. Yet, the regularity of the language of an MSG can be ensured if one restricts to locally-synchronised MSGs.

The *communication graph* of an MSC $M = (E, \leq, \lambda)$ is the directed graph $(\text{Proc}_M, \mapsto)$ where Proc_M is the set $\{p \in \text{Proc} \mid E_p \neq \emptyset\}$ of active processes of M, and $p \mapsto q$ if and only if there exists $e \in E_p$ such that $\lambda(e) = p!q(m)$. We say that an MSC is *connected* if its communication graph consists of one non-trivial strongly connected component and isolated vertices. The communication graph shows the directions of communications between processes. The condition for an MSG to be locally-synchronised ensures that the communications between two processes are not indefinitely one-way: an MSG $\mathcal{G}$ is said to be *locally-synchronised* if for every loop $\pi = q \to q_1 \to \dots \to q$, the MSC $M(\pi)$ is connected. The MSG of Fig. 2(a) is locally-synchronised. It is shown in [3] that locally-synchronised MSGs define regular languages. In fact, regular languages exactly are the languages defined by locally-synchronised MSGs [12].

3. FORMAL TESTING

The aim of this paper is to propose different formal frameworks for testing from MSG specifications. A formal testing framework relies on the definition of a *conformance relation* $\text{Conf}(S, \mathcal{G})$, which formalises the relation that the system under test S and its specification $\mathcal{G}$ must verify.

According to the controllability and the observability one is given on the system under test, the notion of test is defined. The success of a test is determined by the verdict associated to the result of its execution on the system. We consider three verdicts: pass,

[2]Note that we only consider MSCs describing closed systems, where all the messages are exchanged between the processes and no message is sent to or received from the environment.

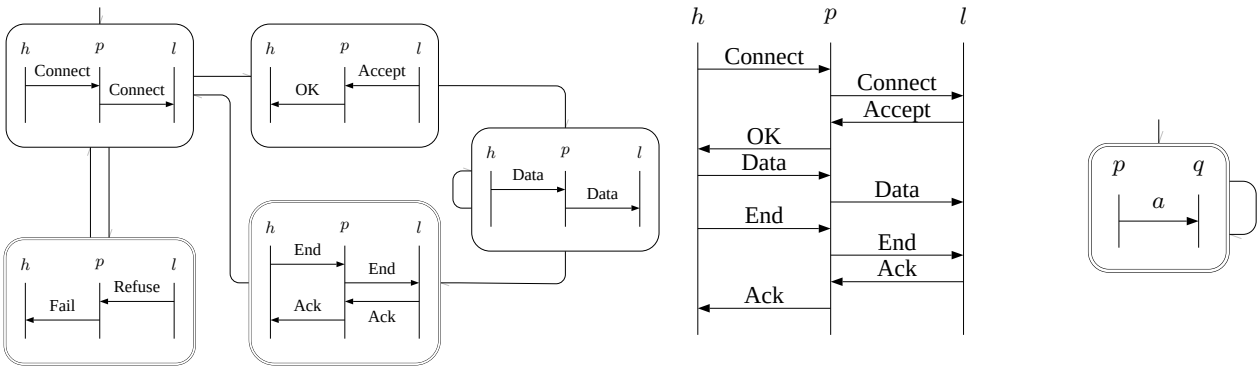


Figure 2: (a) An MSG and one of its derived MSC. (b) A non-regular MSG.

fail and inc. The pass verdict means that the result of the test is consistent with the specification according to the conformance relation. The fail verdict means that a counter-example to conformance has been found. The inc verdict (for inconclusive) is used when the test does not show non-conformance but does not either show the behaviour targeted by the test. For instance, the execution stopped without error before the end of the test or the observed output is not the expected one but still is a possible output. We say that the submission of a test t is a success (resp. a failure), denoted by S passes t (resp. S fails t), when the verdict of its execution is pass or inc (resp. fail). The submission of a test set is successful if and only if the submission of every test in this set is.

In order for a conformance relation to be testable, one must prove that a (potentially infinite) test set exists whose success is equivalent to conformance: such a test set is called *exhaustive* for this conformance relation and denoted by $\text{Exh}_{\text{Conf}}(\mathcal{G})$. Two dual properties must hold:
(1) Any correct system passes the test set:

$$\text{Conf}(S, \mathcal{G}) \Rightarrow \forall t \in \text{Exh}_{\text{Conf}}(\mathcal{G}), S \text{ passes } t$$

(2) Any incorrect system fails the test set:

$$\neg\text{Conf}(S, \mathcal{G}) \Rightarrow \exists t \in \text{Exh}_{\text{Conf}}(\mathcal{G}), S \text{ fails } t$$

The existence of an exhaustive test set ensures the testability of the conformance relation.

Test Hypotheses.

We assume here that the system under test can be modelled by a network of asynchronous concurrent automata. Concurrent automata are the usual framework used for studying realizability of MSCs [17, 1, 2], i.e. the property for a set of MSCs to describe the behaviour of an actual distributed system. The behaviour of the system is given by the product automaton of its components. An execution is a finite path in the product automaton starting from the initial state, and the word labelling an execution is called a trace.[3] Since we are in a black-box testing context, only traces of the system are observable, so given an MSG specification $\mathcal{G}$ over the alphabet of communication actions Σ, the observable behaviour of the system under test S is given by its set of traces written $\text{Tr}(S)$, which is a subset of words over Σ. Not all words are possible traces of a system. The least requirement is that any receipt corresponds to an emission that occurred before it. Following [1], we say that a word $w \in \Sigma^*$ is *well-formed*, denoted by $\text{WF}(w)$, if and only if all its prefixes satisfy $|E_{p!q(m)}| \geq |E_{q?p(m)}|$ for all $p, q \in$ Proc and $m \in$ Msg. A well-formed word is complete if every message is received. We assume in the following that traces of S are well-formed words over Σ. We also assume that the communication channels of the system under test have a bounded capacity. It ensures that the traces observed during testing will not contain an arbitrarily large number of pending messages between two processes.

Finally, since concurrent systems are highly non-deterministic, we need to set the usual *fairness hypothesis* [5]: there exists a bound n such that if a test is executed n times on the system under test, then all possible behaviours are observed. It implies that it is possible to decide without ambiguity the verdict of any test after n executions of it. Obviously, the bound n can not be guessed a priori and needs to be determined experimentally, which is a classical problem when testing non-deterministic systems.

4. GLOBAL TESTING FROM MSCS

We first consider the case where all the communications between the system processes can be controlled and observed at a global level. This setting can be used to check the consistency of MSC scenarios with an SDL specification for instance, in a similar way to [9].

The natural conformance relation one can think of is language inclusion: the executions of the system under test must all be valid scenarios of the specification. In the case of an MSG specification $\mathcal{G}$, it means that the executions of the system must not contradict the partial order described by $\mathcal{G}$. In other words, all the traces of the system must be (prefixes of) linearizations of $\mathcal{G}$.

Definition 1. (Global conformance relation) A system S globally conforms to its MSG specification $\mathcal{G}$, denoted by $\mathsf{GC}(S, \mathcal{G})$, if and only if $\text{Tr}(S) \subseteq \text{Pref}(\text{Lin}(\mathcal{G}))$.

To check this conformance relation, testing that the system under test can perform the scenarios of its specification is useless, since it will not allow to conclude on the language inclusion. We would rather want to ensure that no execution of the system violates the partial order of the specification. To this aim, a test will be designed to force the system to exhibit an execution violating this partial order. For instance, in the MSC of Fig. 1, the sending of b by q is forbidden before the reception of a by q. Then the test following the sequence of communications actions $p!q(a)\ q!r(b)$ must fail on

[3]We do not consider final states here, so any finite path from the initial state is an execution.

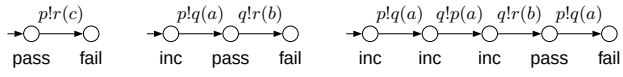
a correct implementation. Therefore, we would like to submit to the system tests built from words in Σ^+ violating the partial order of $\mathcal{G}$. Actually, it is sufficient to consider the words diverging from (prefixes of) linearizations of $\mathcal{G}$:

$$\text{Div}_{\mathcal{G}} = \{w.a \in \Sigma^+ \mid w \in \text{Pref}(\text{Lin}(\mathcal{G})) \wedge a \in \Sigma \\ \wedge\, w.a \notin \text{Pref}(\text{Lin}(\mathcal{G})) \wedge \text{WF}(w.a)\}$$

A basic test built over a word in this set is a linear automaton over Σ whose every state is labelled by a verdict pass, fail or inc. We want the execution of the test to yield the following verdicts: if the word $w.a$ is observed as an execution of S, then fail; if S can perform w but not a (it stops after w), then pass; if S cannot perform w entirely but only a prefix of it, then inc: no violation of the partial order is observed but the test did not reach the end. Then we have the following definition of a basic test for a word.

Definition 2. Let $w \in \Sigma^+$, $w = a_1 \dots a_n a$ with $n \geq 0$. A *basic test for* w, denoted by $T(w)$, is an automaton (Q, δ, q_0) where: $Q = \{q_i\}_{0 \leq i \leq n+1}$; $\delta = \{(q_i, q_{i+1})\}_{0 \leq i \leq n}$; states q_0 to q_{n-1} are labelled by inc, state q_n is labelled by pass and state q_{n+1} is labelled by fail.

Therefore, basic tests for the global conformance relation are basic tests built over words in $\text{Div}_{\mathcal{G}}$. We denote by $\text{Exh}_{\mathsf{GC}}(\mathcal{G})$ the test set $\{T(w) \mid w \in \text{Div}_{\mathcal{G}}\}$. Here are some basic tests for the MSC of Fig. 1. (a) Process p must not be able to send c to r in the initial state of the system. (b) Process q must not send b to r before the reception of a from p. (c) Process p must not send a to q again after process q sent b to r.

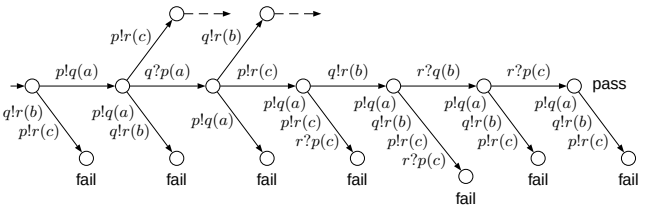


Since the system under test is closed (it does not send or receive messages to or from the environment), the test cannot interact with it by sending and receiving messages, but it can still control and observe the behaviour of the system by synchronising with it. The execution of a test on the system is then defined as the synchronous product of the two automata: the test and the system synchronise on identical communication actions, allowing the execution of the test to follow the execution of the system and to yield a verdict whenever a leaf is reached or a deadlock occurs. We have the following result.

THEOREM 1. *The test set $\text{Exh}_{\mathsf{GC}}(\mathcal{G})$ is exhaustive for the global conformance relation GC.*

A particular feature of MSG specifications is to describe not only acceptable behaviours but complete scenarios. The implementation is then expected both to be consistent with the partial order specified and to show the complete scenarios of the specification. In order to refine the outcome of a test, we can slightly modify the verdicts associated to its states: in a test built from a word $w.a$, the last but one state q_n is labelled by pass only if w is a complete linearization, it is labelled by inc otherwise.

The next step is to factorise tests to avoid tests without pass states. Tests can be factorised by common (non-empty) prefixes, leading to the following automaton for the MSC of Fig. 1. For the sake of readability, we gather on the same transition the labels of transitions leading to a state labelled by the same verdict. We only show the factorisation of tests around the linearization $p!q(a)\ q?p(a)\ p!r(c)\ q!r(b)\ r?q(b)\ r?p(c)$, the dashed arrows leading to the rest of the automaton. All the internal states are labelled by inc.



An automaton gathering the basic tests built from all the words with common (non-empty) prefixes then has a tree-like structure and all its leaves labelled by pass or fail. Moreover, only states reachable by a complete linearization of the specification are labelled by pass.

5. LOCAL TESTING FROM MSCS

We consider now that the communications inside the system under test cannot be observed at a global level. The system can only be controlled and observed locally, from the point of view of each process individually. We will first study a pure local testing framework where the system is tested through only one process at a time. Then we will study the case where tests have to be executed by several processes and observations have to be gathered at the end of the test.

5.1 Local Testing with Local Observations

A natural way to test a distributed system is to replace one by one each process of the system and to record the interactions between this process and the rest of the system. This is what we will call pure local testing. The system is tested locally from the point of view of each process and the verdict of the tests is also local, since the observations obtained from the different processes are never gathered.

To define it generally, a local observation of an MSC is the set of totally ordered sequences of events occurring on a given set of processes P. In this subsection, we will only consider local observations where P is reduced to one process.

Definition 3. Let M be an MSC over Proc and Σ, p be a process in Proc and w be a word in Σ^*.
The *projection* of w on p, denoted by $w_{|_p}$, is defined inductively by $\varepsilon_{|_p} = \varepsilon$ and $(a.w')_{|_p} = a.(w'_{|_p})$ if $a \in \Sigma_p$ and $w'_{|_p}$ otherwise.
For $P \subseteq$ Proc a set of processes, the *P-observation* of M, denoted by $M_{|_P}$, is defined as the collection $\{w_{|_p}\}_{p \in P}$ for any complete linearization w of M.[4]

Observing the executions of the system on each process individually, we can only verify that the communications on each process occur according to the total order specified. We want the projections of the traces of the system on each process to correspond to projections of (prefixes of) linearizations of the MSG specification.

Definition 4. The set of *1-observations* of an MSG $\mathcal{G}$, denoted by $\text{locLin}_1(\mathcal{G})$, is the set $\{w_{|_p} \mid w \in \text{Pref}(\ell), p \in \text{Proc}\}$ for any complete linearization ℓ of $\text{Lin}(\mathcal{G})$.

We write $\text{locTr}_1(S)$ for the set $\{t_{|_p} \mid t \in \text{Tr}(S), p \in \text{Proc}\}$ of projections of the traces of S on each process of Proc. We then define the following conformance relation.

Definition 5. (Local conformance relation) A system S locally conforms to its MSG specification $\mathcal{G}$, denoted by $\mathsf{LC}_1(S, \mathcal{G})$, if and only if $\text{locTr}_1(S) \subseteq \text{locLin}_1(\mathcal{G})$.

[4]Note that the projection of an MSC on a set of processes is unique since events are totally ordered on each process.

Ideally, we would like this conformance relation to be equivalent to the global one: we would like local testing to be sufficient for detecting implementations that do not respect the global partial order of the specification. For instance, we would like to be able to discard an implementation exhibiting the scenario M of Fig. 3, when only scenarios M_1 and M_2 belong to the specification. Unfortunately, the third scenario cannot be distinguished from the others with local observations only. Such a scenario is said to be implied.

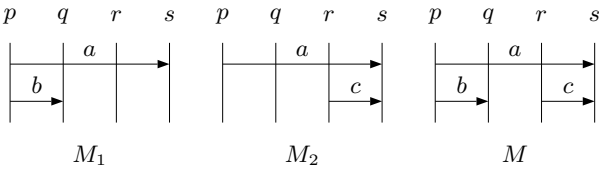


Figure 3: M is 1-implied by $\{M_1, M_2\}$ but not 2-implied.

Definition 6. Let $\mathbb{P} \subseteq 2^{\text{Proc}}$ be a family of subsets of processes. An MSC M is said to be $\mathbb{P}$-*implied* by an MSG specification $\mathcal{G}$ if for every subset $P \in \mathbb{P}$, there is an MSC M' in $M(\mathcal{G})$ such that $M'_{|_P} = M_{|_P}$.
We denote by P_k the set $\{P \subseteq \text{Proc} \mid |P| = k\}$ of all subsets of Proc of size k and we say that an MSC is k-*implied* if it is P_k-implied.

The scenario M of Fig. 3 is 1-implied by the two first scenarios. Since tests designed from local observations only would not reject an implementation allowing such a scenario, it has to be already present in the specification. Thus, local testing allows to decide global conformance if and only if all the scenarios 1-implied by the MSG specification are included in the specification. This condition is called 1-testability [4].

Definition 7. An MSG specification $\mathcal{G}$ is said to be *1-testable* if all the MSCs 1-implied by $\mathcal{G}$ are already in $M(\mathcal{G})$.

THEOREM 2. *Let $\mathcal{G}$ be a locally-synchronised MSG specification and S a system under test. If $\mathcal{G}$ is 1-testable, then S locally conforms to $\mathcal{G}$ iff S globally conforms to $\mathcal{G}$.*

We know from [17] that this property is decidable for locally-synchronised MSG specifications in the non-FIFO setting.[5] In the FIFO setting, it is undecidable in general [4]. However, as underlined in [4], since we assume that the system under test has bounded FIFO buffers, it becomes decidable whether its behaviours, observed only locally, are included in the behaviours allowed by the MSG specification. So in a practical setting, 1-testability is decidable also for FIFO architectures.

Therefore, for locally-synchronised MSG specifications that are 1-testable, pure local testing is equivalent to global testing. It implies that tests for the global conformance relation can be built from projections of (prefixes of) linearizations.

We build basic local tests for a process p from the projections of linearizations of MSCs $M \in M(\mathcal{G})$ on p like basic global tests are built from linearizations of $\mathcal{G}$. The only difference is that the divergence from a prefix of a projection need not be a well-formed word:

$$\text{Div}_{\mathcal{G},p} = \{w.a \in \Sigma_p^+ \mid \exists \ell \in \text{Pref}(\text{Lin}(\mathcal{G})), w = \ell_{|_p} \wedge a \in \Sigma_p \wedge \forall \ell \in \text{Pref}(\text{Lin}(\mathcal{G})), w.a \neq \ell_p\}$$

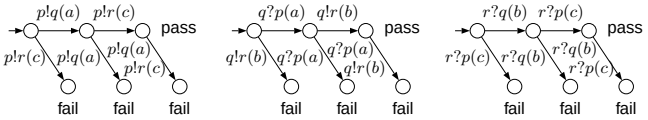
[5] In [17], this result is shown for a larger class of MSG specifications, where communication graphs of loops are only weakly connected (called globally-cooperative MSG in [10]).

Basic local tests for a process p then are basic tests built over words in $\text{Div}_{\mathcal{G},p}$. We denote by $\text{Exh}_{\mathsf{LC}_1}(\mathcal{G})$ the test set $\{T(w) \mid w \in \text{Div}_{\mathcal{G},p}, p \in \text{Proc}\}$.

In the context of pure local testing, executing a basic test boils down to replacing the corresponding process by a testing process whose behaviour is described by the test automaton. The execution of the test is then defined as the asynchronous product between this automaton and the rest of the system. The verdict is yield whenever a leaf is reached or a deadlock occurs. We have the following result.

THEOREM 3. *The test set $Exh_{\mathsf{LC}_1}(\mathcal{G})$ is exhaustive for the local conformance relation LC_1.*

Like in the global testing setting, basic local tests for a given process can be refined so that pass states are reachable only by the projection of a complete linearization. Then they can be factorised by common non-empty prefixes. We obtain the following local tests for the example of Fig. 1 (internal states are all labelled by inc).



5.2 Local Testing with Gathered Observations

If a specification is not 1-testable, it means that testing for global conformance to this specification cannot be purely local. If we consider again the scenarios of Fig. 3, we can see that the scenario M is 1-implied by $\{M_1, M_2\}$ but not 2-implied: if the observations of M from the two processes p and s for instance, are considered together, it appears that the scenario M is not consistent with any of the two scenarios M_1 and M_2. Such a scenario can then be discarded by a test executed by the two processes p and s whose observations are gathered at the end.

In this context, testing then consists in replacing, in turn, k processes of the system by k testing processes, locally recording their interactions with the rest of the system, and gathering the obtained observations at the end of each test. The corresponding conformance relation would be the inclusion of the P-observations obtained from the system under test in the P-observations of the specification, for any set of processes P of size k. We recall that P_k is the set of subsets of Proc of size k.

Definition 8. The set of k-*observations* of an MSG $\mathcal{G}$, denoted by $\text{locLin}_k(\mathcal{G})$, is the set $\{\{w_{|_p}\}_{p \in P} \mid w \in \text{Pref}(\ell), P \in P_k\}$ for any complete linearization ℓ of $\text{Lin}(\mathcal{G})$.

We denote by $\text{locTr}_k(S)$ the collection of the projections of the traces of S on each subset $P \in P_k$: $\text{locTr}_k(S) = \{\{t_{|_p}\}_{p \in P} \mid t \in \text{Tr}(S), P \in P_k\}$. We then define the following conformance relation.

Definition 9. (k-local conformance relation) A system S k-locally conforms to its MSG specification $\mathcal{G}$, denoted by $\mathsf{LC}_k(S, \mathcal{G})$, if and only if $\text{locTr}_k(S) \subseteq \text{locLin}_k(\mathcal{G})$.

Following the same reasoning as in the previous subsection, this conformance relation is equivalent to global conformance if all the scenarios k-implied by the MSG specification are already present in the specification. We have the following notion of k-testability [4]:

Definition 10. Let $k \in \mathbb{N}$, $1 \leq k \leq |\text{Proc}|$. An MSG specification $\mathcal{G}$ is said to be k-*testable* if all the MSCs k-implied by $\mathcal{G}$ are already in $M(\mathcal{G})$.

THEOREM 4. *Let $\mathcal{G}$ be a locally-synchronised MSG specification and S a system under test. If $\mathcal{G}$ is k-testable, then S k-locally conforms to $\mathcal{G}$ iff S globally conforms to $\mathcal{G}$.*

We do not know about the decidability of k-testability in the non-FIFO setting. However, in the FIFO setting, as for the 1-testability condition, k-testability is undecidable in the general case [4]. Assuming bounded buffers fortunately makes it decidable, so in practice, it can be verified that an MSG specification is closed by k-implied scenarios.

We denote by Σ_P the set $\bigcup_{p\in P}\Sigma_p$. We define the projection of a word $w \in \Sigma^*$ over a set of processes $P \subseteq$ Proc, written $w_{|_P}$, inductively by $\varepsilon_{|_P} = \varepsilon$ and $(a.w')_{|_P} = a.(w')_{|_P}$ if $a \in \Sigma_P$ and $w'_{|_P}$ otherwise.

As a generalisation of the construction of tests in the previous subsection, a test for this local conformance relation will be built from projections of (prefixes of) linearizations of $\mathcal{G}$ on each set of processes of size k. Therefore, a local test for a set of processes P will be built from words in $\text{Div}_{\mathcal{G},P}$:

$$\text{Div}_{\mathcal{G},P} = \{w.a \in \Sigma_P^+ \mid \exists \ell \in \text{Pref}(\text{Lin}(\mathcal{G})), \ell_{|_P} = w \wedge a \in \Sigma_P \\ \wedge \forall \ell \in \text{Pref}(\text{Lin}(\mathcal{G})), \ell_{|_P} \neq w.a\}$$

To base the decision of the verdict of a test on the combined observations made on the k processes of a set P, the projections of a given word in $\text{Div}_{\mathcal{G},P}$ on each process p of P must be kept together. A local test for a word in $\text{Div}_{\mathcal{G},P}$ will be distributed on processes of P in the following way.

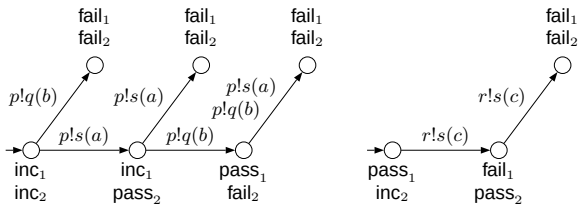
Definition 11. Let $P \subseteq$ Proc. Let $w \in \Sigma_P$, $w = w'.a$ with $a \in \Sigma_p$ for $p \in P$. A *basic test for w distributed on the set of processes P*, denoted by $T_P(w)$, is a tuple of automata $(T_q)_{q\in P}$, where for all $q \in P$, $q \neq p$, T_q is a linear automaton over $w_{|_q}$ where internal states are labelled by inc and the leaf is labelled by pass, and $T_p = T(w_{|_p})$.

A distributed test then consists of a tuple of automata that have to be run together, with the rest of the system under test, to yield a verdict. Formally, the execution of a distributed test on the system under test is the asynchronous product of the k automata of $T_P(w)$ and the rest of the system. The test is a success if all the automata of $T_P(w)$ reach a pass or inc state, and is a failure if the fail state is reached. We denote by $\text{Exh}_{\mathsf{LC}_k}(\mathcal{G})$ the test set $\{T_P(w) \mid w \in \text{Div}_{\mathcal{G},P}, P \in P_k\}$ and prove it to be exhaustive.

THEOREM 5. *The test set $Exh_{LC_k}(\mathcal{G})$ is exhaustive for the k-local conformance relation LC_k.*

As we already did in the previous frameworks, we would like to refine the verdicts of tests in order to distinguish complete linearizations from other authorised behaviours. We then change a little the verdicts associated to states of distributed tests: in a basic distributed test $T_P(w.a)$ where $a \in \Sigma_p$, the final states of all automata T_q, $q \neq p$ and the last but one state of T_p will be labelled by pass only if w is a complete linearization; they will be labelled by inc otherwise. As before, we want to factorise the tests built from words with common prefixes, in order to avoid inc verdicts as much as possible. However, if we factorise naively the tests obtained for each process separately, we lose the global observation that we need to conclude on the result of a test. The information of the linearization (or equivalently, the MSC) from which a distributed test is issued must be kept on the automata of this test. We will then label each MSC of $\mathcal{G}$ by a natural, and transfer this labelling on the states of each projection on a set of processes.[6] Thus, the success of the execution of a test is determined by reaching pass or inc states in all the automata of the test, but these states have to be labelled by the same number, meaning that they were reached by projections of a linearization of the same MSC.

Coming back to the example of Fig. 3, we want to build tests for 2-local conformance to the specification $\mathcal{G}$ composed only of the MSCs M_1 and M_2. These tests will be able to discard the scenario M, since M is not 2-implied. Tests are built from words in $\text{Div}_{\mathcal{G},P}$ for any subset of two processes $P \subseteq \{p, q, r, s\}$. These words are divergences of projections on P of a linearization of M_1 or a linearization of M_2. To keep the information of the MSC M_i from which each basic test is built, we index by i, $i = 1, 2$, each verdict of the basic test built from a linearization of M_i. For instance, tests built from words in $\text{Div}_{\mathcal{G},\{p,r\}}$ with states labelled by verdict pass_i, inc$_i$ and fail_i, $i = 1, 2$, can be safely factorised, giving the following automata.



Executing these tests on a system allowing the scenario M yields the verdicts pass_1 and fail_2 for p, and fail_1 and pass_2 for r, showing the incompatibility of the observations.

We can generalize this way of distributing tests to the case where $k = n$, with n the number of processes of the specification. Notice that n-local conformance is equivalent to global conformance without condition since an MSG specification is always n-testable. Thus we obtain local tests distributed on n processes for global conformance as an alternative to global tests.

6. CONCLUSION AND FUTURE WORK

We showed in this paper how to locally test a system for global conformance to its MSG specification. We saw that a closure condition on the specification was needed to ensure the equivalence of the global and local conformance relations. It does not mean that local testing is impossible without this condition, but only that the local conformance relation is weaker than the global one, which is not necessarily a problem. For instance, one can consider the closure by implied scenarios as the semantics of the MSG specification and then accept them as valid scenarios.

A natural extension of the proposed frameworks is to consider an MSC as a description of the interactions of a (distributed) system with a distributed environment, the system being represented as a strict subset of processes. Testing such a system consists in setting a testing process at each port (instead of each process representing the environment) which controls and observes the system through this interface. Different conformance relations can be defined depending on how strictly the partial order specified for the system must be implemented.

Afterwards, selection criteria must also be investigated in order to choose a representative subset of the exhaustive test set and then generate a test set of reasonable size to submit to the system. Selection by test purposes like in [15] must also be dealt with.

7. REFERENCES

[1] R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence charts. *IEEE Transactions on Software Engineering*, 29(7):623–633, 2003.

[6] $M(\mathcal{G})$ is countable if Msg is countable and $\mathcal{G}$ is locally-synchronised.

[2] R. Alur, K. Etessami, and M. Yannakakis. Realizability and verification of MSC graphs. *Theoretical Computer Science*, 331(1):97–114, 2005.

[3] R. Alur and M. Yannakakis. Model checking of message sequence charts. In *CONCUR*, volume 1664 of *LNCS*, pages 114–129, 1999.

[4] P. Bhateja, P. Gastin, M. Mukund, and K. N. Kumar. Local testing of message sequence charts is difficult. In *FCT*, volume 4639 of *LNCS*, pages 76–87, 2007.

[5] A. Cavalcanti and M.-C. Gaudel. Testing for refinement in CSP. In *ICFEM*, volume 4789 of *LNCS*, pages 151–170, 2007.

[6] A. Cavalcanti, M.-C. Gaudel, and R. M. Hierons. Conformance relations for distributed testing based on CSP. In *ICTSS*, LNCS, 2011. To appear.

[7] I. S. Chung, H. S. Kim, H. S. Bae, Y. R. Kwon, and B.-S. Lee. Testing of concurrent programs based on message sequence charts. In *PDSE*, pages 72–82, 1999.

[8] H. Dan and R. M. Hierons. Conformance testing from message sequence charts. In *ICST*, pages 279–288. IEEE Computer Society, 2011.

[9] D. D'Souza and M. Mukund. Checking consistency of SDL+MSC specifications. In *SPIN*, volume 2648 of *LNCS*, pages 151–165, 2003.

[10] B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state high-level MSCs: Model-checking and realizability. *Journal of Computer and System Sciences*, 72(4):617–647, 2006.

[11] J. G. Henriksen, M. Mukund, K. N. Kumar, M. A. Sohoni, and P. S. Thiagarajan. A theory of regular MSC languages. *Information and Computation*, 202(1):1–38, 2005.

[12] J. G. Henriksen, M. Mukund, K. N. Kumar, and P. S. Thiagarajan. On message sequence graphs and finitely generated regular MSC languages. In *ICALP*, volume 1853 of *LNCS*, pages 675–686, 2000.

[13] ITU-TS. Recommendation Z.100: Specification and description language, 2002.

[14] ITU-TS. Recommendation Z.120: Message sequence charts, 2004.

[15] C. Jard. Synthesis of distributed testers from true-concurrency models of reactive systems. *Information & Software Technology*, 45(12):805–814, 2003.

[16] S. Mauw and M. A. Reniers. High-level message sequence charts. In *SDL Forum*, pages 291–306. Elsevier, 1997.

[17] R. Morin. Recognizable sets of message sequence charts. In *STACS*, volume 2285 of *LNCS*, pages 523–534, 2002.

[18] OMG. Unified Modeling Language version 2.3, 2010.