Realizability criteria for compositional MSC

Arjan Mooij, Judi Romijn, and Wieger Wesselink

Technische Universiteit Eindhoven
Department of Mathematics and Computer Science
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{a.j.mooij,j.m.t.romijn,j.w.wesselink}@tue.nl

Abstract. Synthesizing a proper implementation for a scenario-based specification is often impossible, due to the distributed nature of implementations. To be able to detect problematic specifications, realizability criteria have been identified, such as non-local choice.

In this work we develop a formal framework to study realizability of compositional MSC [GMP03]. We use it to derive a complete classification of criteria that is closely related to the criteria for MSC from [MGR05]. Comparing specifications and implementations is usually complicated, because different formalisms are used. We treat both of them in terms of a single formalism. Therefore we extend the partial order semantics of [Pra86,KL98] with a way to model deadlocks and with a more sophisticated way to address communication.

1 Introduction

For scenario-based specifications of distributed systems (e.g. in terms of Message Sequence Chart, MSC), it is often impossible to synthesize an implementation with exactly the same behavior. This is caused by the distributed nature of implementations. The best-known phenomenon leading to problems is non-local choice [BAL97], but also other criteria [HJ00,Gen05,MGR05] have been proposed to determine realizability of specifications in practice [MG05]. In this work we develop a formal framework to study such criteria for the MSC extension that is called compositional MSC [GMP03,MM01]. Our work differs from [AEY05], which studies decidability and worst-case complexity of checking whether an MSC specification is realizable, but provides no practical criteria.

Most realizability criteria seem to be tricky formalizations of intuitions about realizability. In contrast, we formally study under what circumstances specifications are trace equivalent to their implementations, and derive a condition that is both necessary and sufficient. From this condition, we derive a complete classification of realizability criteria for compositional MSC. The resulting formal criteria can easily be related to our intuitive criteria in [MGR05].

Several kinds of semantics have been proposed for MSC specifications (e.g. [KL98, Ren99, Hey00, UKM03]), while implementations are typically expressed in

* This research is supported by the NWO under project 016.023.015: “Improving the Quality of Protocol Standards”.

terms of finite state machines. To compare specifications and implementations, two different formalisms must then be related, usually via execution traces (in fact a third formalism). We prefer to use one single formalism for both implementations and specifications, and we want to stay close to the MSC specification formalism. Therefore we use a partial order semantics [Pra86] for our study, and sketch the relation with operational formalisms. In addition to the partial order model in [Pra86,KL98], we introduce a way to model deadlocks and a more sophisticated way to deal with communication.

Overview In Section 2 we introduce our partial order model, which we extend with communication in Section 3. These two sections are rather independent from MSC, but they are the basis of the semantics of compositional MSC in Section 4. In Section 5 we define the typical way of synthesizing an implementation; trace equivalence between specifications and such implementations is studied in Section 6. Finally in Section 7 we classify various realizability criteria. The conclusions and further work are presented in Section 8.

2 Extended partial order model

In this section we define a partial order model and extend it with deadlocks, to make it suitable for studying realizability criteria.

2.1 Running example

We illustrate our techniques using a running example. Figure 1 contains a (high-level) MSC consisting of the three basic MSCS ex1, ex2 and ex3. It specifies the behavior of process instances X and Y, such that first the behavior of ex1 occurs, followed by either the behavior of ex2 or the behavior of ex3. For reference purposes we have included arbitrary event names (e1 to e13) in the basic MSC.

2.2 LATERs: LAbelled Transitive Event Relations

As a semantic model of behavior, we introduce the notion of a later, which is an acronym for labelled transitive event relation. A later \((E, <, l)\) is a triple that
consists of an event set $E$, a transitive causality relation $\prec \subseteq E \times E$ and a labeling function $l : E \rightarrow L$ for a given set of labels $L$. The behavior of a later is such that any event $e : e \in E$ models a single action with label $l.e$; the event can occur at most once and it may only occur after all events $f : f < e$ have already occurred. Compared to the partial orders in [Pra86], a later is an lpo set in which the partial order constraint has been weakened.

In our running example, let laters $p_1$, $p_2$ and $p_3$ correspond to the basic MSCs ex1, ex2 and ex3, such that only the causalities per instance (on each vertical axis) are considered, i.e. without communication. So, $p_1 = (\{e_1, e_2, e_3\}, \{e_2 < e_3\}, l_1)$ and as we will see later on $l_1 = \{e_1 \mapsto !(a, X, Y), e_2 \mapsto ?(a, X, Y), e_3 \mapsto !(b, Y, X)\}$. The structure of $p_1$ can be visualized as $e_1 \rightarrow e_2 \rightarrow e_3$. In an interleaved execution model where the events are labeled with atomic actions, the maximal behaviors of a partially ordered later are its linearizations; in this example: $e_1 \cdot e_2 \cdot e_3$, $e_2 \cdot e_1 \cdot e_3$ and $e_2 \cdot e_3 \cdot e_1$. Each linearization represents an execution trace, i.e. a sequence of action labels. We prefer to reason about partial orders, because they are better related to MSC and they avoid decomposing each partial order into several over-specific total orders. Another advantage is that true concurrency can be modeled.

The most elementary laters are the empty later, with no events, and the singleton laters, with only one event with a label $k : k \in L$. We introduce the following abbreviations for them:

$$[\emptyset] = (\emptyset, \emptyset, \emptyset)$$
$$[k] = (\{e\}, \emptyset, \{[e \mapsto k]\}) \quad \text{for } k : k \in L \text{ and arbitrary } e$$

2.3 Isomorphism

The event set of a later is abstract in the sense that a consistent renaming of the events yields a later with the same behavior. This is formalized in the following notion of isomorphism. Laters $(E, \prec, l)$ and $(E', \prec', l')$ are isomorphic, denoted $(E, \prec, l) \simeq (E', \prec', l')$, if there is a bijection $\sim : \sim \subseteq E \times E'$ such that both

1. $(\forall e, e' :: e \sim e' \Rightarrow l.e \equiv l'.e')$
2. $(\forall e, f, e', f' :: e \sim e' \land f \sim f' \Rightarrow (e < f \equiv e' < f'))$

Relation $\simeq$ is an equivalence relation. In what follows we will hardly mention $\simeq$ explicitly, and implicitly assume that where necessary $\simeq$ has been exploited to obtain suitable laters, e.g. ones that are event disjoint.

2.4 Elementary later operators

We often need to relate events to the instance (i.e. computational unit or process) in which they occur. We assume a fixed set of instance names $I$, and a function\footnote{For a later $(E, \prec, l)$, [HJ00] uses the slightly different function $\phi' : E \rightarrow I$, which can be obtained from our later-independent $\phi$ as follows: $\phi'.e = \phi.(l.e)$ .}
\( \phi : L \rightarrow I \) that maps labels to the instance in which the actions with that label occur. To construct larger laters from the elementary laters, we use the following elementary operators on event disjoint laters (i.e. \( E_p \cap E_q = \emptyset \)):

\[
(E_p, <_p, l_p) \parallel (E_q, <_q, l_q) = (E_p \cup E_q, <_p \cup <_q, l_p \cup l_q)
\]

\[
(E_p, <_p, l_p) \circ S (E_q, <_q, l_q) = (E_p \cup E_q, <_p \cup <_q \cup l_p \cup l_q)
\]

where \(<_{St} = E_p \times E_q\)

\[
(E_p, <_p, l_p) \circ W (E_q, <_q, l_q) = (E_p \cup E_q, (<_p \cup <_{St} \cup <_q) \triangleright, l_p \cup l_q)
\]

where \(<_{St} = \{(e, f) | e, f : e \in E_p \land f \in E_q \land \phi.(l_p, e) = \phi.(l_q, f)\}\)

Operator \(\parallel\) denotes parallel composition, and operators \(\circ_S\) and \(\circ_W\) denote strong and weak sequential composition, respectively. These operators are associative and they have unit element \([e]\). Since parallel composition is also commutative, we can use \(\parallel\) as a quantifier.

In our running example, \(\phi.(!((a, X, Y)) = X\) and \(\phi.(?((a, X, Y)) = Y\). Let laters \(p_4\) and \(p_5\) be defined as \(p_4 = p_1 \circ_W p_2\) and \(p_5 = p_1 \circ_W p_3\). The structure of \(p_5\) is visualized as \(\begin{array}{ccc}
\circ_1 & \circ_2 & \circ_3 \\
\circ_8 & \circ_9 & \circ_{10} & \circ_{11} & \circ_{12} \\
\circ_6 & \circ_7 & \circ_8 & \circ_9 & \circ_{10} & \circ_{11} & \circ_{12} & \circ_{13}
\end{array}\) .

### 2.5 Deadlocks

A later \((E, <, l)\) contains a deadlock if there is an event \(e : e \in E\) such that \(e < e\). Conversely, a later is deadlock-free if the (transitive) causality relation is a strict partial order (i.e. irreflexive, asymmetric and transitive). These definitions are consistent, since asymmetry implies irreflexivity, and transitivity plus irreflexivity implies asymmetry. In particular, all laters that can be obtained from the elementary laters using the elementary later operators are deadlock-free.

For example, consider later \(p'_5\) (to be defined in Section 3) with the following structure: \(\begin{array}{ccc}
\circ_1 & \circ_2 & \circ_3 \\
\circ_8 & \circ_9 & \circ_{10} \\
\circ_6 & \circ_7 & \circ_8 & \circ_9 & \circ_{10} & \circ_{11} & \circ_{12} & \circ_{13}
\end{array}\) . In this later there is a circular dependency between events \(e_{10}\) and \(e_{11}\). From the transitivity of relation \(<\) it follows that \(e_{10} < e_{10}\), hence \(e_{10}\) is a deadlock.

The interpretation of the causality relation is such that the set of events “behind any deadlock” cannot occur either. We define the set of deadlocked events \(\Delta\) for a later \((E, <, l)\) as follows:

\[
\Delta.(E, <, l) = \{f | e, f : e \in E \land f \in E \land e < e \land e < f\}
\]

In our example we obtain \(\Delta.p'_5 = \{e_{10}, e_{11}, e_{12}, e_{13}\}\), and hence events \(e_1, e_2, e_3, e_8\) and \(e_9\) are the only events that can occur in later \(p'_5\).

### 2.6 Prefix

A natural way to compare laters is to compare their possible behaviors. If all possible behaviors of a later \(p\) are contained in the possible behaviors of a later \(q\), we call \(p\) a prefix of \(q\). To determine whether \(p\) is a prefix of \(q\), we only need to consider the deadlock-free part of \(p\). If \(p\) is a prefix of \(q\), then \((1)\) \(p\) may contain
fewer events than \( q \), (2) on this smaller event set, \( p \) may contain more causalities than \( q \), (3) \( q \)’s labeling of events is respected by \( p \), and (4) for each event that is in both \( p \) and \( q \), all events that precede the event in \( q \) are also in \( p \).

Formally, later \( p \) is a prefix of later \( q \), denoted \( p \preceq q \), if for some laters \((E_p, <_p, l_p) \simeq p \) and \((E_q, <_q, l_q) \simeq q \) the following four conditions hold:

1. \( E_p \subseteq E_q \)
2. \( <_q \cap (E_p \times E_p) \subseteq <_p \)
3. \( l_p \cap (E_p \times L) = l_q \cap (E_p \times L) \)
4. \((\forall e, f :: e <_q f \land f \in E_p \Rightarrow e \in E_p)\)

where \( E_p = E_p \setminus \Delta. \)

The running example several prefix relations hold, such as \( p_1 \preceq p_4 \) and \( p_1 \preceq p_5 \).

As a corollary of \( p \preceq q \), we have \( E_p \subseteq E_q \) for \( E_q = E_q \setminus \Delta. \). Prefix order \( \preceq \) is a pre-order (i.e. reflexive and transitive) with smallest element \( \epsilon \).

Some typical prefixes are \( p \preceq p \parallel q \), \( q \preceq p \parallel q \), \( p \preceq p \circ S_q \) and \( p \preceq p \circ W_q \).

In comparison with [KL98], our definition is more explicit, it can deal with \( <_q \cap (E_p \times E_p) \) to be strictly smaller than \( <_p \).

Parallel composition is monotonic in both arguments, while both kinds of sequential composition are only monotonic in their second argument (since deadlocks are invisible). A special kind of prefix is a causality extension:

\[ < \subseteq <' \Rightarrow (E, <', l) \preceq (E, <, l) \]

As an example consider later \( p_5' \), which is a causality extension of later \( p_5 \).

### 2.7 Projection

To restrict the set of events of a later, we define a projection operator \( \pi \) that restricts a later to the events in instance \( i \) as follows:

\[ \pi_i.(E, <, l) = (F, < \cap (F \times F), l \cap (F \times L)) \]

where \( F = \{ e :: e \in E \land \phi.(l.e) = i \} \)

Its relation with parallel composition is \( p \preceq (\| i :: i \in I : \pi_i.p) \), and it is monotonic with respect to causality extensions:

\[ < \subseteq <' \Rightarrow \pi_i.(E, <', l) \preceq \pi_i.(E, <, l) \]

### 2.8 Sets of laters

Usually a single later cannot describe all possible behavior of a system. Therefore we study a set of laters (which is the notion of process in [Pra86], and pomset in [KL98]), which represents the set of behaviors of the individual laters. We lift each elementary later operator \( \oplus \) and the projection operator \( \pi \) as follows:

\[ P \oplus Q = \{ p \oplus q :: p \in P \land q \in Q \} \]

\[ \pi_i.P = \{ \pi_i.p :: p \in P \} \]
To lift the prefix order \(\preceq\), we define order \(\sqsubseteq\) as follows:

\[
P \sqsubseteq Q \equiv (\forall p : p \in P : (\exists q : q \in Q : p \preceq q))
\]

Order \(\sqsubseteq\) is a pre-order with smallest element \(\emptyset\). Like before, parallel composition is monotonic in both arguments, while both kinds of sequential composition are only monotonic in their second argument. Relation \(\doteq\) defined as

\[
P \doteq Q \equiv P \sqsubseteq Q \land Q \sqsubseteq P
\]

is an equivalence relation. Equivalence \(P \doteq Q\) denotes that \(P\) and \(Q\) have the same sets of deadlock-free prefixes, which means that they are trace equivalent.

3 Asynchronous communication

In this section we develop an operator that introduces in a later the causalities that correspond to asynchronous message communication. To model distributed systems with communication via message passing, some labels are used to denote sending or receiving a message. The most liberal causalities are obtained by matching sends and receipts in their order of occurrence. This does not require that messages with identical names are communicated in FIFO order.

3.1 Label-wise trichotomy

To match events properly, we need to determine the order in which events with identical labels occur. For simplicity reasons, we assume for each label that the events with that label are totally ordered; at least, in the deadlock-free part of the later. Since this deadlock-free part is strict partially ordered, we only need trichotomy (or comparability) for events with identical labels. For notational convenience, we require this property for the whole later and for all labels.

The label-wise trichotomy property \(T\) is defined as follows:

\[
T.P \equiv (\forall p : p \in P : T.p)
\]

\[
T.(E, <, l) \equiv (\forall e, f :: l.e = l.f \Rightarrow e = f \lor e < f \lor f < e)
\]

As we will see in Section 4, this only imposes a few, acceptable restrictions on MSCs. This property is maintained under causality extensions and event restrictions, it holds for the elementary later, and it is maintained under sequential composition; only for a parallel composition \((E_p, <_p, l_p) \parallel (E_q, <_q, l_q)\) label-disjointness is required, i.e. \((\forall e, f : e \in E_p \land f \in E_q : l_p.e \neq l_q.f)\).

3.2 Communication causalities

We define operator \(\Gamma.p\), which introduces the communication causalities in a later \(p\). For compositional MSC, we must also address communication between two sequentially composed laters. Therefore we introduce an extra parameter \(t\) to denote the entire preceding behavior of later \(p\) in terms of a later.
For each message \( m \), we must ensure that each receipt event (with label \(?m\)) is preceded by the corresponding matching send event (with label \(!m\)). In case there are more receive events than send events, these remaining receipt events are turned into deadlocks. Thus we obtain (provided \( T.t \) and \( T.P \) hold):

\[
\begin{align*}
\Gamma^t.P &= \{\Gamma^t.p \mid p : p \in P\} \\
\Gamma^t.(E, <_b, l) &= (E, (<_b \cup <_c) \cup <_d, l) \\
\text{where } <_c &= <_c' \cap (E \times E) \text{ and } <_d = <_d' \cap (E \times E) \\
\text{and } (E', <', l') &= t \circ_W (E, <_b, l) \text{ and } \overline{E'} = E \setminus \Delta(E', <', l') \\
\text{and } <'_c &= \{(e, f) \mid e, f, m : e \in E' \land f \in \overline{E'} \land l'.e = !m \land l'.f = ?m \land (\# g :: g <' e \land l'.g = !m)\} \\
\text{and } <'_d &= \{(f, f) \mid f, m : f \in \overline{E'} \land l'.f = ?m \land (\# g :: g <' f \land l'.g = !m)\}
\end{align*}
\]

In this definition, first an auxiliary later \((E', <', l')\) is computed as the sequential composition of \( t \) and \((E, <_b, l)\). Then causalities <\(_c\) are defined for the matching communications, and causalities <\(_d\) are defined for the deadlocked receipt events. Finally, only the causalities on events \( E \) (i.e. not on events from previous behavior \( t \)) are added to later \((E, <_b, l)\).

For the running example, we define later \( p_4' = \Gamma^{[6]}p_4 \) and \( p_5' = \Gamma^{[7]}p_5 \). When visualizing \( p_4' \) and \( p_5' \), we add the additional communication causalities according to <\(_c\) with dashed arrows, and the additional deadlock causality for unmatched receipts (\(<'_d\)) with a dotted arrow as follows:

\[
\begin{align*}
\Gamma^{[6]}p_4' \quad \Gamma^{[7]}p_5'
\end{align*}
\]

For \( p_4' \) this then boils down to:

\[
\begin{align*}
\Gamma^{[6]}p_4' \end{align*}
\]

For \( p_5' \), the result was already visualized in Section 2.

The role of parameter \( t \) of \( \Gamma \) is illustrated in the following important property of sequential composition (see also Section 6):

\[
\Gamma^t.(\{p\} \circ_W Q) \simeq \Gamma^t.(\{p\} \circ_W \Gamma^{id\circ_W p}.Q)
\]

Since \( \Gamma \) is a causality extension, it maintains predicate \( T \). However, \( \Gamma \) can introduce deadlocks. The following are some other properties of \( \Gamma \):

\[
\begin{align*}
\text{(shrinking)} \quad & \Gamma^t.p \preceq p \\
\text{(idempotence)} \quad & \Gamma^t.p = \Gamma^t.(\Gamma^t.p) \\
\text{(monotonicity)} \quad & p \preceq q \Rightarrow \Gamma^t.p \preceq \Gamma^t.q
\end{align*}
\]

These properties can even be generalized to sets of laters.

4 Semantics of compositional MSC

Using the preceding concepts, we define a semantics of compositional MSC as an extension of the MSC semantics of [KL98]. For simplicity reasons, we delay
the introduction of the communication causalities; in Section 6 we will show how they can be introduced earlier (like in [KL98]). We start by giving the semantics of basic MSC, then the semantics of high-level MSC, and finally we complete this semantics by including the communication causalities.

4.1 Basic MSC

The semantics (without communication) of basic MSC $B$ in instance-oriented textual representation [Ren99] is defined as a later $M_{bmsc}[B]$ as follows:

$$M_{bmsc}[\{\}] = [\varepsilon]$$
$$M_{bmsc}[\text{inst } i; S \text{ endinst}; B] = M_{inst}[S](i) \parallel M_{bmsc}[B]$$
$$M_{bmsc}[(a)](i) = [\varepsilon]$$
$$M_{bmsc}[S](i) = M_{inst}[a](i) \circ S \circ M_{inst}[S](i)$$
$$M_{inst}[\text{in } n \text{ from } j](i) = (\pi(n, j, i))$$
$$M_{inst}[\text{out } n \text{ to } j](i) = (\sigma(n, i, j))$$
$$M_{inst}[\text{local } b](i) = (b(i))$$
$$M_{inst}[\text{co } () \text{ endco}](i) = [\varepsilon]$$

Function $\phi$ can then be defined as follows: $\phi.((n, j, i)) = i$, $\phi.((n, i, j)) = i$ and $\phi.(b(i)) = i$. By construction, each later $M_{bmsc}[..]$ is a strict partial order.

To ensure that predicate $T$ is satisfied, we assume that no instance name occurs more than once per bMSC [Ren99], and we require that in each co-region the events are label disjoint. The interest in co-regions is usually very limited (they are completely excluded in [HJ00,GMP03]), so this is no severe restriction. The unrealistic assumption that for each message name there is at most one send event and at most one receipt event per bMSC [KL98], is not required here.

4.2 High-level MSC

The semantics (without communication) of high-level MSC $A$ in textual representation is defined as a set of laters $M_{hmsc}[A]$ as follows:

$$M_{hmsc}[\text{empty}] = \{[\varepsilon]\}$$
$$M_{hmsc}[\text{msc } \text{name}; B \text{ endmsc}] = \{M_{hmsc}[B]\}$$
$$M_{hmsc}[A \text{ seq } B] = M_{hmsc}[A] \circ W M_{hmsc}[B]$$
$$M_{hmsc}[A \text{ alt } B] = M_{hmsc}[A] \cup M_{hmsc}[B]$$

By construction, each later in $M_{hmsc}[..]$ is a strict partial order, and satisfies predicate $T$. We do not explicitly address iteration, since it is just repeated sequential composition. Sometimes the parallel composition of high-level MSCs, denoted by $\text{par}$, is also considered. Its semantics can easily be expressed in terms of operator $\parallel$ on sets of laters, but we will not consider it in our study.
4.3 MSC

Finally we introduce the causalities imposed by communication:

\[ M_{\text{msc}}[A] = M^{[\|]}_{\text{msc}}[A] \]
\[ M^{T}_{\text{msc}}[A] = \Gamma^{T}.M_{\text{msc}}[A] \]

This is a proper definition since \( M_{\text{msc}}[A] \) satisfies predicate \( T \). By construction, predicate \( T \) also holds for \( M^{T}_{\text{msc}}[A] \). Note that the application of \( \Gamma^{T} \) may introduce deadlocks, which violate the strict partial order property. This illustrates one of the reasons for our extended partial order semantics.

Using the example laters from Sections 2 and 3, the semantics of the MSC in Figure 1 corresponds to \( \Gamma^{[\|]}.(\{p_1\} \circ_W (\{p_2\} \cup \{p_3\})) \), which simplifies via \( \{\Gamma^{[\|]}.(p_1 \circ_W p_2), \Gamma^{[\|]}.(p_1 \circ_W p_3)\} \) into \( \{p'_4, p'_5\} \). These two laters represent the possibility of either performing ex1 followed by ex2, or ex1 followed by ex3.

In [GMP03] there is a restriction that receive events in bMSCs may not be matched to send events in future bMSCs. In [MM01] an extension is proposed that drops this restriction. We consider the extension, since the original restriction conflicts with elegant rules, like sequential composition of two bMSCs being equal to simply connecting the instance axis [Ren99].

5 Implementations

In this section we explain how specifications are implemented. The difference between a specification and an implementation is that a specification describes behavior in terms of all instances, while an implementation describes behavior in terms of each individual instance. Thus an implementation for an instance can be represented by a set of laters that contain events of that instance only.

To synthesize an implementation, the specification is decomposed according to the instances. The joint execution behavior of an implementation is obtained by recomposing the instances. We do not consider the unusual implementation with message parameters proposed in [Gen05], which effectively boils down to renaming the messages and shifting the moments of choice. In such an implementation, additional parameters in a request message are sometimes used to fix the choice that should made by the receiver of the request.

5.1 Decomposition

The typical decomposition \( D \) of a set of laters \( M \) to its instances is:

\[ D.M = \{[i \mapsto \pi_i.M] \mid i : i \in I\} \]

In this set, each instance name is mapped to the corresponding projection of \( M \). Since projection is an event restriction, predicate \( T \) is maintained.

For our running example, the decomposition of the laters, \( D.\{p'_4, p'_5\} \), yields the following: \( \{[X \mapsto \{[e_1 \rightarrow e_4 \rightarrow e_5], [e_1 \rightarrow e_9 \rightarrow e_{10} \rightarrow e_{11} \rightarrow e_{12}]\}],
\[ [Y \mapsto \{ [e_2 \rightarrow e_3 \rightarrow e_6 \rightarrow e'_2], [e_2 \rightarrow e_3 \rightarrow e_8 \rightarrow e_{13}]\} \right\}.\)
Let us briefly investigate what might be lost by decomposition. For a singleton set \( \{(E, \prec, l)\} \), note that \( E \) and \( l \) are partitioned per instance, and hence only the causalities between different instances are lost. For each later in a larger set \( M \), also the link between its projections in the different instances is lost.

5.2 Recomposition

To study the joint execution behavior of the decompositions, the decomposition has to be recomposed. Using the definition from the previous section, the typical recomposition \( R \) of a decomposition becomes:

\[
R^t.\{[i \mapsto \pi_i.M] \mid i : i \in I\} = \Gamma^t.(\{[i : i \in I : \pi_i.M]\}
\]

This is a proper definition provided \( T.M \) holds, since \( T \) is maintained under parallel composition with disjoint labels. The projections are label-disjoint, since for each label \( k \) all events with that label belong to one instance, viz. \( \phi.k \).

We emphasize that \( R^t \circ D \), where \( \circ \) denotes function composition, is not monotonic with respect to \( \sqsubseteq \). For causality extensions like \( \Gamma^t \), we have:

\[
(R^t \circ D).P \sqsubseteq (R^t \circ D).P
\]

5.3 Implementations in operational formalisms

Using our later representation, implementations in operational formalisms can easily be obtained. In an interleaved execution model where the labels denote atomic actions, the maximal behaviors of a single later are the linearizations of the maximal deadlock-free prefix. The set of maximal behaviors of a set of laters is the union of the linearizations of the individual laters. In turn, linearizations can easily be transformed to process algebraic expressions using the delayed choice operator [BM95]. The implementation of our running example corresponds to the following CSP-style implementation:

\[
\begin{align*}
X : & !a \cdot ( ?b \cdot !c + ?d ) \\
Y : & ?a \cdot !b \cdot ( ?c + !d \cdot ?c )
\end{align*}
\]

6 Relation between specification and implementation

In this section, we investigate whether compositional MSC specifications are trace equivalent to their implementations, i.e. for all \( A \) and \( t \):

\[
M^t[A] \equiv (R^t \circ D).M^t[A]
\]

For details of the proofs we refer to [MRW06].
6.1 The implementation contains the specification

In this section we show that the specification is contained in the implementation, i.e. for all $A$ and $t$: $M^t_{\text{mac}}[A] \subseteq (R^t \circ D).M^t_{\text{mac}}[A]$. It can be proved as follows:

$$(R^t \circ D).M^t_{\text{mac}}[A] \quad \subseteq \quad (R^t \circ D).M^t_{\text{mac}}[A]$$

\[
\begin{align*}
\Gamma^t.\{ & \{ \text{property of } \pi \text{ and } ||; \text{monotonicity of } \Gamma \} \\
& \{ \text{definition of } M^t_{\text{mac}}[A]; \text{idempotence of } \Gamma \} \\
& M^t_{\text{mac}}[A] 
\end{align*}
\]

6.2 The specification contains the implementation

In this section we derive conditions under which the implementation is contained in the specification, i.e. for all $A$ and $t$: $(R^t \circ D).M^t_{\text{mac}}[A] \subseteq M^t_{\text{mac}}[A]$. We will set up an inductive argument based on the structure of the high-level MSC. We assume that the following rewrite rules have been applied:

- $(\text{empty}) \text{ seq } C \rightarrow C$
- $(A \text{ seq } B) \text{ seq } C \rightarrow A \text{ seq } (B \text{ seq } C)$
- $(A \text{ alt } B) \text{ seq } C \rightarrow (A \text{ seq } C) \text{ alt } (B \text{ seq } C)$

These rules do not change the occurrences of choice, but they ensure that the first argument of sequential composition is just a single bMSC. Using the property of $\Gamma$ and $\circ W$ in Section 3, we derive an alternative characterization of $M^t_{\text{mac}}[\ldots]$ in which communication is addressed earlier (like in [KL98]):

\[
\begin{align*}
M^t_{\text{mac}}[\text{msc name}; A \text{ endmsc}] &= M^t_{\text{mac}}[\text{msc name}; A \text{ endmsc seq empty}] \\
M^t_{\text{mac}}[\text{empty}] &= \{[]\} \\
M^t_{\text{mac}}[\text{msc name}; A \text{ endmsc seq } B] &= \Gamma^t.(\{M^t_{\text{mac}}[A]\} \circ_W M^{t \circ_W M_{\text{mac}}[A]}[B]) \\
M^t_{\text{mac}}[A \text{ alt } B] &= M^t_{\text{mac}}[A] \cup M^t_{\text{mac}}[B]
\end{align*}
\]

Empty For sake of space, we omit the very simple proof of this base case.

Sequential composition This inductive case can be proved as follows:

\[
\begin{align*}
(R^t \circ D).M^t_{\text{mac}}[\text{msc name}; A \text{ endmsc seq } B] &\quad \supseteq \quad (R^t \circ D).M^t_{\text{mac}}[\text{msc name}; A \text{ endmsc seq } B] \\
\Gamma^t.(\{M^t_{\text{mac}}[A]\} \circ_W M^{t \circ_W M_{\text{mac}}[A]}[B]) &\quad \supseteq \quad (R^t \circ D).M^t_{\text{mac}}[\text{msc name}; A \text{ endmsc seq } B] \\
\Gamma^t.(\{M^t_{\text{mac}}[A]\} \circ_W M^{t \circ_W M_{\text{mac}}[A]}[B]) &\quad \supseteq \quad (R^t \circ D).M^t_{\text{mac}}[\text{msc name}; A \text{ endmsc seq } B] \\
\Gamma^t.(\{M^t_{\text{mac}}[A]\} \circ_W M^{t \circ_W M_{\text{mac}}[A]}[B]) &\quad \supseteq \quad (R^t \circ D).M^t_{\text{mac}}[\text{msc name}; A \text{ endmsc seq } B]
\end{align*}
\]
The step marked $\bullet$ follows from the following rule, where $m$ denotes a later that does not order events in different instances, and $M$ denotes a set of laters:

$$(R^t \circ D).\{(m) \circ_W M\} \equiv I^t.\{(m) \circ_W (R^t \circ_W m \circ D) \circ M\}$$

Alternative composition

This inductive case can be proved as follows:

$$(R^t \circ D).M_{\text{lim}}^{t}[A \text{ alt } B]$$

= \{alternative characterization\}

$$(R^t \circ D).M_{\text{lim}}^{t}[A] \cup M_{\text{lim}}^{t}[B]$$

\(\sqsubseteq\) \{alternative characterization\}

$$(R^t \circ D).M_{\text{lim}}^{t}[A] \cup (R^t \circ D).M_{\text{lim}}^{t}[B]$$

\(\equiv\) \{induction hypothesis (twice)\}

$$M_{\text{lim}}^{t}[A] \cup M_{\text{lim}}^{t}[B]$$

\(\equiv\) \{alternative characterization\}

$$M_{\text{lim}}^{2}[A \text{ alt } B]$$

The step marked $\blacklozenge$ is not only a sufficient condition, but also a necessary one. Since it does not hold for each MSC, we will study it further.

6.3 Sound choice

Checking condition $\blacklozenge$ is quite involved in practice, since by definition of $R^t \circ D$ arbitrary combinations of projected laters (i.e. from both $M_{\text{lim}}^{t}[A]$ and $M_{\text{lim}}^{t}[B]$) need to be considered. In Section 7 we will relate various realizability criteria to this condition, but in this section we first strengthen it into a more convenient condition for this purpose; for the details we refer to [MRW06].

We strengthen condition $\blacklozenge$ into what we call the sound choice property: there exists an instance $k$ such that for each instance $j : j \neq k$ both

- $\forall g :: [I \rightarrow M_{\text{lim}}^{t}[A]], n : n \in \pi_j,M_{\text{lim}}^{t}[B] \land \{n\} \not\subseteq \pi_j,M_{\text{lim}}^{t}[A]$: 
  $$I^t.((||i : i \neq j : \pi_i,g_i|| \parallel n) \leq I^t.((||i : i \neq j : \pi_i,g_i||)$$

- $\forall h :: [I \rightarrow M_{\text{lim}}^{t}[B]], m : m \in \pi_j,M_{\text{lim}}^{t}[A] \land \{m\} \not\subseteq \pi_j,M_{\text{lim}}^{t}[B]$: 
  $$I^t.((||i : i \neq j : \pi_i,h_i|| \parallel m) \leq I^t.((||i : i \neq j : \pi_i,h_i||)$$

Here functions $g$ and $h$ represent a chosen later per instance. Later $n : n \in \pi_j,M_{\text{lim}}^{t}[B] \land \{n\} \not\subseteq \pi_j,M_{\text{lim}}^{t}[A]$ denotes a later from MSC $B$ that is no prefix of any later from MSC $A$. Note that behaviors occurring both in MSC $A$ and MSC $B$ are not problematic for the choice between $A$ and $B$. The $\leq$-term expresses that later $n$ (or later $m$) cannot perform any behavior. Instance $k$ and condition $j \neq k$ ensure that some instance may have initiative.

The choice in our running example is not a sound choice, as can be pointed out by considering both options for $k$. For $k = X$, we can choose $n = \pi_Y.(I^{p_1} \cdot p_3)$ and $g_X = I^{p_1} \cdot p_2$, which violate the first $\leq$ term; and similarly for $k = Y$. We will discuss it in more detail using the non-local choice criterion in Section 7.

Notice that instead of considering arbitrary combinations of projected laters, on the left-hand side of the $\leq$ in this condition, the combinations of projected laters contain only one later $n$ from $B$, or only one later $m$ from $A$ respectively. Finally we stress that this condition is stronger than condition $\blacklozenge$. 
7 Realizability criteria

The sound choice property of the previous section implies that the specification and the implementation are trace equivalent; otherwise the specification may not be realizable. In this section we convert the realizability criteria from [MGR05] to high-level MSCs with binary choice, and generalize them to compositional MSC with co-regions. We first depict how the criteria are classified in comparison with sound choice and derived condition:

<table>
<thead>
<tr>
<th>Derived condition</th>
<th>Non-local choice</th>
<th>Non-deterministic choice</th>
<th>Propagating choice</th>
<th>Race choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>sound choice</td>
<td>¬</td>
<td>¬</td>
<td>¬</td>
<td></td>
</tr>
</tbody>
</table>

Example MSCs for (combinations of) these criteria can be found in [MGR05].

7.1 Non-local choice

A choice between two MSCs is local if at most one instance has initiative in these MSCs; otherwise several instances can independently start executing different MSCs. An instance has initiative in an MSC if some first event of the instance is labeled with either an internal action, or sending a message, or receiving a message that was sent before the choice. The choice in our running example is non-local, since due to events \( e_4 \) and \( e_8 \) both \( X \) and \( Y \) have initiative.

Non-local choice follows naturally from sound choice, and in particular from its \( \preceq \)-terms. Observe that a later \( n \) is likely to be problematic if for each label-disjoint later \( x \) we have \( \Gamma^x.(x\|n) \not\preceq \Gamma^x.x \). This condition follows from \( \Gamma^x.n \not\preceq [\epsilon] \), which means that later \( n \) contains an initiating event. Due to condition \( j \neq k \) in the definition of sound choice, only instance \( k \) may have initiative, i.e. no two different instances, say \( i \) and \( j \), may have initiative. This leads to the non-local choice criterion:

\[
(\exists i, j, m, n :: i \neq j \land m \in \pi_i.M_{inv}^i[A] \land \{m\} \not\subseteq \pi_i.M_{inv}^i[B] \land \Gamma^i.m \not\preceq [\epsilon] \\
\land n \in \pi_j.M_{inv}^j[B] \land \{n\} \not\subseteq \pi_j.M_{inv}^j[A] \land \Gamma^i.n \not\preceq [\epsilon] )
\]

The difference with other variants of non-local choice in [BAL97, HJ00, MGR05] is in our first two conjuncts on both \( m \) and \( n \), where we ensure that sound choice is violated.

7.2 Propagating choice

Absence of non-local choice is not sufficient to guarantee sound choice. It does guarantee that there is at most one instance that determines the choice, viz. instance \( k \) in the definition of sound choice. The other instances \( j \) have no initiative and hence their chosen laters \( n \) are characterized by \( \Gamma^j.n \preceq [\epsilon] \). What remains to guarantee sound choice is that the other instances can resolve the choice, which is characterized by the propagating choice property (see also [MGR05]): for each instance \( j \) both
- $\forall g :: [I \rightarrow M^t_{\text{msc}}[A]], n : n \in \pi_j, M^t_{\text{msc}}[B] \land \{n\} \not\subseteq \pi_j, M^t_{\text{msc}}[A] \land \Gamma^t, n \preceq [e];$
  $\Gamma^t, (\{i : i \neq j : \pi_i, g_i\} \parallel n) \preceq \Gamma^t, (\{i : i \neq j : \pi_i, g_i\})$
- $\forall h :: [I \rightarrow M^t_{\text{msc}}[B]], m : m \in \pi_j, M^t_{\text{msc}}[A] \land \{m\} \not\subseteq \pi_j, M^t_{\text{msc}}[B] \land \Gamma^t, m \preceq [e];$
  $\Gamma^t, (\{i : i \neq j : \pi_i, h_i\} \parallel m) \preceq \Gamma^t, (\{i : i \neq j : \pi_i, h_i\})$

### 7.3 Non-deterministic choice

Propagating choice is an important property, but it is not easy to apply. A simple case that violates it is when the MSCs contain behaviors $m$ and $n$ that are different, although they share a common prefix. This criterion can be made more syntactic by weakening the inner existential quantification into condition $p \not\preceq [e]$. Although non-deterministic choice violates sound choice, it does not guarantee that the condition $\Delta$ in Section 6 is violated; so sound choice has been a real strengthening.

### 7.4 Race choice

Absence of non-deterministic choice is not sufficient to guarantee propagating choice. It does guarantee that the choice can be resolved when no initiating receipt event can end up receiving a message intended for a non-initial receipt event in another MSC. The other cases are characterized by the race choice criterion (see also [MGR05]). Its formal definition is very similar to the definitions of propagating choice and non-deterministic choice, see also [MRW06].

In [HJ00] the reconstructible choice criterion is proposed in order to guarantee realizability. However, this claim contradicts their example of a reconstructible MSC (see Figure 15 in [HJ00]). In terms of our classification, it suffers from race choice: if instance $A$ sends message $m1$ before message $m5$, instance $B$ may receive message $m6$ (related to $m5$) before message $m3$ (related to $m1$).

### 8 Conclusions and further work

We have developed a denotational semantics for compositional MSC through our extension of pomsets with deadlocks. In this formalism we have studied realizability, especially of the choice construct. We have discussed various proposed realizability criteria and shown completeness of our classification in [MGR05].
Realizability problems can also be detected by verifying the implementation [UKM03]. However, it is far more effective to have criteria for specifications, and to develop ways to make specifications realizable [HJ00]. For the latter, we plan to evaluate our proposals in [MG05,MGR05] using the current framework, and to automate them.

A possible extension is to explore other realizability criteria, especially since sound choice is a real strengthening. In addition, more syntactical criteria would better allow automation. Also the realizability of other MSC constructs may be studied, of which parallel composition is a challenging one.

References


