From UML 1.x to UML 2.0 Semantics for Sequence Diagrams

O. Tahir, C. Sibertin-Blanc
Université Toulouse 1 / IRIT
21 allée de Brienne F-31042 Toulouse cedex
{otahir, sibertin}@univ-tlse1.fr

Abstract. Sequence diagrams are one of the most popular elements of the UML notation to model the dynamics of systems. In versions 1.0 to 1.5 of UML, Sequence diagrams have no formal semantics, since a behavior is described as an ordering of messages and not as an ordering of sending and receiving actions. In UML 2.0, Sequence diagrams became provided with an operational semantics, based upon the semantics of Message Sequence Charts, as a partial order on the set of the sending and receiving actions. This paper investigates how Sequence diagrams designed in the UML 1.x context can be interpreted in the new UML 2.0 context.

Keywords: UML Sequence Diagrams, Message Sequence Charts, formal semantics.

1. Introduction

Sequence Diagrams (SD) are one of the UML diagrams to model the dynamic dimension of a system. They allow to describe interactions between the system and the actors of its environment or between the component objects of the system. A SD describes the sequence of communications that occur in the course of a selected system run, that is the trace of the messages that are exchanged during this run. SD are a very popular language to specify the processing of operations because of their clear graphical layout that gives an immediate intuitive understanding of the behavior.

SD have been introduced among the UML notations in accordance with the Jacobson’s interaction diagram of the Objectory method [5], to describe the various scenarios that can be followed to realize a Use Case of a system. So, SD are an enhancement of interaction diagrams gained by the use of the graphical notation of Message Sequence Charts (MSC), the trace language defined by the International Telecommunication Union [4]. However, the UML 1.x versions of UML (from version 1.0 published in 1997 to version 1.5 published in 2003 [1]) provide SD with an interpretation that differs from the MSC interpretation. In a SD, the dynamics of interaction between objects is described by a global ordering of the messages that are exchanged, while in a MSC the interaction dynamics is described by (1) a local ordering, for each object, of the message sending and receiving actions performed by this object, together with (2) the synchronizations between objects resulting from the fact that the sending of a message necessarily occurs before its reception. Thus, the difference between the UML 1.x SD
and MSC semantics is twofold: in MSC the ordering concerns the message sending and receiving actions while it concerns the messages in UML 1.x SD (so that SD are not provided with an operational semantics [6]) on the one hand, and on the other hand this ordering is global with a single clock in a SD, while it is local in MSC, each object being provided with its own clock. This results in a conflict in the reading of a SD: the representation suggested by the visual layout of a SD – the top-to-down lifelines along the time direction crossed by the horizontal synchronization messages – is not in accordance with a unique thread of control gathering all the messages in a single sequence. This conflict reflects an incompatibility between the two implicit execution models: the behavior of a distributed system that results from the synchronization of concurrent processes in MSC, and the mainly linear thread of a dialog between a user and an interactive application in UML 1.x SD.

The lack of a formal semantics for SD is one of the drawbacks of UML 1.x. Notably, it prevents to deduce the internal behavior of an active object from the set of SD where it intervenes without bringing additional specification items [7, 8, 9]; it prevents the automatic generation of code, and the use of SD as test cases for the functional validation of a system once it is developed [19]; from a cooperative engineering point of view, it entails ambiguities in the understanding of SD and thus frustrates the UML objective to ease the communication between the stakeholders of a software project. To counter these effects, many works have been done to propose a formal semantics for UML 1.x SD, e.g. [6, 11, 12, 13, 14, 15, 16, 17, 20, 21]. Different means are used in the definitions of these semantics; some of them go away from the UML definition of SD, most of them by decorating SD with additional items, and thus are not of practical use in the context of UML-based projects; others restrict the whole class of SD by only considering SD that satisfy some (often not explicitly stated) structural or behavioral properties. In any case, these semantics are defined in translating SD into a domain provided with a well-defined semantics such as Petri nets [13, 15, 17], Labeled Transition Systems [21], a Temporal Logic [6, 20], Hoare’s CSP or Milner’s CCS [16], or in MSC (more precisely, in the domain of Partially Ordered Multi-Sets [18] that is the more straight and easy way to define the semantics, in an operational forms, of MSC) [6, 11, 12, 14].

The new 2.0 version of UML [2] reduces this deficiency by making in the UML meta-model a change that modifies the SD abstract definition and provides SD with a MSC-like operational semantics: the dynamics of an interaction is no longer defined as a global ordering of messages, but as the union, for each object, of the local ordering of the message sending and receiving actions performed by this object. This change raises questions about the reuse of SD that have been earlier drawn in a UML 1.x perspective and thus require a new interpretation into the UML 2.0 semantics. What about the conversion, or translation, of UML 1.x SD into UML 2.0 SD? Let us notice that this conversion does not apply to the concrete syntax of the graphical representation of SD, since the move from the 1.x to the 2.0 version of UML concerns mainly the meta-model and the semantic dimension, while the UML 2.0 SD graphical representation just extends the UML 1.5 representation. The conversion in question is more subtle and concerns the semantics of SD. Even if UML 1.x SD do not have a formal semantics, a UML user that draws SD refers to an (eventually implicit) execution model that, in a more or less precise way, defines the possible schemes of execution for that SD; doing so he/she implicitly provides his/her diagrams with a (more or less formal) semantics. So the question raised by the move from UML 1.x to UML 2.0 SD is the following: to what extent the informal semantics that UML users have given to UML 1.5 SD is compatible with the UML 2.0 formal semantics of SD?
In previous papers ([11] and [14]), we have investigated these users’ implicit and informal semantics of UML 1.x SD, and proposed three different semantics:

- The emission-semantics that is based upon the idea that if a message \( m_1 \) precedes a message \( m_2 \), then \( m_1 \) must be created and thus be emitted before \( m_2 \);

- The MSC-semantics that breaks the global ordering of messages and only considers its projections on the lifelines of objects;

- The causality-semantics that focuses on the causality relationships between the occurrences of the message sending and receiving actions.

These semantics have been defined in terms Partially Ordered Multi-Sets [18], like the semantics of MSC, and thus reveal to be three different proposals for converting UML 1.x SD into UML 2.0 SD.

Now the question arises to compare these various semantics, in order to decide e.g. in which cases each one is more relevant than the others, or in which cases they are equivalent. The aim of this paper is to study this question, and this leads first to consider other semantics that are variations of the three above mentioned ones, and second to identify three essential structural properties of SD, that we call respectively the sequential, the local control and the local causality. Informally, a SD is sequential if each message is sent by the object that has received the previous message, a SD is locally controlled [11] if, at each step of a run, an object that has the focus can decide whether it is its turn to emit the next message of the interaction, and a SD is locally causal if within each object, each action is directly caused by the just previous one(s).

The main results of this paper is to give conditions for the realizability of each semantics and for their equivalence.

In the following of this paper, we first recall the changes from UML 1.5 to UML 2.0 with regard to SD. Section 3 gives an overview of the semantics, while sections 4 to 7 are devoted to the presentation of the linear-, emission-, MSC- and causality-semantics and give examples where each appears as being the most natural semantics, the one that captures in the best way the SD’s intuitive meaning. Theorems that use the sequential, the locally controlled and the local causality properties of SD to relate the four semantics are given in section 8.

2. UML 1.x and UML 2.0 Sequence Diagrams

The definition of SD in UML 2.0 extends the previous definition of interaction diagrams given in UML 1.5 and earlier. Both of them aim at describing the interactive behavior between the system’s components and its external actors; in both cases, interactions take the form of transmissions of messages between actors, system’s instances or objects.

This paper focuses on basic SD. A basic SD describes a finite scenario that does not comprise composition operators. In this context, in terms of concrete syntax or graphical layout, the UML 2.0 and UML 1.x SD are essentially identical¹. For illustrative purposes, the left side of Fig. 1 shows a SD depicting an interaction between the objects \( o1 \), \( o2 \) and \( o3 \). Each vertical line in the chart corresponds to one object, and is called the object’s life-line; messages exchanged between these objects are represented by horizontal arrows. The tail of each arrow corresponds to the action of sending the message, while the head corresponds to its receipt.

¹ In UML 2.0, each SD is surrounded by a frame including a compartment displaying its name.
However, beyond the similarity of the graphical representations, there are some important semantic differences between UML 2.0 and UML 1.x SD. The most noteworthy addition is the fact that an interaction is defined as an order relation between the actions of sending and receiving messages in UML 2.0, instead of an ordered collection of messages in UML 1.x. To do this, modifications are carried out on the abstract syntax of the SD, i.e. on their definition in terms of the metamodel of UML.

In the following of this section, we first recall the abstract syntax of UML 1.x SD, more precisely the UML 1.4 version published by the Object Management Group (OMG), and recently, since January 2005, standardized by the International Organization for Standardization (ISO) as ISO/IEC 19501 specification [3]; then we present the UML 2.0 SD abstract syntax.

2.1. Abstract Syntax of UML 1.x SD

Fig. 2 shows how the interactions are defined in the metamodel, as a set of partially ordered messages.

A message specifies one communication between two objects, or an object of the system and its environment. Each message references a sender and a receiver, each of them being one instance of a class of objects, or more precisely a role that plays this instance in the interaction. Each message is sent by an action which specifies the statement that causes the communication specified by the message.

The isAsynchronous attribute indicates if the message sending action is asynchronous or synchronous: in the latter case, the sender is blocked as long as the receiver does not return a result or an acknowledgement of delivery.

The dynamic aspect of interactions is described by means of two associations between the messages: the predecessor and activator associations.

- The predecessor association orders the messages, it defines a partial order between the interaction's messages, but does not precise in which order the sending and receiving actions of messages must be executed.

- The activator association defines a hierarchical relation on the set of messages, more precisely a set of trees (a forest): the activator of a message is the message that invoked the procedure that in turn invokes the current message. In fact, a set of messages activated by a message corresponds to the notion of a transaction, or a procedure.
II.2. Abstract Syntax of UML 2.0 SD

Now, we briefly present the most important concepts of the metamodel concerning basic interactions in UML 2.0. Mainly, the abstract syntax is closely inspired by the definition MSC. In this sense, the metamodel focuses on ordering sending and receiving actions that describe the operational semantics of interactions. So, underlying interaction diagrams of the metamodel are completely revised. The concept of classifierRole representing a specific way to take part in the interaction is replaced by the concept of lifeline. The concept of the message is redefined: a message consists of two event occurrences that represent respectively its sending and receiving events, each of them being an EventOccurrence. An EventOccurrence is a kind of MessagesEnd that represents what can occur at the end of a message. EventOccurrences of an interaction are partially ordered by a GeneralOrdering, a binary relation between events that tell which events must occur before and after each one Fig. 3 shows the portion of the UML 2.0 metamodel concerning basic interactions.

The dynamic aspect of interactions in UML 2.0 is modelled by the GeneralOrdering metaclass which defines the order of occurrences of the emission and reception actions. The operational semantics of a SD is thus defined by instances of the GeneralOrdering metaclass. This semantics is defined as a structure \( (A, <) \), where \( A = \{!m; m \in M\} \cup \{?m; m \in M\} \) is the set of all the message sending and receiving actions to be executed in a run of SD and \( < \) is an
acyclic and irreflexive relation on $A$. $M$ denotes the finite set of messages exchanged among the objects, and $!m$ and $?m$ denote respectively the sending and receiving action of a message $m$; the notation $a < a'$ means that action $a$ is to be performed before action $a'$.

The right side of Fig. 1 shows the UML 2.0 order of the actions included in the left side SD. In the UML 1.x context, the scheduling of sending and receiving actions of this SD is not defined; for instance, message $m_1$ has to precede message $m_2$, but it not specified whether the receiving action $?m_1$ precedes the sending action $!m_2$ or not.

3. An overview of the various semantics

To enhance the UML 1.x SD with an operational semantics of the same kind as the of UML 2.0 semantics without making modifications into their concrete or abstract syntax, we investigate the interpretation of the precede association between messages in terms of scheduling on the actions sending and receiving messages. It proves that several interpretations are possible which constitute, in fact, the various ways of interpreting in UML 2.0 the SD designed in the UML 1.x context.

According to [14], we consider that the synchronous character of a message, its return nature and the activation association of the SD metamodel can all be caught in the precede association among the messages. Then, a UML 1.x SD is defined as a 5-uple $(O, M, To, From, Pre)$, where:

- $O$ is the finite set of instances, or objects, participating in the interaction;
- $M$ is the finite set of messages exchanged among the objects, all the messages being asynchronous;
- $From$ and $To$ are two functions from $M$ to $O$. $From(m)$ denotes the instance which sends message $m$, while $To(m)$ denotes the instance which receives message $m$;
- $Pre$ is an relation defined on $M$, that corresponds to the predecessor association of the UML metamodel.

Some more precisions on properties of the Pre relation are needed. First of all, Pre is of course an acyclic relation, that is there is no set of messages of $M$ such as $m_1 Pre m_2 Pre .... m_k Pre m_1$. In addition, we have advantage to consider that Pre is an intransitive relation, that is $m_1 Pre m_2$ implies there is no $m$ such that $m_1 Pre m$ and $m Pre m_2$. This latter property allows to deal only with pairs of messages that are just consecutive and to avoid to consider ordering constraints that are redundant since they can be deduced from other elementary constraints. In the following, we will use the following terminology: intransitive order relation for an acyclic and intransitive relation, order relation for an acyclic relation, and transitive order relation for an acyclic and transitive relation. More generally, for any order relation $R$, we will notice $R°$ its intransitive restriction and $R*$ its transitive closure extension. Thus, we always have: $R° \subset R \subset R*$. For simplicity of the presentation, we will also consider that Pre is a total order relation, but definitions and results also apply in the cases where Pre is a partial order relation.

According to UML, « For each message, its predecessors are the set of Messages whose completion enables the execution of the current message. All of them must be completed before execution begins ». So, providing the message predecessor relation with a rigorous semantics relies upon the interpretation of the terms “completion” and “execution”. Is the execution of a message the performance of its sending, of its reception or both? Is the completion of a message the achievement of its sending, of its reception, or both? According to the answer given
to these questions, we get different orderings of the message sending and receiving action. The various interpretations that we consider lead to the four following main semantics.

The linear-semantics considers that a message is completed when it is received by its addressee (according to the MSC and UML semantics, the reception of a message is not its arrival or delivery at an entry port of the addressee, it is its accounting and processing), and the execution of a message starts by its sending; according to this semantics, the execution of a SD is a fully sequential process since for each messages \( m \) and \( m' \) such that \( m \preceq m' \), we have \!m < \?m < \!m' < \?m'.

The emission-semantics puts the focus on the production of messages; accordingly to this semantics, a message is considered as being completed as soon as it is emitted, and, as in the linear-semantics case, the execution of a message starts by its sending; so for each messages \( m \) and \( m' \) such that \( m \preceq m' \), we have \!m < \!m' and the sending of \( m' \) is no longer synchronized with the reception of \( m \).

The MSC-semantics accounts for the autonomous processing capabilities of objects, and completion and execution are considered as local concepts; accordingly, for a message \( m \) sent from an object \( o1 \) to an object \( o2 \), the \( o1 \) view of the \( m \) completion is the sending of \( m \), the \( o2 \) view of the \( m \) completion is the reception of \( m \), while other objects have no knowledge of \( m \). In this case, the global ordering of the \( \preceq \) relation is broken into one ordering for each object, and it induces a local ordering of the actions performed by each object; each object performs its actions in sequence, and the only synchronizations between these processes are the ones performed by the messages themselves.

The causal-semantics weakens the MSC-semantics: completion and execution are analogously local concepts and in addition attention is paid to the fact that only messages that enable the execution of another one have to be taken into account; notably, the reception of a message by an object is not directly caused by the previous sending or reception actions performed by this object. Thus, this semantics removes some ordering constraints inside each object so that the process executed by each object is not necessarily fully sequential.

All these semantics differ by the constraints they impose on the ordering of the message sending and receiving actions, that is on the level and type of concurrency they allow, inside each object and between the objects. In any case, we have to wonder whether a semantics can be supported by a given SD, that is if it is possible to design a set of processes, one for each object, that perform the sending and receiving actions so that the resulting behaviour of the whole system follows this semantics. This issue is know as the realizability question [10, 18]. More formally, let us consider a SD and \(<\) an intransitive order relation on its messages sending and receiving actions describing its operational semantics. We say that \((\text{SD}, <)\) is realisable iff for any actions \( a \) and \( a' \) such that \( a < a' \), either (1) there exists a message \( m \in M \) such that \( a = \!m \) and \( a' = ?m \), or (2) \( a \) and \( a' \) are performed by the same object, that is: for two consecutive messages, either (1) the second message is sent by the addressee of the first one or they are sent by the same object.

In particular, we will see that there are restrictions on the SD that are able to realise the linear- and emission-semantics. These restrictions may be characterised by a structural property of SD, the local control.

A SD is said to be locally controlled iff for any messages \( m \) and \( m' \) such that \( m \preceq m' \), either (1) \( \text{To}(m) = \text{From}(m') \) or (2) \( \text{From}(m) = \text{From}(m') \), that is: for two consecutive messages, either the second message is sent by the addressee of the first one or they are sent by the same object.

A SD is said to be sequential if only case (1) happens, that is \( m \preceq m' \) implies \( \text{To}(m) = \text{From}(m') \).
Each of these semantics will be defined as the union of acyclic relations on the set $A = \{!m; m \in M\} \cup \{?m; m \in M\}$ of all the message sending and receiving actions. In each case it is easy to prove that the union is also an acyclic relation, so the proof is not given. An essential relation to consider, because it is a part of any semantics, is the $\langle_{\text{sync}}$ relation that expresses the fact that the sending of a message always occurs before its reception: $\langle_{\text{sync}} = \{(!m, ?m); m \in M\}$.

The ordering constraints of this relation are mandatory in any semantics and they do not deserve additional explanation: a message can be received only if it has been previously sent! This relation is a specific part of any semantics because it contains all the synchronizations, or inter-objects ordering constraints, that take place between the objects of the SD. So the realizability notion can be expressed in terms of this relation: a semantics is realizable iff the only inter-objects constraints are those contained in the $\langle_{\text{sync}}$ relation.

4. The linear-semantics

Let’s consider the relation saying that the emission of a message requires the reception of the preceding message:

$\langle_{\text{re}} = \{(?m, !m); m, m' \in M \text{ and } m \text{ Pre } m'\}$

(where the index re stands for reception-emission), and the semantics defined by the relation $\langle_{\text{linear}} = \langle_{\text{sync}} \cup \langle_{\text{re}}$.

For any two messages $m$ and $m'$ such that $m \text{ Pre } m'$, we have $!m <_{\text{linear}} ?m <_{\text{linear}} !m' <_{\text{linear}} ?m'$, so that $\langle_{\text{linear}}$ is a total order on the set $A$ of all the sending and receiving actions if Pre is a total order. Thus, according to this semantics, all the runs of a SD follow the same fully sequential process Fig. 4 shows a sequential SD and its (realizable) linear-semantics that consists of: $!m_1 <_{\text{linear}} ?m_1 <_{\text{linear}} !m_2 <_{\text{linear}} ?m_2 <_{\text{linear}} !m_3 <_{\text{linear}} ?m_3$.

This semantics is very restrictive, and it can be realized only by sequential SD. Indeed, realizability requires that any constraint of the kind $?m <_{\text{linear}} !m'$ involves a unique object that executes the two actions $?m$ and $!m'$, since no couple $(?m, !m')$ belongs to $\langle_{\text{sync}}$; this means that $m \text{ Pre } m'$ implies that the $?m$ and $!m'$ actions are performed by the same object, $\text{To}(m') = \text{From}(m)$, which is just the definition of sequential SD. Thus, as far as we are interested in semantics that are realisable, this semantics is relevant only for sequential SD. More formally, we have the following:

**Theorem 1:** for any SD, its linear-semantics is realizable if and only if it is a sequential SD.

![Fig. 4. A sequential SD and its linear-semantics.](image-url)
5. The emission-semantics

This semantics, that has been defined and studied in [11], considers that if a message \( m \) precedes another message \( m' \), \( m \) must “exist” before \( m' \) and thus we must have \(!m < !m'\).

Considering the relation \(<_{ee}=\{(!m, !m'); m, m' \in M \text{ and } m \text{ Pre } m'\}\), the emission-semantics denoted \(<_{\text{emission}}\) we are looking for is the smaller order relation on \( A \) that includes both \(<_{\text{sync}}\) and \(<_{ee}\).

We have to consider the special case of consecutive messages \( m \) and \( m' \) such that \( \text{To}(m) = \text{From}(m') \), as, for instance, messages \( m1 \) and \( m2 \), or \( m2 \) and \( m3 \) in Fig. 4. Indeed, since we already have \(!m <_{\text{sync}} ?m\), the constraint \(!m <_{\text{emission}} !m'\) can result from a local constraint of the kind \(?m < !m'\). Such a constraint is a very natural interpretation of the precedence relation among the message along the lifeline of an object: sending \( m' \) is its reaction to receiving \( m \). So we consider the relation \(<_{\text{emission, re}}=\{(?m, !m'); m, m' \in M, m \text{ Pre } m' \text{ and } \text{To}(m) = \text{From}(m')\}\) that, gathered with \(<_{\text{sync}}\), ensures \(!m < !m'\) whenever \( \text{To}(m) = \text{From}(m') \). To deal with the cases where \( \text{To}(m) = \text{From}(m') \) does not hold, we define the relation:

\[<_{\text{emission, ee}}=\{(!m, !m'); m, m' \in M, m \text{ Pre } m' \text{ and } \text{To}(m) \neq \text{From}(m')\}\]

Now the emission-semantics is defined as the transitive closure of the union of these relations 
\[<_{\text{emission}}=\left(<_{\text{sync}} \cup <_{\text{emission, re}} \cup <_{\text{emission, ee}}\right)^*,\]
and it is the smallest order relation on \( A \) that includes both \(<_{\text{sync}}\) and \(<_{ee}\) [11].

Concerning the realizability of this semantics, let’s consider the various cases for two messages \( m \) and \( m' \) such that \( m \text{ Pre } m' \). If \( \text{To}(m) = \text{From}(m') \), the proper ordering of \(!m \text{ and } !m'\) is ensured by the \(<_{\text{sync}}\) and \(<_{\text{emission, re}}\) relations; if \( \text{From}(m) = \text{From}(m') = o \), the ordering of \(!m \text{ and } !m'\) can be locally ensured in the object \( o \) by the \(<_{\text{emission, ee}}\) relation. In the third remaining case, where \( \text{To}(m) \neq \text{From}(m') \) and \( \text{From}(m) \neq \text{From}(m') \), \(!m <_{\text{emission, ee}} !m'\) requires an inter-objects synchronisation; this constraint cannot be ensured by the \(<_{\text{sync}}\) relation since it includes only pairs of actions of the kind \((!m, ?m)\). As a consequence, the emission-semantics is not realisable in SD having consecutive messages satisfying neither \( \text{To}(m) = \text{From}(m') \) nor \( \text{From}(m) = \text{From}(m') \), that is in SD that have not a local control. The converse holds, so that we have:

Theorem 2: For any SD, its emission-semantics is realizable if and only if it is locally controlled.

Fig. 5. A SD that is neither locally controlled nor locally causal and its four different semantics.
In the emission-semantics of the SD in Fig. 5, the ordering constraint !m1 <emission !m2 prevents the realization of this semantics. Comparing the linear- and emission-semantics, it is clear that the linear one is more restrictive: <emission \( \subseteq \) <linear* and any ordering constraint in the emission-semantics also belongs to the linear-semantics; so we have the following straightforward result:

**Theorem 3:** For any SD, we have emission-semantics \( \subseteq \) linear-semantics, and linear-semantics = emission-semantics if and only if SD is sequential.

As a corollary, the emission-semantics is realizable for any sequential SD, since a sequential SD is always locally controlled, and if the emission-semantics of a SD is not realizable then its linear-semantics is not realizable too, since it is not locally controlled (theorem 2), and thus not sequential.

### 6. The MSC-semantics

With regard to the graphical layout of SD, the UML global ordering of messages suggests an in width reading of a SD, step by step from top to down, going from the horizontal arrow of a message to the just below message arrow. This UML’s implicit semantics is accurately caught by the emission-semantics that puts all the message sending actions into the same unique thread of control; it is well suited for modelling the runs of interactive applications in information systems, where only the user does control the activity of the system’s objects. But this semantics is not realisable for most SD, the ones that are not locally controlled, and this is a serious drawback.

The main motivation for the **MSC- and causal-semantics** is to provide any SD, be it locally controlled or not, with a realisable operational semantics, that is a semantics where all the inter-objects synchronisations are ensured by the exchanged messages. To this end, they go a little away from the UML implicit semantics, in that they transform the UML global ordering of messages into a set of local orderings, one order relation for each object. Doing so, they give an equal importance to the vertical dimension of the graphical representation of SD and consider the actions that are encountered along the life line of each object. This view is well suited for modelling distributed systems, where objects are active processes that synchronise their behaviours by message sending.

The MSC-semantics reads a SD as a Message Sequence Chart [4]. Its definition uses the **Pre** relation to define a local ordering of the messages sent or received by each object, and then schedules the corresponding actions. **Pre**<sub>local</sub> is the order according to which, while following the life line of any object from top to down, we encounter the sent or received messages. It is defined in the following way:

\[ Pre_{local} = \{ (m, m') \} \text{ such as:} \]

1. \( m, m' \in M \) are sent or received by a same object \( o \),
2. \( m \ Pre^* m' \), that is \( m \) precedes \( m' \),
3. for each \( m'' \in M \) sent or received by object \( o \), \( (m'' \ Pre^* m) \) or \( (m' Pre^* m'') \).

The second clause considers the normal closure **Pre**<sup>*</sup> of **Pre** because each object processes only some messages, e.g. in the object \( o1 \) in Fig. 5, the message processed after \( m1 \) is \( m3 \), not \( m2 \). The third clause allows to consider only pairs of messages that are just consecutive with
regard to an object so that $\text{Pre}_{\text{local}}$ is a intransitive order relation that contains no transitive implicates.

Now we define the total ordering of actions executed by each object as

$$<_{\text{MSC, local}} = <_{\text{MSC, ee}} \cup <_{\text{MSC, er}} \cup <_{\text{MSC, re}} \cup <_{\text{MSC, rr}},$$

where:

$$<_{\text{MSC, ee}} = \{(m, m'); m \text{Pre}_{\text{local}} m' \text{ and } \text{From}(m) = \text{From}(m')\},$$

$$<_{\text{MSC, er}} = \{(m, m'); m \text{Pre}_{\text{local}} m' \text{ and } \text{From}(m) = \text{To}(m')\},$$

$$<_{\text{MSC, re}} = \{(m, m'); m \text{Pre}_{\text{local}} m' \text{ and } \text{To}(m) = \text{From}(m')\},$$

$$<_{\text{MSC, rr}} = \{(m, m'); m \text{Pre}_{\text{local}} m' \text{ and } \text{To}(m) = \text{To}(m')\}.$$

and finally $<_{\text{MSC}} = (<_{\text{MSC, local}} \cup <_{\text{sync}})^*.$

Each object performs a fully sequential process, and the synchronizations between these processes are performed by the messages themselves. The following is straightforward:

**Theorem 4:** For any SD, we have $<_{\text{MSC}} \subseteq <_{\text{linear}}^*$, that is MSC-semantics $\subseteq$ linear-semantics.

### 7. The causal-semantics

The *causal-semantics* weakens the MSC-semantics and keeps an ordering constraint between two actions in an object only if the second action directly results from, or is a consequence of the execution of the first one. According to this semantics, the sending of a message is caused by the reception of the precedent messages, it is the object’s reaction to this reception, and thus the sending of a message is postponed after the reception of all its precedent messages. Second, if two successive messages are sent just one after the other by the same object, this ordering is meaningful of the object’s behavior and the respective emission actions must be ordered in the same way. On the other hand, an object does not control the fact that it has to receive a message, since this message was sent by another object. Thus the causal-semantics considers that a receive actions has no local predecessor action in the same object. These causal rules are shown in Fig. 6.a; inside object $o$, there is no causal relationship between $m1$ and $m2$, neither towards $m5$. Thus, this semantics does not impose a total serialization on the emission and reception actions of successive messages executed by an object [14].

![Ordering constraints between actions within an object.](image)

(a) rules for the causal semantics, (b) an additional constraint for the emission and causal semantics

**Reception-Emission rule.** A reception of a message is the cause of the actions of emission that are directly consecutive to it: these emissions constitute the reaction of the object to the messages it has just received. In terms of causality, a send action is caused by any receive action that has been specified to occur earlier. In Fig. 6.a, the send action of message $m3$ is caused by
the reception actions of \( m_1 \) and \( m_2 \), that means the message \( m_3 \) can be sent only after having received the two messages \( m_1 \) and \( m_2 \). Formally:

\[
\langle \text{causal, re} \rangle = \{ (?m, !m') \mid m, m' \in M \text{ and } \\
1. \quad \text{To}(m) = \text{From}(m'), \\
2. \quad m \text{ Pre}_\text{local} m', \text{ that is } m \text{ precedes } m', \\
3. \quad \text{for any } m'' \in M \text{ such as } m \text{ Pre}_\text{local} m'' \text{ and } m'' \text{ Pre}_\text{local} m', \text{ To}(m'') = \text{To}(m) \}.
\]

The third clause allows to consider in \( \langle \text{causal, re} \rangle \) only pairs of actions that are directly consecutive with regard to this relation: \( (?m, !m') \in \langle \text{causal, re} \rangle \) implies there is no send action between \( ?m \) and \( !m' \).

**Emission-Emission relationship.** This rule keeps the idea that if a message \( m \) precedes another message \( m' \), then \( m \) must be created and sent before \( m' \) in as much as this does not prevent the realizability of the action ordering. This principle entails no realizability issue in the case where \( m \) and \( m' \) are sent by the same object. This rule is illustrated in Fig. 6.a, the object \( o \) sends the message \( m_4 \) after having sent the message \( m_3 \). Formally:

\[
\langle \text{causal, ee} \rangle = \{ (!m, !m') \mid m, m' \in M, \text{ From}(m) = \text{From}(m') \text{ and } m \text{ Pre}_\text{local} m' \}.
\]

Now, the causal-semantics is defined as the transitive closure of the union of these relations

\[
\langle \text{causal} \rangle = (\langle \text{causal, ee} \rangle \cup \langle \text{causal, re} \rangle \cup \langle \text{sync} \rangle^*)^*.
\]

It is clear that \( \langle \text{causal, ee} \rangle \cup \langle \text{causal, re} \rangle \subseteq \langle \text{MSC, local} \rangle^* \), so that we have the following:

**Theorem 5:** For any SD, we have causal-semantics \( \subseteq \) MSC-semantics.

### 8. Relationships between the semantics

In the previous section, we have already shown the following inclusion between the four semantics:

causal-semantics \( \subseteq \) MSC-semantics \( \subseteq \) linear-semantics \( \subseteq \) emission-semantics

Now we wonder in which cases some of these semantics are *equivalent*, that is they include the same ordering constraints on the set \( A \) of all the message sending and receiving actions.

The first result is about the linear semantics. It is a very specific and easy to characterize semantics since it defines a total (that is maximal) order relation on \( A \). This semantics reveals to be closely related to a structural property of SD, sequentiality: it is the condition for realizability (cf. Theorem 1), and also for equivalence with other semantics.

**Theorem 6:** If a SD is sequential, then its four semantics are equivalent. Conversely, if the linear-semantics of a SD is equivalent to one of its emission, causal or MSC-semantics, then it is a sequential SD.

**Proof.** Let us consider a sequential SD that is such that for any \( m, m' \in M, m \text{ Pre } m' \) implies \( \text{To}(m) = \text{From}(m') \). The emission semantics is a total order, and thus equivalent to the linear one, since for any \( m, m' \in M \) such that \( m \text{ Pre } m' \), we have
\(!m <_{\text{emission}} ?m <_{\text{emission}} !m'\). The causal- and MSC-semantics are also equivalent since no object performs two consecutive reception actions, then each sending action is the local consequence of the immediate preceding action; and if a sending action is locally followed by a reception one, then there is a causal thread from the sending to the reception since the SD is sequential. Thus the causal-semantics defines a total order on each object, as the MSC-semantics does. Finally, the MSC-semantics is a total order, at any step there is only one action to perform, and thus it is equivalent to the linear one.

Conversely, let us consider a SD having equivalent emission and linear semantics. So we have \(?m <_{\text{emission}} !m\) for any \(m, m' \in M\) such that \(m \text{ Pre } m'\), and thus \(\text{To}(m) = \text{From}(m')\), that is the SD is sequential. With regard to the causal and MSC-semantics, suppose that SD is not sequential and includes two messages \(m\) and \(m'\) such that \(m \text{ Pre } m'\), \(\text{To}(m) \neq \text{To}(m')\) and \(\text{To}(m) \neq \text{From}(m')\); according to both the causal and MSC-semantics, there is a step where the \(?m\) and \(!m'\) actions may be executed concurrently; thus neither the causal nor the MSC-semantics is a total order strictly contained in the linear-semantics.

Now we consider the relations between the causal and MSC semantics, and we will show that there is a structural condition for their equivalence.

A SD is said to be **locally causal** if it satisfies the two following conditions:

1. for any messages \(m\) and \(m'\) such that \(m \text{ Pre}_{\text{local}} m'\), \(\text{To}(m) \neq \text{To}(m')\), that is an object never performs two consecutive reception actions;
2. for any messages \(m\) and \(m'\) such that \(m \text{ Pre}_{\text{local}} m'\) and \(\text{From}(m) = \text{To}(m')\), there exists a sequence of messages \(m = m_1 \ldots m_k = m'\) such that \(m_i \text{ Pre } m_{i+1}\) and either \(\text{From}(m_i) = \text{To}(m_{i+1})\) or \(\text{To}(m_i) = \text{From}(m_{i+1})\), that is if an object receives \(m'\) after sending \(m\), then \(!m <_{\text{causal}} ?m'.\)

**Theorem 7**: A SD is locally causal iff its causal and MSC-semantics are equivalent.

**proof.** Examining the definition of the causal-semantics, it is easy to verify that the two conditions of local causality are the ones that provide each object with a total ordering of its actions, that is the same ordering as the MSC-semantics. Conversely, if the set of actions executed by each object is totally ordered by the causal-semantics, there is no successive reception actions in an object (clause 1 is satisfied), and each reception is caused by its immediate local sending, necessarily by a causality thread from the sending to the reception – clause 2 is also satisfied.

Concerning the emission and causal-semantics, the following theorem together with its corollary shows that they become comparable thanks to the local control and local causality properties.

**Theorem 8**: A SD is locally controlled iff its emission-semantics is included in its causal-semantics:

\(<_{\text{emission}} \subseteq <_{\text{causal}}\).

**proof.** If a SD is not locally controlled, its emission-semantics is not realizable (theorem 2) and thus includes inter-objects ordering constraints that do not belong to the causal-semantics since this later is realizable in any case. Now consider a locally controlled SD. Both semantics are realizable and thus have the same inter-objects ordering constraints. With regard to local constraints, if \(?m <_{\text{emission, re}} !m'\) then \(m \text{ Pre } m'\) and \(\text{To}(m) = \text{From}(m')\) and thus \(?m <_{\text{causal, re}} !m'\); so \(<_{\text{emission, re}} \subseteq <_{\text{causal, re}}\).
= From(m') since SD is locally controlled and ′m ≤<causal, ee ′m'; the converse holds so that
<emission, ee = ≤<causal, ee. Thus we have <emission ⊆ ≤<causal. ■

Corollary: If a SD is locally controlled and locally causal, then its emission and causal semantics are equivalent: <emission = ≤<causal.

proof. According to theorem 8, we just have to prove that <emission, re = ≤<causal, re, which results from the fact that locally causal objects newer perform two consecutive receptions. The converse does not holds; if ≤<emission = ≤<causal then the SD is locally controlled according to theorem 8, but it is not necessarily locally caused: as a counter example, removing the message m4 of the SD in Fig. 7 gives a not locally caused SD that satisfies ≤<emission = ≤<causal. ■

- Fig. 7. A locally controlled SD and its four different semantics.

Both the emission and causal semantics include no ordering constraints between reception actions, the emission semantics because it deals mainly with emission actions, the causal semantics because there is no direct local cause to the reception of a message. However, it seems to be reasonable to consider the constraint depicted in Fig. 6.b: if two messages m and m′ are such that m Pre local m′, To(m) = To(m′) and From(m) = From(m′), then the constraint ?m < ?m′ may be view as a direct consequence on the addressee side of the constraint ′m < ′m′ that holds on the sender side.

This leads to define the relations

<rr = {(?m, ?m′); m Pre local m′, From(m) = From(m′) and To(m) = To(m′)}

and the two following semantics that are variants of the emission and causal ones:

≤<emission = (≤<emission ∪<rr)° and ≤<causal = (≤<causal ∪<rr)°.

On the other hand, accounting for the <rr relation does not extend the linear and MSC-semantics that already include the constraints of this relation.

All the results concerning the emission-semantics or the causal-semantics still hold if they are defined as the ≤<emission or ≤<causal Relations instead of the ≤<emission and ≤<causal ones: theorems 2, 3, 5, 6 and 8 remain true, and also the corollary (even with a slightly weaker definition of clause 1 of the local causality).
9. Conclusion

Sequence diagrams, as they are defined in the 1.x versions of UML, are ambiguous in nature. On the one hand, the implicit UML semantics based on the global ordering of messages suggests the emission-semantics (or even the linear-semantics) that suits well the modelling of interactive applications; on the other hand, this semantics is not realizable for most Sequence Diagrams, and it is not coherent with the intuitive reading of Sequence Diagrams like in MSC as a set of processes that synchronize their respective executions by sending messages.

The aim of this paper is to highlight this ambiguity by disclosing different operational semantics that make explicit possible interpretations of SD and to compare these semantics. This semantics are defined using the same theoretical framework than the UML 2.0 semantics, that is the partial order theory [18].

Comparisons between these semantics leads to identify two structural properties of SD that also correspond to the fulfillment of some behavioural properties. The first one is the locality of control – any message is sent by the object that has also sent or received the previous message –, and the second one is the locality of causality – any receive action in an object is preceded by a send action that is the cause of this reception.

Concerning the realizability, the causal- and MSC-semantics are always realizable since the only inter-objects synchronizations are the ones performed by the messages exchanges. On the other hand, the realizability of the linear- and emission-semantics required the sequential and the locally control properties of SD.

References