Extended Message Sequence Charts With Time-Interval Semantics

P. S. Muniz Silva
Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, 158, tv. 3 05508-900 São Paulo SP, Brazil
E-mail: psmuniz@pcs.usp.br

Abstract

The paper describes the main ideas of an extension of Message Sequence Charts (MSCs) in a time-interval structure to improve the temporal knowledge of the environment where a software system will be built. The extended MSC could be used at the requirements analysis phase and its interpretation is not limited to the analysis of real-time software systems (where MSCs are mostly used), but rather is aimed for helping the qualitative temporal analysis of the application domain, whatever the software solution for the system. The paper briefly overviews MSCs, describes the goals of the proposed interpretation for ***basic*** *MSCs, and defines their syntax and semantics.*

1. Introduction

Every software developer agrees that requirements analysis usually faces great difficulties to produce a defined model which will be taken as a prescription for the rest of the software development cycle. It is now common sense the notion that the sooner we catch faults in the cycle, the more reliable the resulting software and the lower the costs of the project will be. But the concept of faults has different meanings across the development cycle. At analysis stage, faults arise mostly from misunderstandings in the knowledge of the domain. Despite the fact that the understanding of the domain is incremental over the cycle (sometimes the acceptable picture happens too late!), a part of the domain information already available at requirements analysis is not adequately checked. Problems in checking stem from the fact that traditional verification and validation tools have trouble in dealing with the partial information available at the requirements analysis step [13]. One of these pieces of information is time.

Mainly, time analysis is only taken into consideration when the devised solution for a software system is a real-time architecture, but time insights are often neglected in the absence of real-time constraints. Sometimes time is consciously discarded to avoid the complexities derived from the introduction of their assumptions and where appropriate tools are not available. Sometimes it is unconsciously not considered. This attitude may cause troubles in the future system. One tool that helps in the analysis of temporal issues and in the detection of some classes of faults related to temporal issues is the Message Sequence Chart (MSC).

MSCs have been used in the development of telecommunications and reactive systems and adopted within a wide range of software engineering methods [7]. In telecommunications, the International Telecommunication Union (ITU) defined an MSC standard in Recommendation Z.120 [14]. In software engineering, they are used at different levels of abstraction: as a requirements analysis tool [24] [17] [10] [9] [5]; as a design tool [16] [23] [8]; as a test tool [11]. As a requirements analysis tool, in general, MSCs are used to represent sequences of events between the main abstractions within the system and between the system and its environment. In the latter case, the description of flows of events (or stimuli) focuses on the interaction between the system and its users. While this is a sound and practical viewpoint (after all, a system has to be built), committing to a system-centered view at the very earlier stages of the analysis could obfuscate some environment angles.

Recent research in the field of requirements engineering has provided more rigorous presentations of the notion that the vocabulary of requirements is pinned to the environment. Zave and Jackson [26] demonstrate that the "terminology used in requirements engineering should be grounded in the reality of the environment" for which a computer-based artifact (a 'machine') is to be built, and that it is not necessary to (abstractly) describe the machine. What is under description at requirements analysis is the environment, "as it would be without or in spite of the machine and as we hope it will become because of the machine." The environment is the *locus* that provides the meaning of primitive terms and relations, which constitute the requirements. Nevertheless, these terms and relations have informal explanations made of (possibly) clear sentences with verbs and their tenses.

For the sake of brevity, we will not develop further consequences from these observations, but note that as long as we abstract a fixed and suitable “present tense”, time may not require special attention [18]. However, when we reason about the possible course of events of interest in the whole environment where a machine will be built, our explanations are sentences in past, present and future tenses. Moreover, the notion of truth when reasoning about relations between explanations relates sentences not to instants of time but to intervals of time. The structure of time underlying the reality of the environment is actually an interval structure.

Our goal is to use an MSC as a tool in the earlier stages of requirements analysis to help the reasoning about the course of relevant events in the environment where a machine will be built. For such, we need an extended MSC with a temporal semantics to support the universe of discourse of the environment with its time-interval structure. Our proposal relies upon Allen’s time-interval theory [1] [2]. Allen’s theory was developed in the context of the so-called ‘Naive Physics’ with its simple ‘common sense’ representations [3], and from the perspective of artificial intelligence, describing a temporal representation that takes the notion of temporal interval as a primitive. This framework adequately meets the above requirements for the temporal structure. We impose an additional constraint for the MSC representation: our proposal should remain as close to the Z.120 recommendation as possible, and should maintain the intuitiveness of MSCs.

In section 2, we summarize some definitions of MSCs, examine some methodological issues concerning current trends in the analysis of MSCs, and outline our proposal for the analysis. In section 3, we present a brief syntactic description of the extended MSC. In section 4, we present an interpretation of the extended MSC in a time-interval structure, where relations between intervals are Allen’s relations. In section 5, we summarize the main ideas and provide a brief overview of our current research and planned future works.

2. Methodological Issues

2.1 Message Sequence Charts

We extensively quote [19] [4] [7] in this overview of MSCs. MSCs are a graphical representation which shows message exchanges between concurrent processes in a system. Figure 1 shows a *basic* MSC (bMSC). Each vertical line has a start and end symbol and represents processes or autonomous agents (*P1*, *P2*, and *P3*). Each horizontal arrow describes a message sent from one process to another (*a*, *b*, and *c*). The tail of an arrow corresponds to the event of sending a message, while the head corresponds to its receipt. Communication is one-to-one and asynchronous, and control flows independently within each process from the start symbol to the end symbol. In each process, the events are temporally ordered from top to bottom. The system terminates when all processes have terminated.

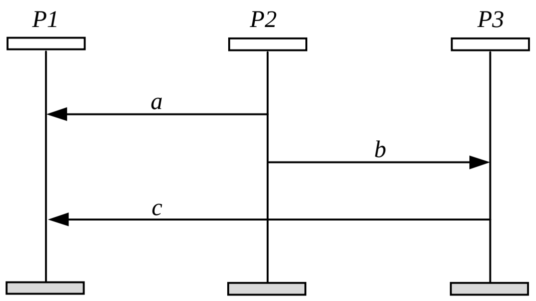

Figure 1

The behavior of a bMSC is the set of sequences of sent and received messages. For each message in a bMSC, its sending event is ordered before its receiving event. Within each process, there is a *local total order*, where the events are totally ordered according to their position in the vertical line. A *visual order* [4] defines an acyclic digraph over the events, since sending and receiving edges cannot go upwards in the chart. Normally, the overall events in a bMSC are partially ordered.

The recommendation Z.120 also defines *high-level MSCs* (hMSCs), composed by bMSCs, to describe systems having recursive and non-deterministic behavior. An hMSC is a digraph where nodes denote bMSCs and edges denote possible continuations of bMSCs by others. An hMSC may present iteration and branching. In this paper we will not discuss hMSCs. They are described in the proposed framework in [21].

2.2 Current Trends in the Analysis of Message Sequence Charts

The formalization of Message Sequence Charts has gained a significant concentration of research efforts, mainly fostered by recommendation Z.120 [14]. Among the proposed approaches, we quote the process algebra [15] [20], and varieties of a kind of model checking [19] [4] [7], to mention a few. Our interest is centered on this latter approach. We present below a very brief summary of the principal arguments developed in [4] and [7] to support analysis issues of MSCs.

Alur et al. [4] formalize timed and untimed bMSCs, i.e., bMSCs with and without timing constraints, respectively. An untimed bMSC exhibits a *visual order* for the events to occur, which does not necessarily correspond to the actual semantics of the bMSC: in some scenarios the events may occur in an order that differs from the visual one. The semantics of a bMSC depends on the causal precedence enforced by the underlying architecture (*enforced order*), and on the causal precedence inferred by the analyst (*inferred order*). These orderings should not be discrepant. Such considerations introduce a system design bias in the analysis process, which constrains the orderings: the system underlying architecture and the queueing disciplines used. The timed bMSC specify timing constraints on delays of messages delivery and between events that lie in processes. Furthermore, it incorporates a semantics for Z.120 *timers*, having their set, reset and time-out events. A timed bMSC has timing functions that map each event pair from the enforced and the inferred order to a time interval, defined as intervals of the set of nonnegative real numbers with integer endpoints. The analysis of timed bMSCs tries to detect timing inconsistencies (there exists no satisfiable set of timing assignments), visual conflicts (timing constraints imply that two events appear in an order opposite to their visual order), and timing conflicts (the inferred interval bounds are not necessarily satisfied by the timing assignments).

Ben-Abdallah and Leue [7] assume the timing assignments extensions, but note that all of them only address bMSCs. They analyze the syntactic features, expressiveness and limitations of that extensions from the hMSC viewpoint. They propose to consolidate these extensions and extend the timing consistency analysis to deal with iterating and branching hMSCs. They also address the so-called process divergence problem in hMSCs: in the presence of iteration a process sends messages a number of times ahead of the messages being received. In hMSCs the timing constraints can spread across sequentially connected bMSCs, e.g. a timer may be set in a bMSC and reset in a subsequent bMSC. The interpretation of iteration raises some important questions, such as: how to interpret multiple occurrences of the set event of the same timer? how to settle the correspondence between several set and time-out events of timers? The interpretation of branching requires the adoption of an appropriate semantics: *local* (bMSCs shared by different paths in an hMSC graph may have different temporal behavior, depending on the past and future behavior of the system), or *global* (bMSCs shared by different paths have the same temporal behavior, independently of the path where it resides). The authors formalize an *MSC specification* (basically an hMSC structure), augmenting the timing analysis proposed by [4] to cope with a scenario where a timer is set in a bMSC, but is not followed by a reset or time-out event in the same bMSC, and provide an extension to handle branchings and iterations. The determination of timing consistency is similar to [4].

2.3 The Proposed Analysis of Message Sequence Charts

Our proposal for the analysis of MSCs takes the [4] and [7] approaches for the starting point. As a convenience, we classify their analyses as *quantitative analyses*, due to timing assignments. These approaches are very appropriate for the analysis of real-time systems, where timing constraints are under the permanent focus of the analyst. But when faced with an application domain that does not demand real-time analysis, usually the analyst forgets about the temporal reasoning. Our goal is to encourage the requirements analyst to carry out the investigation of the requirements from the temporal viewpoint using the extended MSC as a *qualitative temporal analysis tool*, which does not require any timing assignment and is not bound to any underlying architecture of the future machine. MSCs are very intuitive and effective to capture sequences of events occurring in the requirements domain and, if supported by a rigorous semantics, they can compose a useful framework to validate the temporal structure of the requirements description.

Our proposal extends MSCs to allow their interpretation in a *time-interval structure*, more specifically the Allen's linear time-interval semantics. For bMSCs, will we call their extension a *Basic Time-Interval MSC* (bTIMSC). bTIMSCs could be used, for example, to detect, in a pure qualitative way, the potential conflicts in the desired visual order expressed in the produced MSCs. It is a common situation in non-real-time projects, customers and analysts take the intended visual orders depicted in requirements scenarios, described by MSCs, as *the* actual and unambiguous course of events. Unfortunately, subtle conflicts eventually emerge at customers' sites. bTIMSCs could also guide the first trials of timing assignments for a subsequent quantitative analysis made by the aforementioned tools. In this paper we will neither consider bTIMSCs with the Z.120 timer extension, nor the *High Level Time Interval MSC* (hTIMSC), an extension of hMSCs. They are described in [21]. Our basic methodological approach to bTIMSCs is:

1. Produce bMSCs for identified scenarios from the environment, using any adequate software engineering method for the description of scenarios. The bMSCs result from the analyst's effort.
2. Interpret the bMSCs as bTIMSCs (cf. section 3), focusing the temporal properties (the universe of discourse) of the environment. The bTIMSCs are a product of the analyst's effort.
3. Translate the bTIMSCs into extended *Basic Message Flow Graphs - bMFGs*, which formalize the charts

produced (section 3). An automatic tool may carry out this step.

4. Translate the extended bMFGs into *Interval Calculus Graphs - ICGs*, which formalize the semantics for the bTIMSCs (section 4). An automatic tool may carry out this step.
5. Analyze the resulting ICGs, by applying constraint propagation analysis methods (section 4). This step is carried out in a semi-automatic way, with the analyst interacting with a tool.

3. Syntax of the Basic Time-Interval Message Sequence Charts

Initially, we will extend the bMSC with the notion of *focus of interest*. They are intervals in which a process (e.g. an actor in an application domain) perceives the intervals where messages occur. They bring the mutual temporal position of messages into the analyst's focus. He/she can focus on interesting temporal positions of messages relative to processes. We borrow the representation of *focus of control*, an adornment in the Sequence Diagram of the UML notation [22], to represent the focus of interest. For example, the bMSC depicted in figure 1 could be drawn as in figure 2, where *f1*, *f2*, and *f3* denote focuses of interest. Usually, focuses of interest such as *f1*, *f2*, and *f3*, are of primary concern to the analyst. They allow him to reason about the essential semantics of the bMSC. As a convenience, we call them *main focuses of interest.*

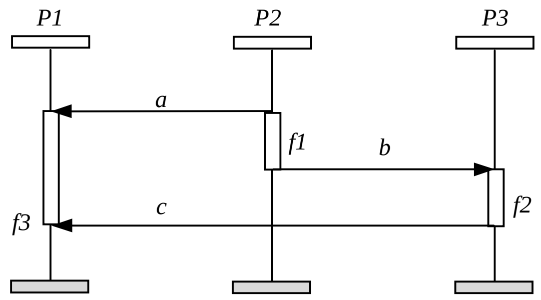

Figure 2

To describe a bMSC, we use the *Message Flow Graph* (MFG) [19]. The MFG provides a rigorous description of MSCs, and the *basic* MFG (bMFG), i.e., a MFG without branching or iteration, adequately describes a bMSC. In this paper, we do not consider timers and only extend the bMFG syntax to represent the notion of focus of interest. For the semantics, we do not obtain a finite-state automaton derived from a possibly extended Global State Transition Graph, to define an interpretation of the MFG (and bMFG), as [19] do. Rather, we take another direction to interpret a bMSC in the Allen's time-interval structure, by associating time intervals with messages and with focuses of interest.

The following bMFG in figure 3 is derived from the bMSC of figure 2. There, the nodes *!a*, *!b*, and *!c* denote *sending* events; *?a*, *?b*, and *?c* denote *receiving* events; *Top* denotes the *top* nodes of process *P1*, *P2*, *P3*; *Bottom* denotes the *bottom* nodes of process *P1*, *P2*, *P3*; *a*, *b*, and *c* denote *messages*; and *f1*, *f2*, and *f3* denote *focuses of interest*.

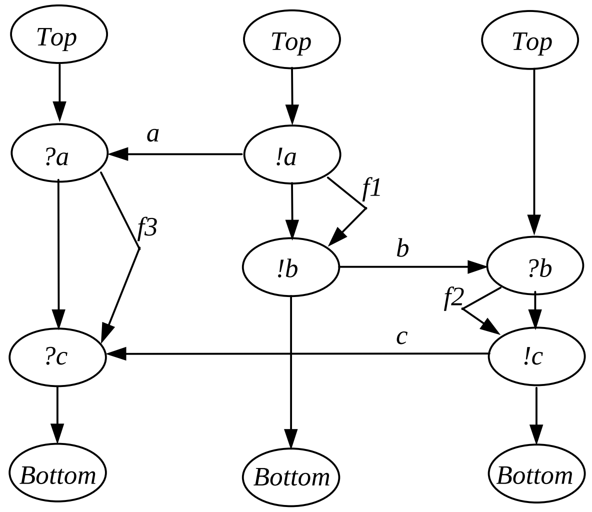

Figure 3

The syntax of the extended bMFG is an extension of the Ladkin and Leue's syntax description [19]. We extensively transcribe Ladkin and Leue's definitions and incorporate our extension, while preserving their style. We do not provide the complete formalization, for the sake of space. The complete formalization (timers included) appears in [21]. In this paper the only extension added to Ladkin and Leue's syntax description is the focus of interest.

Let *S*, *C*, and *X* denote arbitrary pairwise disjoint sets, their elements are called *sending* events, *receiving* events, and *extra* nodes (*top* and *bottom* nodes), respectively. Let *ST*, *ET*, and *FT*, denote arbitrary disjoint sets (also disjoint from *S*, *C*, and *X*). Their elements are called *signal* type, *event* type, and *focus* type, respectively. The bMFG is the tuple:

$$G = (S, C, X, sig, ne, foi, stype, ST, etype, ET, ftype, FT, Top, Bottom)$$

where $(S \cup C \cup X, ne, etype, ET)$ is a digraph with node labels. *G* satisfies the following conditions:

- $sig \subseteq S \times C$ is a bijective relation, named *signal* edge. Each process node is connected by precisely one *signal* edge to a unique node in another process.
- $ne \subseteq V \times V$ is a bijective relation, where $V = S \cup C \cup X$. *ne* is called the *next event* edge. Each

process node is connected by precisely one *next event* edge to a unique node in the same process.

- $foi \subseteq V \times V$ is a bijective relation, where $V = S \cup C \cup X$. *foi* is called the *focus of interest* edge.
- The set $ET = (\{!, ?\} \times ST) \cup \{Top, Bottom\}$ contains the *event* types. We write $!t$ for $(!, t)$, and $?t$ for $(?, t)$.
- *stype*: $S \times C \rightarrow ST$. *ST* is a label of a *signal* edge.
- *etype*: $V \rightarrow ET$ where $V = S \cup C \cup X$.
- *ftype*: $V \times V \rightarrow FT$, where $V = S \cup C \cup X$, and *FT* is a label of a *focus of interest* edge.
- If the type of a signal is *m*, then the corresponding sending and receiving events are of type $!m$ and $?m$, respectively:
  $(a, b) \in sig$ **then**
  $(\exists m \in ST)\ (stype(a, b) = m \wedge$
  $(etype(a) = !m \wedge etype(b) = ?m))$.
- If the type of a focus of interest is *f*, then the corresponding nodes are:
  $(a, b) \in foi$ **then**
  $(\exists f \in FT)\ (ftype(a, b) = f \wedge a, b \in ET)$.

Every component of the digraph with node labels contains one starting event at most. The definitions of *start* and *finish* nodes, of type *Top* and *Bottom*, respectively, and the *process* type are the same as in [19]. Other conditions state that a bMFG does not have branching or cycles in the *ne* relation, that all elements in some component are reachable from the start node, and for any signal type there is a unique sender and a unique receiver process [19].

4. Semantics of the Basic Time-Interval Message Sequence Charts

The semantics is formalized by a network representation we call *Interval Calculus Graph* (ICG), inspired in the Interval Algebra network [6]. The nodes in the ICG directed graph are time intervals and the edges the Allen's relations. We can translate a bMFG into an ICG. We call *communication interval* an interval associated with a message, and *focus interval* an interval associated with a focus of interest. The relations between two intervals follow the Allen's framework, with its 13 basic relations [1]. Let ***I*** be the set of all basic mutually exclusive relations {***b***, ***bi***, ***m***, ***mi***, ***o***, ***oi***, ***s***, ***si***, ***d***, ***di***, ***f***, ***fi***, ***eq***}, where ***b*** stands for *before*, ***bi*** for *after*, ***m*** for *meets*, ***mi*** for *met by*, ***o*** for *overlaps*, ***oi*** for *overlapped by*, ***s*** for *starts*, ***si*** for *started by*, ***d*** for *during*, ***di*** for *contains*, ***f*** for *finishes*, ***fi*** for *finished by*, and ***eq*** for *equals*. The relation between two intervals is any subset of ***I***. For each basic relation there exists a relation inverse to it, i.e., if a relation *r* holds between intervals *A* and *B*, then its inverse relation **inv**(*r*), denoted by *ri*, holds between *B* and *A* [12]. For example, $A\ \boldsymbol{b}\ B \Leftrightarrow B\ \boldsymbol{bi}\ A$. As the primary concern of MSCs are the messages exchanged between processes, we state the following precedence rule for the default direction when translating an MSC into an ICG directed graph: from a communication interval to a focus interval. We may have focuses of interest embedded in a bMSC, depending on the goals of the analysis. In this particular case, there are also interval relations between the focuses, with no precedence rule.

Let *C* and *F* denote arbitrary disjoint sets. Their elements are called *communication* intervals and *focus* intervals, respectively. Let *VT* and *BT* denote arbitrary disjoint sets (also disjoint from *C* and *F*). Their elements are called *interval* type and *basic relation* type, respectively. The ICG is the tuple:

$G_{IC} = (C, F, r, ri, vtype, VT, btype, BT)$

The G_{IC} satisfies the following conditions:

- The set $VT = ST \cup FT$, contains the *interval types*, where *ST* and *FT* were defined in the bMFG description, i.e., they are labels for *messages* and *focuses of interest* respectively.
- The set *BT* = {***b***, ***bi***, ***m***, ***mi***, ***o***, ***oi***, ***s***, ***si***, ***d***, ***di***, ***f***, ***fi***, ***eq***}, contains the *basic relation* types from the Allen's interval algebra.
- *vtype*: $C \cup F \rightarrow VT$.
- *btype:* $C \times F \rightarrow BT$ **or** *btype:* $F \times F \rightarrow BT$
  or *btype:* $F \times C \rightarrow BT$
- $r \subseteq V \times V$ is a bijective relation, where $V = C \cup F$. *r* is called the *interval relation*.
- If the type of an *interval relation* is *t*, then its corresponding domain and range nodes are of type *communication interval* and *focus interval*, respectively, or both are of type *focus interval*.
  $(a, b) \in r$ **then**
  $(\exists t \in BT)\ (btype(a, b) = t \wedge$
  $((vtype(a) = C \wedge vtype(b) = F) \vee$
  $(vtype(a) = F \wedge vtype(b) = F)))$.
- $ri \subseteq V \times V$ is a bijective relation, where $V = C \cup F$. *ri* is called the *inverse interval relation*.
- If the type of an *inverse interval relation* is *t*, then its corresponding domain and range nodes are of type *focus interval* and *communication interval*, respectively, or both are of type *focus interval*.
  $(a, b) \in ri$ **then**
  $(\exists t \in BT)\ (btype(a, b) = t \wedge$
  $((vtype(a) = F \wedge vtype(b) = C) \vee$
  $(vtype(a) = F \wedge vtype(b) = F)))$.
- If there is an *interval relation* between two nodes with a particular set of basic relations, its *inverse interval relation* is the set whose elements are the inverse of each element from the former set. Let *t* be a basic relation. We denote by *ti* its inverse basic relation, as defined in [1]:

$(a, b) \in r$ **then**
$(\exists t \in BT)\ (btype(a, b) = t \wedge btype(b, a) = ti)$.
The inverse interval relation is useful for the composition operation in Allen's algebra.

We define a mapping between the elements of the bMFG and the elements of the ICG as:

$$ICG_{MAP} = \{i \mapsto C, j \mapsto F \mid i \in \{sig\} \wedge j \in \{foi\}\},$$

where **domain**$(ICG_{MAP}) \subset G$ and **range**$(ICG_{MAP}) \subset G_{IC.}$

We informally describe the translation of a bMFG into an ICG. The translation imposes an ordering in the mapping, which follows the *visual order* of the bMFG [4]. Every process in the bMFG has a *local total order* $<_p$ over the events $V = S \cup C \cup X$, which corresponds to the order in which the events appear in the bMFG. The *visual order relation* is the relation:

$$<_v \equiv (\cup_p <_p) \cup foi \cup sig.$$

The relation contains all the local total orders and all the edges. We define an *initial signal edge*, formed by any first sending and receiving events that follows the *Top* events in the processes of the bMFG. More formally, let V_0 denote the set of these events. Then,

$$V_0 = \{(a, b) \in ne \mid (ne \rhd \{a\}) = 0\},$$

where $\rhd$ is the range restriction operator of the Z language [25]. The initial signal edge is any signal edge chosen from the set:

$$\{(a, b) \mid a, b \in \mathbf{range}(V_0) \wedge (a, b) \in sig\}.$$

This initial signal edge maps into an *initial communication interval*. Next, we search for a focus of interest which relates the sending or receiving event of the initial signal edge to the sending or receiving events of the next signal edge in the visual order. For convenience, we call this signal edge the second signal edge. That focus of interest maps into a focus interval in the ICG. We relate the initial communication interval to that focus interval with an appropriate interval relation. We also relate the second signal edge to that focus interval. We continue the mapping procedure following the signal edges and focuses of interest in the visual order of the bMFG. A more rigorous and thorough description of the mapping procedure can be found in [21].

We define a collection of basic rules for the default translation of the interval relation between a communication interval and a focus interval. These rules express the basic configurations between one message and one focus of interest in a bMSC. Let $<$ denote the precedence operator over the events $V = S \cup C \cup X$. Let PT denote an arbitrary disjoint set whose elements are names of processes. We can define a process type as $ptype: V \rightarrow PT$.

In a G graph, we have the following default translation rules:

1. Let $v_1 < v_2 < v_3$ and $v_1, v_3 \in V$ and $v_2 \in C$, where $ptype(v_1) = ptype(v_2) = ptype(v_3) = P1$. Let $v_4 \in S$, where $ptype(v_4) = P2$. Let also $(v_1, v_2), (v_2, v_3) \in ne$, $(v_1, v_3) \in foi$ and $(v_4, v_2) \in sig$.
Let $(v_4, v_2) \mapsto a$, $(v_1, v_3) \mapsto b$ be nodes of the resulting G_{IC} from G. If $(a, b) \in r$ **then** $(\exists t \in BT)$
$(btype(a, b) = t \wedge t = \{\boldsymbol{d}, \boldsymbol{s}, \boldsymbol{o}\})$.

2. Let $v_1 < v_2 < v_3$ and $v_1, v_3 \in V$ and $v_2 \in S$, where $ptype(v_1) = ptype(v_2) = ptype(v_3) = P1$. Let $v_4 \in C$, where $ptype(v_4) = P2$. Let also $(v_1, v_2), (v_2, v_3) \in ne$, $(v_1, v_3) \in foi$ and $(v_2, v_4) \in sig$.
Let $(v_2, v_4) \mapsto a$, $(v_1, v_3) \mapsto b$ be nodes of the resulting G_{IC} from G. If $(a, b) \in r$ **then** $(\exists t \in BT)$
$(btype(a, b) = t \wedge t = \{\boldsymbol{d}, \boldsymbol{f}, \boldsymbol{oi}\})$.

3. Let $v_1 < v_2$ and $v_1 \in C$ and $v_2 \in V$, where $ptype(v_1) = ptype(v_2) = P1$. Let $v_3 \in S$, where $ptype(v_3) = P2$. Let also $(v_1, v_2) \in ne$, $(v_1, v_2) \in foi$ and $(v_3, v_1) \in sig$. Let $(v_3, v_1) \mapsto a$, $(v_1, v_2) \mapsto b$ be nodes of the resulting G_{IC} from G. If $(a, b) \in r$ **then** $(\exists t \in BT)$
$(btype(a, b) = t \wedge t = \{\boldsymbol{m}\})$.

4. Let $v_1 < v_2$ and $v_1 \in S$ and $v_2 \in V$, where $ptype(v_1) = ptype(v_2) = P1$. Let $v_3 \in C$, where $ptype(v_3) = P2$. Let also $(v_1, v_2) \in ne$, $(v_1, v_2) \in foi$ and $(v_1, v_3) \in sig$. Let $(v_1, v_3) \mapsto a$, $(v_1, v_2) \mapsto b$ be nodes of the resulting G_{IC} from G. If $(a, b) \in r$ **then** $(\exists t \in BT)$
$(btype(a, b) = t \wedge t = \{\boldsymbol{s}, \boldsymbol{eq}, \boldsymbol{si}\})$.

5. Let $v_1 < v_2$ and $v_1 \in V$ and $v_2 \in C$, where $ptype(v_1) = ptype(v_2) = P1$. Let $v_3 \in S$, where $ptype(v_3) = P2$. Let also $(v_1, v_2) \in ne$, $(v_1, v_2) \in foi$ and $(v_3, v_2) \in sig$. Let $(v_3, v_2) \mapsto a$, $(v_1, v_2) \mapsto b$ be nodes of the resulting G_{IC} from G. If $(a, b) \in r$ **then** $(\exists t \in BT)$
$(btype(a, b) = t \wedge t = \{\boldsymbol{fi}, \boldsymbol{eq}, \boldsymbol{f}\})$.

6. Let $v_1 < v_2$ and $v_1 \in V$ and $v_2 \in S$, where $ptype(v_1) = ptype(v_2) = P1$. Let $v_3 \in C$, where $ptype(v_3) = P2$. Let also $(v_1, v_2) \in ne$, $(v_1, v_2) \in foi$ and $(v_2, v_3) \in sig$. Let $(v_2, v_3) \mapsto a$, $(v_1, v_2) \mapsto b$ be nodes of the resulting G_{IC} from G. If $(a, b) \in r$ **then** $(\exists t \in BT)$
$(btype(a, b) = t \wedge t = \{\boldsymbol{mi}\})$.

For example, the default translation of the very modest bMFG of figure 3 into an ICG can be depicted in figure 4. One could reason about the temporal properties of the resulting ICG by applying *constraint propagation analysis* methods for relations between intervals, used to infer the relative mutual positions of the intervals in a purely qualitative manner [1] [6] [12]. In this paper we will not discuss neither the properties of the resulting ICG that, for example, could make it particularly tractable, nor the results of its application in actual temporal requirements analyses. As an example of a simple analysis, by applying Allen's composition and intersection operations, let us examine the bMFG of figure 3. Its visual order shows that the event *?a* precedes the event *?c*, or *?a* < *?c*. To check this assumption, we analyze the relation between the communication intervals *a* and *c* in the ICG of figure 4.

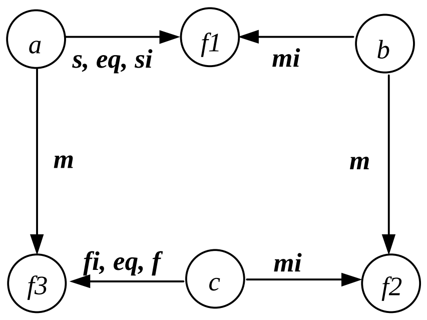

Figure 4

What happens if we do not impose any constraints over the events in process P1? By applying composition over the interval relations (*a*, *f1*), (*b*, *f1*), (*b*, *f2*), and (*c*, *f2*) (taking the inverse interval relations (*f1*, *b*) and (*f2*, *c*)), we obtain the disjunction of basic relations {***b***, ***m***, ***o***, ***fi***, ***di***} between intervals *a* and *c*. The interval relation ***di*** says that *?c* < *?a*, which contradicts the visual order. The precedence relation *?a* < *?c* might hold if we could guarantee the constraints imposed by relations (*a*, *f3*) and (*c*, *f3*). In fact, by applying composition over the interval relation (*a*, *f3*) and the inverse interval relation (*f3*, *c*), we obtain the disjunction of basic relations {***b***, ***m***, ***o***, ***s***, ***d***} between intervals *a* and *c*. By applying intersection we obtain the disjunction of basic relations {***b***, ***m***, ***o***}, which guarantees *?a* < *?c*. By examining the discarded interval relations ***s*** and ***d***, we note that if we consider the relation ***s*** unlikely to occur (*!a* is simultaneous with *!c*), then the relation ***d*** is unrealistic (*!c* < *!a*). Therefore, we could take {***b***, ***m***, ***o***} for the feasible basic relations that are consistent with the intended visual order.

This kind of reasoning is interesting as a first approach to the analysis of the temporal properties of an environment (partially) described by a bMSC, improving the knowledge about the environment. It may be used to validate, at the requirements abstraction level, the constraints to be considered in the specification of the machine, which will implement the desired requirements.

5. Conclusions

The proposed interpretation of *basic* MSCs in this paper integrates two mainstreams from software engineering and artificial intelligence: requirements analysis based on MSCs and temporal reasoning based on Allen's algebra. They are integrated into a time-interval framework underlying the universe of discourse of the requirements for a software system to be built. Normally, temporal extensions of MSCs, which exhibit timing assignments, are used to help the analysis of real-time software systems. The proposed approach uses the intuitiveness of MSCs as a means to encourage the analyst to reason about the temporal properties subsumed into the requirements of non-real-time software systems. The paper presents the main ideas for *basic* MSCs.

Our current research analyzes *high level* MSCs [14], with timer, branching and iteration constructs, in the Allen's time-interval structure. In a future work, we intend to compare the theories for interval structures which fulfill the demands of the extended MSC, and the solutions for the interval constraints in the ICG, in order to achieve complete check consistency in a tractable manner. We believe that the presented approach is useful in two complementary ways: as a front-end to existing tools that analyze the consistency of timing assignments, helping their parameterization; and as tool to uncover the implicit temporal assumptions underlying the requirements from an environment where a software system will be built.

Acknowledgments

I would like to thank João José Neto of the Escola Politécnica da Universidade de São Paulo, and Marcelo Finger of the Instituto de Matemática e Estatística da Universidade de São Paulo, for their interest in the research. I would also like to thank Gerard Holzmann, Paolo Terenziani and Elzbieta Hajnicz for their kindness in sending their works, and the reviewers for their helpful comments.

References

[1] Allen, J.F. "Maintaining Knowledge about Temporal Intervals". *Communications of the ACM* **26** (11), pp. 832-843, 1983.

[2] Allen, J.F. "Towards a General Theory of Action and Time". *Artificial Intelligence* **23** (2), pp. 123-154, 1984.

[3] Allen, J.F. & Hayes, P.J. "A Common-Sense Theory of Time", in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, USA, Morgan Kaufmann, pp. 528-531, 1985.

[4] Alur R., Holzmann, G. J. & Peled, D. "An Analyzer for Message Sequence Charts", in T. Margaria & B. Steffen (eds.) 'Tools and Algorithms for the Construction and Analysis of Systems'. LNCS 1055. Springer Verlag, pp. 35-48, 1996.

[5] Andersson M. & Bergstrand, J. "Formalizing Use Cases with Message Sequence Charts". Master Thesis. Department of Communication Systems at Lund Institute of Technology. Malmö, Sweden, 1995.

[6] van Beek, P. "Reasoning about Qualitative Temporal Information". *Artificial Intelligence*, 58, pp. 297-326, 1992.

[7] Ben-Abdallah, H. & Leue, S. “Expressing and Analyzing Timing Constraints in Message Sequence Chart Specifications”. Technical Report 97-04, Department of Electrical & Computer Engineering, University of Waterloo (1997).

[8] Booch, G. “Object-Oriented Analysis and Design with Applications”, Second Edition. Benjamin-Cummings (1994).

[9] Braek, R. & Haugen O. “Engineering Real Time Systems”. Prentice Hall, 1993.

[10] Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F. & Jeremaes, P. “Object-Oriented Development, the Fusion Method”. Prentice-Hall, 1994.

[11] Grabowski, J., Hogrefe, D. & Nahm, R. “Test Case Generation for Temporal Properties”. Technical Report IAM-93-013, Universität Bern, Institut für Informatik, 1993.

[12] Hajnicz, E. “Time Structures: Formal Description and Algorithmic Representation”, in LNAI 1047. Springer Verlag, 1996.

[13] Holzmann, G. “Early Fault Detection Tools”, in T. Margaria & B. Steffen (eds.) ‘Tools and Algorithms for the Construction and Analysis of Systems’. LNCS 1055. Springer Verlag, pp. 1-13, 1996.

[14] “ITU-TS Recommendation Z.120: Message Sequence Chart (MSC)”. ITU-TS, Geneva, September 1993.

[15] “ITU-TS Recommendation Z.120 Annex B: Algebraic Semantics of Message Sequence Charts”. ITU-TS, Geneva, April 1995.

[16] Jacobson, I., Christerson, M., Jonsson, P. & Overgaard, G. “Object-Oriented Software Engineering: A Use Case Driven Approach”, Addison-Wesley, 1992.

[17] Jacobson, I., Ericsson, M. & Jacobson, A. “The Object Advantage: Business Process Reengineering With Object Methodology”, Addison-Wesley, 1995.

[18] Kamp, H. & Reyle, U. “From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory”, Kluwer Academic Publishers, 1993.

[19] Ladkin, P. & Leue, S. “Interpreting Message Flow Graphs”. *Formal Aspects of Computing* **7** (5), pp. 473-509, 1995.

[20] Mauw, S. & Reniers, M.A., “An Algebraic Semantics of Basic Message Sequence Charts”. *The Computer Journal*, 37 (4), pp. 269-277, 1994.

[21] Muniz Silva, P. S. “Análise dos Requisitos de Software em uma Estrutura de Intervalo de Tempo (Software Requirements Analysis in a Time-Interval Structure)” (in Portuguese). Ph.D. Dissertation, Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, São Paulo, Brazil. *To appear in* 1998.

[22] “Unified Modeling Language: Notation Guide Version 1.0”. Rational Software Corporation, 1997.

[23] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorensen, W. “Object-Oriented Modeling and Design”. Prentice Hall, 1991.

[24] Selic, B., Gullekson, G. & Ward, P. “Real-Time Object-Oriented Modeling”. John Wiley and Sons, Inc. (1994).

[25] Spivey, J. M. “The Z Notation: A Reference Manual”, Second Edition. Prentice Hall, 1992.

[26] Zave, P. & Jackson, M. “Four Dark Corners of Requirements Engineering”. *ACM Transactions on Software Engineering and Methodology*, 6 (1), pp. 1-30 (1997).