Some pathological Message Sequence Charts, and how to detect them.

Loïc Hélouët

France Télécom R&D,
2 avenue Pierre Marzin, 22307 Lannion Cedex, France,
loic.helouet@francetelecom.com

Abstract. This paper identifies some confusing Message Sequence Charts, that can be considered as syntactically correct, but may lead to ambiguous interpretations. The first kind of MSC identified appears when parallel components of a parallel component synchronize implicitly to continue an execution. The second case is called non-local choice, and appears when more than one instance is responsible for a choice. Non-local choice has already been studied before, but an extension of the definitions and algorithms is provided. The third case is confluent MSCs, and appears when concurrency is expressed through a choice.

1 Introduction

Since the first standard appeared in 1992, Message Sequence Charts have gained a lot of expressive power. Many elements have been added to the original language: composition in MSC'96, additional time measurement possibilities, and variables in MSC'2000. All these improvements are obviously aiming at a better usability of Message Sequence Charts, and are mainly driven by expressed needs. However, adding a new element to a language may fully satisfy some users, and introduce at the same time confusion for all the others, or even semantics ambiguities.

The main elements of the language (instances, messages, timers) are usually well understood. Furthermore, as far as basic Message Sequence Charts are concerned, very few ambiguities can arise. It is not true when considering composition through parallel, choice, or loop operator, or through High-level Message Sequence Charts. Some graphical inconsistencies in MSC'96 were already pointed out by [7].

HMSCs, for example, allow for the definition of MSCs that are considered as syntactically correct, but the intuitive understanding of which are different from the behavior allowed by the semantics. In most cases, these HMSCs should be considered as pathological, and rejected. Fortunately, the cases introduced within this paper can be easily detected. This detection is based on global properties of the MSC.

As a correct syntax does not ensure the correct understanding of a MSC specification, we argue that a new document should be added to the appendices of recommendation Z.120, as a "methodological guideline", which should precise what a valid MSC should be. The identification of at least two pathological cases and an extension of the definition of non local choice is a first contribution in this direction.

This paper is organized as follows: first we quickly recall the operational semantics of High-level Message Sequence Charts, as defined by [11, 8]. Then , section 3 shows a first ambiguous case, that arise when two parallel components synchronize without any communication. Section 4 recalls the definition of non-local choice, proposes an algorithm to detect it, and then discusses its pathological character. Section 5 identifies another pathological kind of MSC, and proposes an algorithm to detect it.

2 Operational Semantics

2.1 Basic Message Sequence Charts

Many semantics have been proposed for basic Message Sequence Charts (bMSC). The semantics retained for recommendation Z.120 is based on process algebra. However, we consider as natural to model bMSC as a finite, labeled partial order, as in [6, 1].

A bMSC is a tuple $M = (E, \leq, A, I, \alpha)$ where:

- E is a finite set of events,
- $\leq$ is a partial order relation (antisymmetric, reflexive and transitive) called *causal order* on events,
- I is a set of names of instances that perform at least one action in M, and is called the set of *active instances* of M.
- A is a set of action names.
- $\alpha : E \longrightarrow A \times I$ is a labeling of events.

From now, we will note $\phi(e)$ the instance performing event e, ie the instance $i \in I$ such that $\alpha(e) = (a, i)$ for some $a \in A$. Slightly abusing the notation, we will note $\phi(E) = \{\phi(e) | e \in E\}$ the set of instances appearing in any set of events E. For any MSC M, we will note $min(M) = \{e \in E \mid \nexists\ e' \neq e \text{ and } e' \leq e\}$ the set of minimal events of M. For any labeled order $M = (E_M, \leq_M, A_M, I_M, \alpha_M)$, for any set $E' \subseteq E_M$, we will denote by $M_{/E'}$ the restriction of M to events of E'. We will also denote by $M_\emptyset =< \emptyset, \emptyset, \emptyset, \emptyset, \emptyset >$ the empty MSC.

A bMSC defined with a partial order model has the same semantics as the process algebra definition of [11]. The main difference is that we use one single operational semantics rule :

$$\frac{e \in min(M), \alpha(e) = (a, i)}{M =< E_M, \leq_M, A_M, I_M, \alpha_M > \ \xrightarrow{a}\ M_{E-\{e\}}}$$

2.2 High-level Message Sequence Charts

High-level Message Sequence Charts allow for the definition of more elaborated behaviors. Their graphical syntax is given by means of graphs, the nodes of which are references to bMSCs, to HMSCs, or parallel composition of bMSCs and HMSCs references.

One of the key points for HMSC understanding is the semantics of sequential composition. HMSC H_1 in Figure 1 is a sequential composition of bMSCs M_1 and M_2. The semantics of the sequence of bMSCs defined in the standard is a weak sequential composition[1]. The result is an instance-by-instance concatenation, where, for each instance, the maximum event of the first bMSC is linked to the minimum event of the second bMSC. This gives to MSCs an interesting expressive power since communication messages can be accumulated between instances by concatenating basic patterns. Therefore, the HMSC in Figure 2 should have the same operational semantics as the HMSC in Figure 1. If the semantics of weak sequential composition gives HMSCs a huge expressive power, it is also the source of many misunderstandings. However, we think than weak sequence is essential for the design of asynchronous distributed systems, and that the ambiguous cases it may produce is an acceptable price to pay for it.

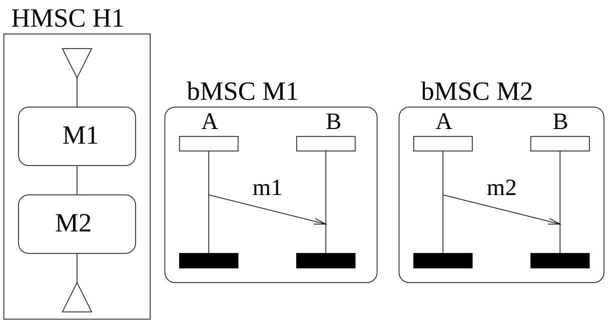

Fig. 1. HMSC H_1: sequence of bMSCs M_1 and M_2.

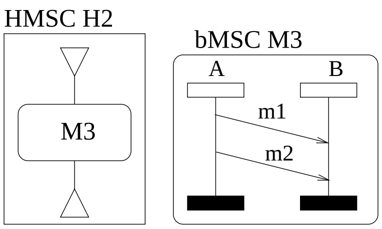

Fig. 2. HMSC H_2 equivalent to HMSC H_1 in Figure 1.

[1] Weak sequential composition is close to the Pratt's local sequencing [10], where ϕ defines locality.

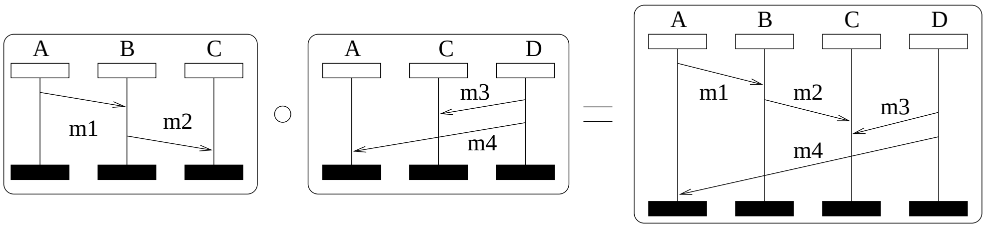

Fig. 3. Chain by chain concatenation of basic message sequence charts.

Let us define the sequencing operator $\circ$ on two MSCs $M_1 = (E_1, \leq_1, A_1, I_1, \alpha_1)$ and $M_2 = (E_2, \leq_2, A_2, I_2, \alpha_2)$: $M_1 \circ M_2 =< E, \leq_{M_1 \circ M_2}, I_1 \cup I_2, \phi >$, where:

- $E = E_1 \uplus E_2$ is the disjoint union of E_1 and E_2,
- $\forall e, e' \in E$, $e \leq_{M_1 \circ M_2} e'$ iff $e \leq_1 e'$ or $e \leq_2 e'$ or $\exists (e_1, e_2) \in E_1 \times E_2\ : \phi_1(e_1) = \phi_2(e_2) \wedge e \leq_1 e_1 \wedge e_2 \leq_2 (e')$
- $A = A_1 \uplus A_2$, $I = I_1 \uplus I_2$,
- $\forall e \in E$, $\alpha(e) = \alpha_1(e)$ if $e \in E_1$ or $\alpha(e) = \alpha_2(e)$ if $e \in E_2$

More intuitively, sequential composition consists in ordering events e_1 in bMSC M_1 and e_2 in bMSC M_2 if they are situated on the same instance, and then calculating the transitive closure of the resulting order. An example of sequential composition is provided in Figure 3. Due to the local sequencing, the emission of m_1 precedes the reception of m_4, and the reception of m_2 precedes the reception of m_3 in the resulting bMSC.

Basic Message Sequence Charts only allow for the definition of very simple scenarios. High-level Message Sequence Charts (HMSC) allow for the definition of more complex behavior, through parallel composition, choice, and sequence operators, and hierarchical construction. HMSC documents can be defined as a collection of graphs, as in [8] :

Definition 1 *A Message Sequence Chart document can be defined by a family of High-level Message Sequence Charts* $\mathcal{F} = \underset{i \in 1..N}{\{} H_i\}$, *where each* H_i *is a High-level Message Sequence Chart.*

Definition 2 *A* HMSC *is a graph* $H = (id, N, Ends, Start, \longrightarrow, \mathcal{M}, l)$, *where:*

- *id is the name of the HMSC,*
- N, *Ends are disjoint finite sets of nodes (Ends is a set of end nodes).*
- *Start is a unique starting node,*
- $\longrightarrow$ *is the transition relation* $(\subseteq (N \cup Ends \cup \{Start\})^2)$,
- $\mathcal{M}$ *is a set of bMSCs, on disjoint sets of events. Each bMSC* $M \in \mathcal{M}$ *is a tuple* $M =< E_M, \leq_M, I_M, \phi_M >$,
- l *is a labeling function on nodes (* $l : N \longrightarrow expr$ *), associating to each node a reference to a basic MSC, to the empty basic Message Sequence Chart* $M_\emptyset$, *to a HMSC, or a parallel composition of references. Nodes in Ends, choice*

and connector nodes will be labeled with $M_\emptyset$*, and will be called* empty nodes. *To simplify notations and algorithms, we will consider the label associated to each node as a parallel composition of bMSC and HMSC :*

$$l(n) = \mathop{\|}_{p \in 1..P} exp_p$$

with $exp_p = r_{M_p}$ *for a reference to a basic MSC* $M_p \in \mathcal{M}$ $exp_p = r_{H_p}$ *for a reference to a HMSC* H_p*. Note that labeling of a node by a reference to a single bMSC or HMSC can be considered as the specific case* $P = 1$*.*

We also require that no cyclic referencing such as in the example in Figure 4 appears in MSC documents. This kind of situation will not be discussed here, but can also be considered as a pathological kind of MSC. It can be easily detected by constructing a graph connecting HMSC referencing each other, and then searching strongly connected components.

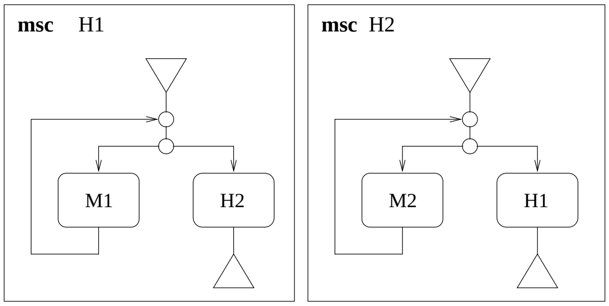

Fig. 4. Cyclic referencing

Definition 3 *A finite* path *of a HMSC* H *is a word* $p = n_1..n_k \in (N \cup Ends \cup Start)^*$ *such that* $\forall i \in 1..k-1, (n_i, n_{i+1}) \in \longrightarrow$*. An* initial path *is a path starting from* $Start$*.*

Definition 4 *A* choice *in a HMSC* H *is a node with more than one successor. A choice* c *defines an alternative between scenarios. Any loop-free path starting from* c *will be called a* branch *of the choice* c*.*

The semantics of HMSC [8] is given by means of regular expressions on bMSC, built from the operators $\mp$ (delayed choice), $\|$ (parallel composition), and $\circ$ (sequential composition). A HMSC defined as a graph can be easily transformed into a set of process algebra expressions with the same meaning. To each node $n \in (N \cup Ends \cup \{Start\})$, we associate an expression:

- if n is an end node, $n = \epsilon$
- if n is a choice node, $n = \mathop{\mp}_{\{n_i | n \longrightarrow n_i\}} ni$

- if n is a node with one single successor n', and $l(n) \neq r_{M_\emptyset}$, then $n = l(n) \circ n'$
- if n is a node with one single successor n', and $l(n) = r_{M_\emptyset}$, then $n = n'$

This definition by means of process algebra has the same expressive power as the hierarchical graphs. This is not really surprising, as HMSCs are just automata labeled with expressions. Hierarchical graphs are often more adapted to algorithmic considerations, and process algebra facilitates discussions on semantics points. During the rest of the paper, we will use both representations.

Let us now recall some operational semantics rules for HMSCs defined in [8]. Note that we slightly adapted the rules to fit our partial order representation. Furthermore, we only recall rules that will be needed in the next sections.

First, a permission relation $\cdots\overset{a}{\cdot}\!\!\rightarrow$ is defined. Using our partial ordering definition for MSCs, this permission relation becomes :

$$\frac{\phi(a) \notin \phi(E_x)}{x \cdots\overset{a}{\cdot}\!\!\rightarrow x} \qquad \frac{x \cdots\overset{a}{\cdot}\!\!\rightarrow x',\ y \cdot\!/\!\overset{a}{\cdot}\!\!\rightarrow}{x \mp y \cdots\overset{a}{\cdot}\!\!\rightarrow x'} \qquad \frac{x \cdots\overset{a}{\cdot}\!\!\rightarrow x',\ y \cdots\overset{a}{\cdot}\!\!\rightarrow y'}{x \circ y \cdots\overset{a}{\cdot}\!\!\rightarrow x' \circ y'}$$

More intuitively, *expr* $\cdots\overset{a}{\cdot}\!\!\rightarrow$ if all events of *expr* and action a are independent, and can be executed concurrently. We can now give some rules that will be needed in the rest of the paper. For a complete semantics, interested reader is referred to [8].

$$\frac{x \cdots\overset{a}{\cdot}\!\!\rightarrow x',\ y \xrightarrow{a} y'}{x \circ y \xrightarrow{a} x' \circ y'} \qquad \frac{x \xrightarrow{a} x'}{x \| y \xrightarrow{a} x' \| y} \qquad \frac{y \xrightarrow{a} y'}{x \| y \xrightarrow{a} x \| y'}$$

$$\frac{x \xrightarrow{a} x',\ y \not\xrightarrow{a} y'}{x \mp y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y',\ x \not\xrightarrow{a} x'}{x \mp y \xrightarrow{a} y'}$$

3 Implicit synchronization

MSC are supposed to be very intuitive and visual. However, the interpretation of some constructions may be in contradiction with the semantics. In most cases, this divergence between the effective behavior and the intended behavior is just due to a bad knowledge of the language semantics, or to some abusive use of its constructs. The first pathological MSC kind defined in this paper is *implicit synchronization*.

Consider the HMSC of Figure 5. From the previous translation rules, we can define the same HMSC with the following expressions:

$Pathology1 = Start_{P1}$	$H1 = Start_{H1}$	$H2 = Start_{H2}$
$Start_{P1} = np1$	$Start_{H1} = n1h1$	$Start_{H2} = n1h2$
$n1p1 = n2p1$	$n1h1 = n2h1$	$n1h2 = n2h2$
$n2p1 = (r_{H1} \| r_{H2}) \circ n1p1$	$n2h1 = n3h1 \mp n4h1$	$n2h2 = n3h2 \mp n4h2$
	$n3h1 = r_{M1} \circ n1h1$	$n3h2 = r_{M3} \circ n1h2$
	$n4h1 = r_{M2} \circ n5h1$	$n4h2 = r_{M4} \circ n5h2$
	$n5h1 = \epsilon$	$n5h2 = \epsilon$

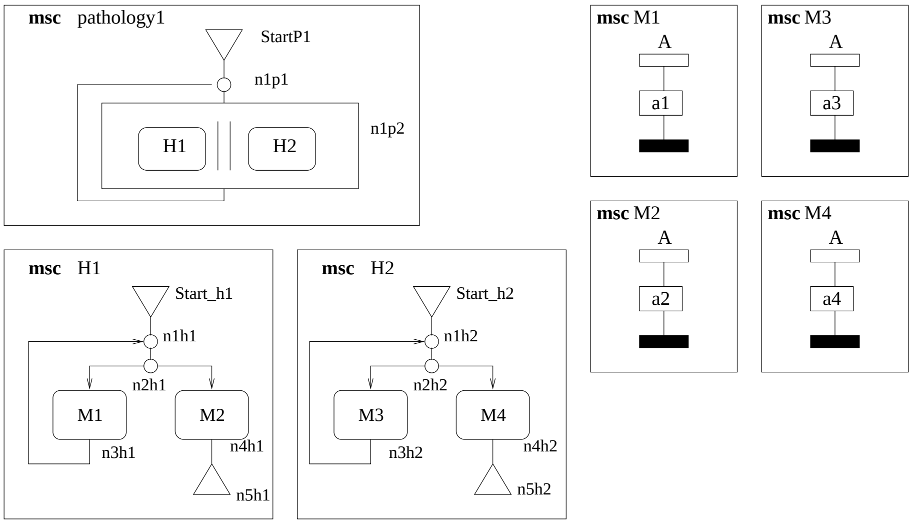

Fig. 5. Implicit synchronization

According to the operational semantics rules, as $H1$ and $H2$ are composed in parallel, $a1$ and $a2$ can be executed concurrently with $a3$ or $a4$, and conversely. This may mean that instance A is composed of more than one entity, that may evolve concurrently. So, events $a3$ and $a4$ can not prevent $a1$ and $a2$ from being executed. However, $np1$ rewrites to $r_{H2} \circ np1$ after executing action $a2$. As $H2 \cdot \not\xrightarrow{a1}$ and $H2 \cdot \not\xrightarrow{a2}$, actions $a1$ and $a2$ can not be executed from $r_{H2} \circ np1$, unless $a4$ is executed, allowing to rewrite $r_{H2} \circ np1$ into $np1$. The automata corresponding to this semantics is provided in Figure 6. The language defined this way is $\big((a1 + a3)^*.((a2.a3^*.a4) + (a4.a1^*.a2))\big)^*$. A synchronization between the two parallel components is defined. One may wonder if this behavior was really intended, and if HMSC of Figure 7, in which the two parallel components behave in parallel without synchronizing does not fit better the expected behavior.

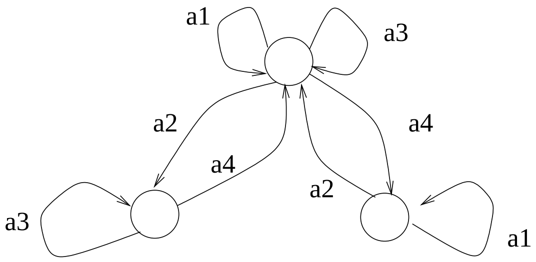

Fig. 6. Operational semantics for HMSC in Figure5

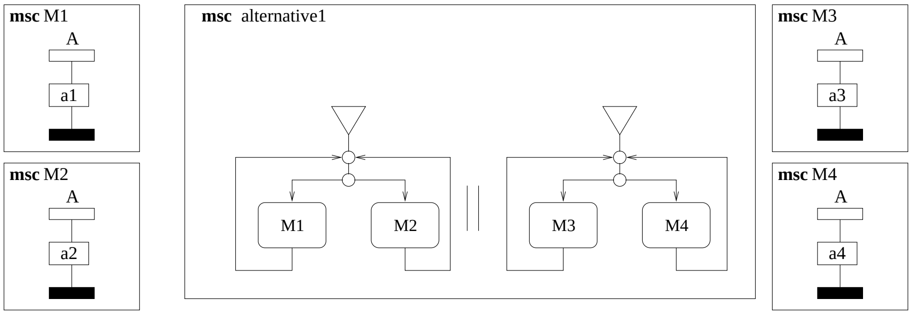

Fig. 7. Alternative intended behavior

Of course, in the general case, it is false that $(x \| y)^*$ and $(x^* \| y^*)$ have the same operational semantics. Hence, the end of the parallel frame acts as a synchronizing point, which may not be obvious for the designer. According to the semantics, one may suppose that there is a fork and join between the two operands of the parallel composition, that is abstracted in the MSC representation.

This implicit synchronization is not dramatic, as HMSCs of Figures 5 and 7 are both correct. However, the fact that a HMSC may express more than what was really intended points out the need for simulation tools exhibiting possible behaviors of a specification. Furthermore, it should be clear that events composed in a parallel frame can be interleaved, but are not always independent, which explains the synchronization when exiting the frame.

One may note that there is not explicit synchronization construct in Message Sequence Charts. Adding this new feature to the language should not be too difficult, as synchronization reduces concurrency, and consequently the set of behaviors described by a MSC. However, the discussion of a synchronization construct is beyond the scope of this paper.

4 Non-local choices

Implicit synchronization is not the only kind of MSC where control is hidden. Another situation potentially leading to erroneous interpretation is called *non-local choice*. The generally admitted meaning of non-local choice [2] is when more than one instance can decide to perform a scenario or another at a choice node. The intended behavior is that the first instance able to perform the choice chooses a behavior. The next instances reaching the same occurrence of this choice have to conform to the chosen scenario. This results in a behavior in which an instance "knows" what to do at a choice node without any communication. An example of non local HMSC is given in Figure 8. In this example, if instance A chooses to send message $m1$ then instance B must conform to scenario $M1$ and receive $m1$. Conversely, if instance B chooses to send message $m2$ then instance A must conform to scenario $M2$ and receive $m2$. To implement only these two scenarios, a designer would have to use a consensus or synchronization mechanism, which is not explicitly represented.

A formal definition of non-local choice was previously given in [2]. This definition does not take parallel composition into account, and assumes that any instance should perform a communication in each bMSC referenced by a successor of the choice node. This assumption limits the search for non-local choice to the set of immediate successors of a choice node of a HMSC. However, [4] shows that when weak sequential composition of bMSCs with disjoint set of instances is considered, non-local choice is not a local property . Therefore, a global definition of non-local choice must be provided. Consider HMSC in Figure 9: choices seem to be local, but the decision to perform a scenario can be taken by A or C. This shows that non-locality is a global property, which has to be computed on the HMSC structure. The definition has thus to be extended, and should take into account parallel composition and hierarchy.

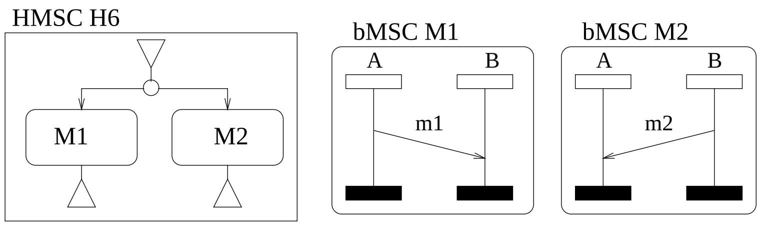

Fig. 8. non local HMSC

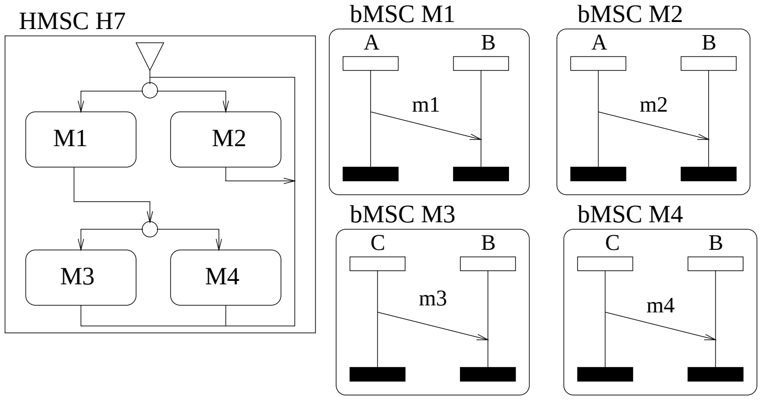

Fig. 9. An example HMSC that may seem local

Definition 5 *Let expr* $= \underset{i \in 1..N}{\mp} n_i$ *be an expression defining a choice. We will say that expr is a non local choice if there are two events* e_1 *and* e_2 *such that* $expr \xrightarrow{e_1}$, $expr \xrightarrow{e_2}$, *and* $\phi(e_1) \neq \phi(e_2)$.

We have shown that the locality of a choice is a global property. So, it may be very difficult to detect it, even with good knowledge of the semantics. As the graphical and process algebra representations of HMSCs are dual, there must be a definition of non local choice holding for graphs. We now show that non locality can be expressed as a global property on the paths of a HMSC document. This immediately provides us with an algorithm for non local choices detection.

Definition 6 *Let* H_i *be a HMSC, and* x *be a node of* H_i *A* maximal path *starting from* x *of* H_i *is either :*

- *a finite path of the form* $w = x.n_1...n_k$ *where* $n_k \in Ends_i$,
- *a finite path of the form* $w = x.v$ *such that it is impossible to leave the sequence* $(v)^\omega$ *(any path of the HMSC* H *starting with* $x.v$ *is a prefix of* $x.v.v^\omega$*).*

Algorithm: Calculus of the maximal paths of HMSC H_i starting from node x

Maximal_path(H_i, x)=
$P = \{x\}$ `/* set of path of H */`
$AP = \emptyset$ `/* set of acyclic path */`
while $P \neq \emptyset$ **do**
  $AP = AP \cup \{w.n | w = n_1...n_k \in P \ , \ n_k \longrightarrow_i n \ , \ n \in End_i\}$
    $\cup \{w = n_1...n_k.n...nk \in P | n_k \longrightarrow_i n\}$
    `/* paths for which adding node n creates a cycle */`
  $P' = \{w.n | w \in P, w = n_1...n_k, n_k \longrightarrow_i n, and w \neq n_1...n_k.n...nk, n \notin End_i\}$
  $P = P'$
end while
$MAP = \emptyset$
`/* remove paths that stop on a cycle but are not prefixes of infinite paths of the form` $v.(v')^\omega$ `*/`
for all $w \in AP$ **do**
  if $\nexists w' \neq w \in AP$ such that $w' = w.v$ **then**
    $MAP = MAP \cup \{w\}$
  end if
end for
return(MAP)

Consider, for example the HMSC of Figure 10. The set of maximal acyclic path of this HMSC is :

$$MAP(H) = \{ \ start.n1.n2.n4.n5; \ start.n1.n2.n3.n2.n4.n5; \\ start.n1.n2.n6.n7.n8.n7; \ start.n1.n2.n3.n2.n6.n7.n8.n7 \}$$

$n1.n2.n3$ is not a maximal path, as it is always possible to leave cycle $(n2.n3)^*$, for example by choosing $n4$. Note that the maximal paths are computed without considering what a node references. Maximal path will be used to detect minimal events on each branch of a choice. Once maximal paths have been found, scenarios attached to each path are computed.

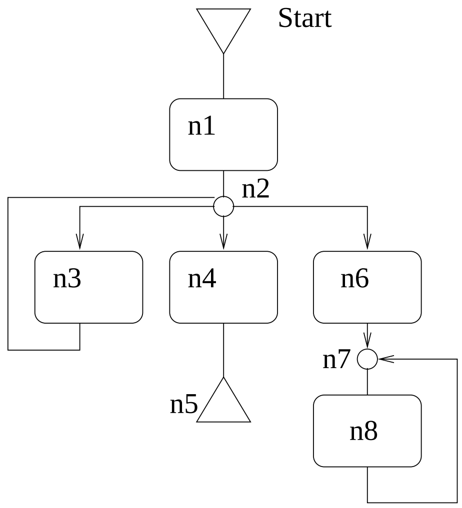

Fig. 10. Example of HMSC

Definition 7 *The* path order family $POF(w)$ *associated to a path* $w = n_1..n_k$ *is a set of partial orders built from the sequences of bMSCs along this path. It is defined recursively:*

- *let* n_1 *be a node and* v *be a path of a HMSC.* $POF(n_1.v) = \{o_1 \circ o_2 | o_1 \in POF(n_1), o_2 \in POF(v)\}$
- *For any node* n, $POF(n) =$
  - $M_\emptyset \quad if \quad l(n) = \epsilon$
  - $\{M\} \quad if \quad l(n) = r_M$
  - $\displaystyle\bigcup_{\pi \text{ permutation on } 1..N} \{o_{\pi(1)} \circ \cdots \circ o_{\pi(N)} | \exists w_i \in MAP(H_i, Start_i), o_i \in POF(w_i)\}$ $if \quad l(n) = \| r_{H_i}$

Theorem 1 *Let* c *be a choice node in a HMSC* H, *and* $expr_c$ *the process algebra expression associated to this choice.* $expr_c \xrightarrow{e}$ *if and only if* $\exists w \in MAP(H, c), \exists o \in POF(w)$, *and* $e_1 \in min(o)$.

proof: To complete the proof, we need some lemmas:
Lemma 1: let $expr_n$ be the expression associated to a node n.

If $expr_n = x \circ y$ and $y = \mp exp_i$ then $expr_n$ can be rewritten into $expr'_n = \mp y \circ exp_i$, If $expr = x$ and $x = y$, then $expr_n$ can be rewritten into $expr'_n = y$

We will say that $expr'_n$ is obtained by rewriting x.

For any event e, we have $expr_n \xrightarrow{e} \Longleftrightarrow expr'_n \xrightarrow{e}$.

proof: From the definitions. $\Box$

Lemma 2: Let c be a choice node. If there is a sequence of rewritings $w = n1.n2.\ldots.nk$ of$expr_c$, such that any pair (n_i, n_{i+1}) is unique in w then w is a maximal path starting from c.

proof: From the definitions. $\Box$

Lemma 3: Let $expr_n$ be the expression associated to a node n. $expr_n \overset{e}{\cdots\rightarrow}$ if and only if

$\exists o =< E_o, \leq_o, A_o, I_o, \alpha_o >\in POF(n)$, such that $\phi(E_o) \neq \phi(e)$.

proof: By induction on the depth of HMSC references. $\Box$

From Lemma 1, and lemma 2, we know that $expr_c \xrightarrow{e}$ if and only if:

- for all w, sequence of rewritings of $expr_c$ rewriting $expr_c$ into $expr'_c$
- $w \in MAP(H, c)$, and
- $expr'_c \xrightarrow{e}$

As $expr'_c \xrightarrow{e}$, then there exists one sequence $w = n1.\ldots.n_k$ and $i \in 1..k$ such that $n_i \xrightarrow{e}$, and for all $x < i$:

- there is an expression $n_x = expr \circ n_{x+1}$ with $expr \overset{e_1}{\cdots\rightarrow}$, or
- $n_x = \underset{j\in 1..N}{\mp} n_j$ with $n_{x+1} = n_j$ for some j, or $n_x = n_x + 1$

So, w is of the form $w = v.n_i.v'$, and using lemma 3, we can say that $\exists o_v \in POF(v)$, $\phi(e) \notin \phi(E_o)$, and $\exists o_i \in POF(n_i)$ such that $e \in min(o_i)$.

Consequently, $expr_c \xrightarrow{e}$ if and only if $\exists w \in MAP(H, c)$, $\exists o \in POF(w)$, and $e \in min(o)$. $\Box$

From this definition, the search for non local choice from graphs is straightforward.

Algorithm: Calculus of the orders associated to a path w

Compute_orders(w)=
$O_w = \emptyset$; $i = 1$
while $i <= |w|$ **do**
  if $l(w[i]) \neq M_\emptyset$ **then**
    if $l(w[i]) = r_M$ **then**
      $O'_w = \{o \circ M | o \in O_w\}$
    else if $l(w[i]) = \underset{j\in 1..P}{\|} expr_j$ **then**
      `/* compute the POF associated to each expression */`
      for all $j \in 1..P$ **do**
        if $expr_j = r_M$ **then**
          $TABO[j] = \{M\}$
        else if $expr_j = H_j$ **then**

$$TABO[j] = \bigcup_{p \in MAP(H_j)} Compute\_Orders(p)$$

```
        end if
      end for
      O_H = ∅
      for all π permutation on 1..P do
        O_H = O_H ∪ {o_1 ∘ ... o_P | o_i ∈ TABO[π(i)]}
      end for
      O'_w = {o ∘ o' | o ∈ O_w, o' ∈ O_H}
    end if
    O_w = O'_w
  end if
  i = i + 1
end while
return(O_w)
```

Algorithm: Non local choice detection

```
Non_Local(c)=
MAP = Compute_MAP(c)
INST = ∅
for all p ∈ MAP do
  O_c = Compute_Orders(p)
  for all o ∈ O_c do
    INST = INST∪φ(min(o)) /* set of instances participating to*/
                           /* the decision at a choice node */
  end for
end for
if |INST| > 1 then
  return(true) /* c is a non local choice */
else
  return(false)
end if
```

Non local choices are not very ambiguous HMSC, they just define multiple scenarios. However, they may require additional synchronization mechanisms when an implementation allowing only the defined behaviors is planned. They can therefore be considered as too abstract for some purposes. Hence, we think that non -local choice is a property that may cause misunderstanding between the requirement capture and the specification phase of a system development when it is not detected.

5 Confluence

We now define another pathological kind of Message Sequence Charts, called confluent MSC. A MSC is said confluent if a parallel composition is expressed by means of a choice node. Consider, for example, HMSC of Figure 11.

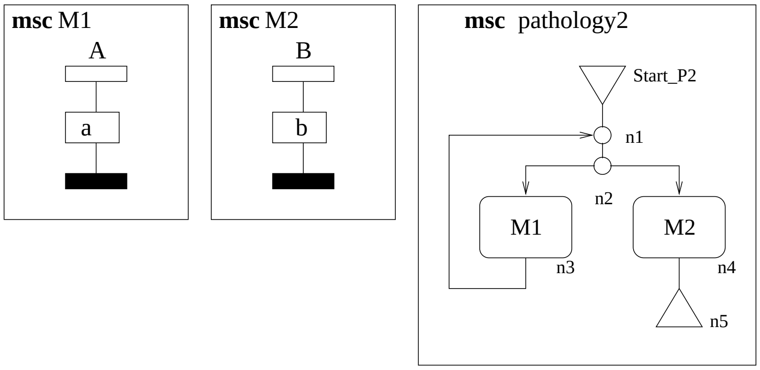

Fig. 11. Confuence

$$
\begin{array}{ll}
Pathology2 = Start_{P2} & n3 = r_{M1} \circ n1 \\
Start_{P2} = n1 & n4 = r_{M2} \circ n5 \\
n1 = n2 & n5 = \epsilon \\
n2 = n3 \mp n4 &
\end{array}
$$

According to the operational semantics rules, $n2 \xrightarrow{a}$ and $n2 \xrightarrow{b}$. Furthermore, as $n3 \overset{b}{\cdots\rightarrow}$ the execution of b may result from unbounded unfoldings of $n3 = r_{M1} \circ n1$. Therefore, executing b does not indicate how many a's should be executed, and the language defined by this MSC is $a^*.b.a^*$. This behavior can also be exhibited by the HMSC of Figure 12, through a parallel composition. However, we can suppose that the intended behavior was to define b as an exit event for the loop. So, the desired behavior would have defined the language $a^*.b$. Obviously, a system where the exit condition of a loop is set by a process that never communicates with processes activated in the loop is highly pathological.

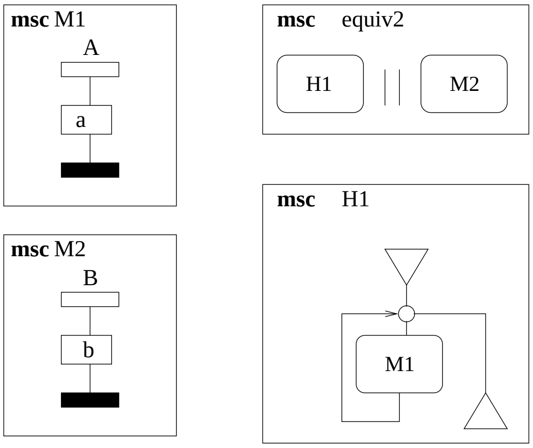

Fig. 12. HMSC with the same behavior as in Figure 11

This kind of specification is called confluent, and was already identified in [9], and proved undecidable. However, this undecidability is a consequence of a weaker ordering relation than what is usually used for MSC ([9] considers that receptions of messages coming from different senders are not ordered even if they are sequential on an instance axis). We now give a formal definition for confluence, an show that it can be detected using a simple algorithm. Up to now, the definition and algorithm only concerns HMSC referencing bMSC, but can probably be extended to include parallel frames and hierarchy.

Definition 8 *Let H be a HMSC referencing bMSCs, and n be a choice node of H. n is said confluent if and only if there is a cycle $c = n.n_1....n_k$, and a path $p = n.x_1...x_k$ starting from n, and there is a minimal event e of O_p such that $\phi(e)$ is not an active instance of O_c*

Note that a confluent Message Sequence Chart is necessarily non-local. We do not define the cycle computation, that can be performed using the well known Tarjan algorithm [13].

Algorithm: Confluence detection for a choice node s of a HMSC H

Confluent(H, n)=
CY = compute_cycles(n) `/* set of cyclic paths containing n */`
MAP = compute_map(H,n)`/*maximal acyclic paths starting from n*/`
for all $c \in CY$ **do**
  for all $p \in MAP$ **do**
    if $\exists i \in \phi(min(Op))$ such that $i \notin \phi(O_c)$ **then**
      choice node n is confluent.
    end if
  end for
end for

Confluent Message Sequence Charts can be considered as a bad use of choice in a specification. Obviously, they may lead to misunderstanding, and therefore should be avoided. Yet, confluence is again a global property, that a correct syntax can not prevent. One may remark that the correction of confluence is easy for the example of this section (transform HMSC of Figure 11 into HMSC in Figure 12). However, it may be more difficult or even impossible to find another HMSC exhibiting the same behavior.

6 Conclusion

This article has show three kind of High-level Message Sequence Charts that may cause misunderstanding. The first case concerns implicit synchronization in loops, and can be considered as a bad specification or not, depending on the designer's awareness of the HMSC semantics. The second case is the well known non-local choice. Again, non-local specifications should be considered as incomplete rather than erroneous. However, when HMSCs are designed as a premise for the production of a specification, non-locality points out parts of the specification where additional information is needed. Therefore, a special attention should be paid to these parts of the specification where erroneous behaviors may be introduced after the requirement phase. The last situation described is confluence, an clearly appears as a bad specification : choices are used as parallel composition. Note that all these cases are expressed with MSC'96. We can also expect new constructs of MSC'2000 to introduce more interpretation problems, due to inheritance, or variables, that will be revealed when the language is put to practice.

Up to now, there is no documentation providing guidelines for correct message sequence Charts construction. We think that such a document could reference cases of bad usage of MSCs, that the grammar can not reject. The responsibility for avoiding such MSCs would then be left to the designer, or to automated detection algorithms implemented in MSC tools when possible.

References

1. Alur R., Holzmann G. , Peled D., *An analyzer for Message Sequence Charts*, Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96), LNCS no 1055 , pp 35-48, Passau, Germany, 1996.
2. Ben-Abdallah H., Leue S., *Syntactic Detection of Process Divergence and non-Local Choice in Message Sequence Charts*, Proceedings of TACAS'97, Lecture Notes in Computer Science, Vol. 1217, Brinksma.E editor, Springer-Verlag publisher, pp. 259-274, 1997.
3. Graubmann P. , Rudolph E. , Grabowski J., *Towards a Petri Net Based Semantics Definition for Message Sequence Charts*, In: SDL'93 - Using Objects (Editors: O. Faergemand, A. Sarma), North-Holland, October 1993.
4. Hélouët L., Jard C., *Conditions for synthesis from Message Sequence Charts*, 5th international workshop on Formal Methods for Industrial Critical Systems (FMICS), Berlin, april 2000.
5. ITU-T, *Message Sequence Chart (MSC 2000)*, ITU-T Recommendation Z120, 2000.
6. Katoen J.P., Lambert L., *Pomsets for message sequence charts*, proceedings of *SAM98:1st conference on SDL and MSC*, pp. 291-300, 1998.
7. Loidl S., Rudolph E., Hinkel U., *MSC'96 and Beyond - a Critical Look*, Proceedings of the Eight SDL Forum, SDL'97: Time for Testing - SDL MSC and Trends, A. Cavalli and A. Sarma, editors, Evry, France, 23-26 September, 1997.
8. Mauw S., Reniers M. , *High-level Message Sequence Charts*, Proceedings of the Eight SDL Forum, SDL'97: Time for Testing - SDL MSC and Trends, pp 291-306, A. Cavalli and A. Sarma, editors, Evry, France, 23-26 September, 1997.
9. Muscholl A., Peled D., *Message sequence graphs and decision problems on Mazurkiewicz traces*, Proc. of MFCS'99, Lecture Notes in Computer Science 1672, pp. 81-91, 1999.
10. Pratt.V , *Modeling Concurrency with Partial Orders*, International journal of Parallel Programming, Vol. 15, No. 1, pp. 33-71, 1986.
11. Reniers M., Mauw S., *An algebraic semantics for basic message sequence charts*, The Computer Journal", Vol. 37, No. 4, pp. 269-277, 1994.
12. Reniers M., *Message Sequence Charts: Syntax and Semantics*, PhD Thesis, Heindhoven University of Technology, 1998.
13. Tarjan.R, *Depth-first search and linear graph algorithms*, SIAM Journal of Computing, 1(2), 1992.