Non-local Choice and Implied Scenarios

Haitao Dan, Robert M. Hierons and Steve Counsell
School of Information Systems, Computing & Mathematics
Brunel University
Uxbridge, Middlesex UB8 3PH, UK
{*haitao.dan, rob.hierons, steve.counsell*}*@brunel.ac.uk*

***Abstract*—A number of issues, such as non-local choice and implied scenarios, that arise in Message Sequence Charts (MSCs) have been investigated in the past. However, existing research on these two issues show disagreements regarding how they are related. In this paper, we analyse the relations among existing conditions for non-local choice free and Closure Conditions (CCs) for implied scenarios. On the basis of this, we propose a new definition for non-local choice and a non-local choice free condition derived from CCs of implied scenarios. Compared to existing conditions, we argue that the new condition covers more non-local choices that satisfy the informal idea of non-local choice. We formally show that the existence of non-local choices in an MSC specification results in implied scenarios and the appearance of implied scenarios according to corresponding CCs means there are non-local choices in the specification.**

***Keywords*-Message sequence charts (MSCs); Non-local choice; Implied scenarios.**

I. Introduction

Message Sequence Charts (MSCs) are a specification language suitable for describing behaviour of distributed systems [12]. MSCs have become increasingly popular in the telecommunication and software industries and are widely used for requirements analysis and system design [18], [10], [23].

In MSCs, communications are abstracted as orders between the sending and the corresponding receiving of messages, so it is relatively simply to use MSCs to model distributed systems. However, ambiguities may be introduced by the abstraction. Research has been conducted to investigate communication patterns that introduce uncertainties and methods to detect these patterns. Patterns leading to unspecified behaviours are referred to as *pathologies* in [5], [11]. Known pathologies include race [3], [19], [22], non-local choice [14], [7], implied scenarios [1], [24], [17], implicit synchronization [11], confluence [11], process divergence [7], boundedness [4], false-underspecification [5] and time inconsistency [6].

MSCs have a *local* assumption which prevents a process from directly accessing the status of other processes. As a consequence, it is possible that the behaviours modelled by MSCs seem correct, but in fact they violate the local assumption. Non-local choice is a pathology induced by the local assumption [16], [15]. It can be described as a choice that depends on information from other processes, but the status of the other processes is not accessible due to the local assumption. Non-local choices in MSCs may cause synchronisation problems in the implemented distributed systems.

We note that there are different non-local choice free conditions in the literature [7], [11], [5]. The condition in [7], [11] is based on explicit branching choice nodes but the condition in [5] is not. However, both conditions are derived according to the same informal description of non-local choice. In this paper, we compare the two conditions and show that an MSC specification may contain hidden non-local choices which may exist without explicit branching structures. In addition, the condition in [5] captures hidden non-local choices but may fail to capture some others which are formed by receive events or termination of processes.

Interestingly, a number of researchers have observed that non-local choice strongly relates to another pathology of MSCs: implied scenarios. Muccini [21] claimed that “an implied scenario is due to a non-local choice situation” whereas Uchitel et al. [24] believed that “non-local choices are implied scenarios; the converse is not the case.” Mooij et al. pointed out the inconsistency between [21] and [24] but did not provide a formal analysis on this topic [20]. It may worth noting that the implied scenario in [1] and [24], [17] are slightly different. [1] considered implied scenario under asynchronous non-FIFO communication while [24], [17] assume the communication is synchronous so the sending and receiving of a message are regarded as one event. According to [1], when attempting to synthesise concurrent automata from a set of basic MSCs (bMSCs), in addition to the behaviours expressed by MSCs, the synthesised automaton may show additional behaviours which are implied scenarios.

In this paper, we analyse the relation between conditions of non-local choice free and Closure Conditions (CCs) for implied scenarios following definitions in [1]. We show that CCs in [1] used for implied scenarios can capture all non-local choices including hidden non-local choices and non-local choices containing receive events or terminations. As a result, we give a new non-local choice definition and a non-local choice free condition (Definition 22) derived from CC3’ and CC2’ in [1]. In order to transform the two CCs to one definition, we introduce the termination of each process into the alphabet of the MSC language and show that the extension leads to combined CCs: CC4 or CC5. Furthermore, CC5 is used in the new definition of implied scenario, definition of non-local choice and condition for non-local choice free.[1] Finally, a polynomial algorithm (Algorithm 1) for detecting non-local choice in MSC specifications with a finite number of bMSCs is proposed by extending the algorithm for checking CC3' in [1].

The remainder of this paper is organised as follows. In the next section, we introduce the semantics of MSCs, High-level MSCs (HMSCs) and other relevant definitions. In Section 3, the relationships and differences among conditions of non-local choice free and implied scenarios are analysed. In Section 4, we derive a new condition for non-local choice. Section 5 gives the algorithm for detecting non-local choice based on results provided in Section 4. In the last section, conclusions are drawn and future work is described.

II. Preliminaries

In this section, we give the formal definition of core concepts of MSCs. For a detailed description of MSCs, the reader is referred to [12]. MSCs are discussed under non-degeneracy conditions proposed in [1] throughout this paper and will be introduced after the formal definition of bMSCs.

A. bMSCs

*Definition 1: (**bMSCs**)* An *bMSC* M is a tuple $\langle E, C, \mathcal{P}, l, msg, < \rangle$ where: E is a disjoint set of events, C is a message alphabet and $\mathcal{P} = \bigcup_{i=1}^{n} P_i$ is a set of processes; E is partitioned into a set S of send events and a set R of receive events, $E = S \cup R$; $l : E \mapsto \mathcal{A}$ is a labelling function and $\mathcal{A} = \mathcal{A}^S \cup \mathcal{A}^R$ where $\mathcal{A}^S = \{send(i,j,a) : 1 \leq i,j \leq n \wedge a \in C\}$ is the set of sending of messages and $\mathcal{A}^R = \{receive(i,j,m) : \ 1 \leq i,j \leq n \ \wedge \ m \in C\}$ is the set of receiving of messages; $send(i,j,m)$ represents the sending of message m from P_i to P_j and $receive(i,j,m)$ the receiving of the corresponding message[2]; $\mathcal{A}_i$ represents the set of labels on process P_i; $msg : S \mapsto R$ is a bijection between send and receive events, matching each send with its corresponding receive; the inverse mapping is $msg^{-1} : R \mapsto S$ between receive and send events; there is a helper mapping $p : E \mapsto [1,n]$ that maps each event $e \in E$ to the index of the process on which e occurs; for each $1 \leq i \leq n$, a total order $<_i$ on the events of process P_i, i.e., on the elements of $p^{-1}(i)$, such that the transitive closure of the relation $< \doteq \bigcup_{1 \leq i \leq n} <_i \cup \{(s, msg(s)) : s \in S\}$ is a partial order on E, namely, visual order ($<^*$). Note that, since $<_i$ is a total order, it is antisymmetric.

[1]No unified formal definition of implied scenario is given in [1] because the conditions are separated into two parts.

[2]We will use $!m$ and $?m$ as abbreviations of $send(i,j,m)$ and $receive(i,j,m)$, respectively, where i,j are clear.

In this paper, for a message m we restrict $p(!m) \neq p(?m)$ which means that the messages sent and received by the same process are not considered. According to [1], under the non-degeneracy condition[3], given a bMSC M, a *linearisation* of M is a string $w = w_1 \cdots w_{|E|}$ over $\mathcal{A}$ if and only if there exists a total order $e_1 \cdots e_{|E|}$ of the events in E such that whenever $e_i <^* e_j$ we have $i < j$, and for $1 \leq i \leq |E|$, $w_i = l(e_i)$. Conversely, a well-formed and complete word uniquely characterises a bMSC. A word is *well-formed* if all its receive events have earlier matching sends and a word w is *complete* if all send events have matching receives. A bMSC M can be represented by a set of well-formed and complete words and the set of words is the language of the bMSC, denoted as $L(M)$. We use $pref(w)$ and $pref(L)$ to denote the set of prefixes of word w and the set of prefixes of language L, respectively.

B. MSC specifications

We consider an MSC specification to be a countable set of bMSCs. The semantics of an MSC specification is the language of the specification.

Definition 2: (**Language of MSC Specifications**) For an MSC specification $\mathcal{M}$, the language of $\mathcal{M}$, $L(\mathcal{M})$, is the union of the sets of words of all bMSCs in $\mathcal{M}$, $L(\mathcal{M}) = \bigcup_{M \in \mathcal{M}} L(M)$. $L(\mathcal{M})$ is regarded as the semantics of $\mathcal{M}$.

Accordingly, we say that bMSC M is a *member bMSC* of $\mathcal{M}$ if $L(M) \subseteq L(\mathcal{M})$. In a bMSC, the behaviour of a process is modelled by a totally ordered sequence of events corresponding to a single word on labels. For an MSC specification, behaviours of process P_i are captured by the projection of the language $L(\mathcal{M})$ on P_i, denoted as $L(\mathcal{M})|P_i$. Process languages have the following property.

Proposition 3: Process languages are regular languages.

Complex MSC specifications are normally defined by HMSCs. HMSCs can be a hierarchical structure and are introduced to describe complex logic among distributed behaviours. An HMSC may refer to a set of bMSCs and a set of HMSCs. HMSCs can be used to form larger scenarios involving sequencing, iteration, and branching among bMSCs.

Definition 4: (**HMSCs**) An *HMSC* H is a graph which can be represented by a tuple $\langle \mathcal{V}, v_0, V_T, V_M, \mathcal{E}, \mathcal{N}, \mu \rangle$ where: $\mathcal{V}$ is a finite set of nodes; $v_0 \in \mathcal{V}$ is the initial node; $V_T \subset \mathcal{V}$ is a set of terminal nodes; $V_M \subset \mathcal{V}$ is a set of MSC nodes; $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges; $\mathcal{N}$ is a set of MSCs (can be bMSCs or HMSCs); a labelling function $\mu : V_M \mapsto \mathcal{N}$ that maps each node $v \in V_M$ to an MSC in $\mathcal{N}$.

A node in HMSCs may refer to another HMSC and expanding the HMSC nodes leads to a hierarchical structure. We note that recursive structures are not permitted by the standard – an HMSC cannot refer to itself. The expanded

[3]Non-degeneracy means degenerate bMSCs are not allowed. A bMSC is degenerate if two send events $e1$ and $e2$ exist such that $l(e1) = l(e2)$, $e1 < e2$ and $msg(e2) < msg(e1)$.

HMSCs which contain no HMSC nodes are called MSC Graphs (MSGs). Each path from v_0 to $v_t \in V_T$ in an MSG describes a scenario which is constructed by connecting the bMSCs represented by the nodes following the path. The standard way to connect bMSCs is through weak sequencing.

Definition 5: (**Weak Sequencing**) The weak sequencing of two bMSCs $M = \langle E, C, \mathcal{P}, l, msg, < \rangle$ and $M' = \langle E', C', \mathcal{P}', l', msg', <' \rangle$ with disjoint sets of events is denoted $(M \bullet M')$ and is defined by the bMSC $\langle E \oplus E', C \cup C', \mathcal{P} \cup \mathcal{P}', l \cup l', msg \cup msg', <^{\bullet} \rangle$, where $\oplus$ is disjoint union, and $<^{\bullet}$ is the union of the set of total orders $<_i^{\bullet}$ such that $e <_i^{\bullet} e'$ if and only if $e <_i e'$ or $e <_i' e'$ or $e \in E$ and $e' \in E'$ and $p(e) = p(e') = i$.

Intuitively, an HMSC defines a set of sequences of nodes from the initial node to a terminal node in $\mathcal{V}$. By applying weak sequencing, these sequences of nodes are mapped to a set of bMSCs. Each constructed bMSC models a complete scenario of the underlying system and is a member bMSC of the specification denoted by the HMSC. We note that there may be cycles in HMSCs, and so an HMSC can correspond to an infinite set of bMSCs. For an HMSC H, the language of HMSC H, $L(H)$, is the union of the sets of words of its member bMSCs.

III. Non-local choice and implied scenarios

In this section, we analyse the relations and differences among the two conditions for non-local choice free from [7] and [5] and CCs from [1] using examples.

A. *Conditions of non-local choice free*

According to [7], non-local choice is a branching behaviour where explicit synchronisation between processes is necessary to resolve non-determinism. That is, the next behaviour of a process P_i depends on the activity of another process P_j. An example of non-local choice originally from [7] is shown in Fig.1. Assume that after executing the $Dreq$ event, process P_1 may be the first process to decide whether to branch left. In order to implement the choice behaviour properly, the processes P_2 and P_3 must be informed about P_1's decision so that they branch accordingly. In the non-local choice free condition from [7, Theorem 7] (classic condition), non-local choice is related with explicit branching nodes and first events after the branching nodes in HMSCs. The condition is as follows.

Definition 6: (Classic condition) An MSC specification $\mathcal{M}$ is non-local choice free if and only if at each of its branching nodes, the first events in all successive bMSCs are sent by the same process.

Another condition of non-local choice free is given by Baker et al. [5]. The informal description in [5] states that non-local choice occurs when the behaviour of one process depends on some aspects of the run-time behaviour of other processes unobservable by the first process. This informal description is similar to that mentioned in [7]. However, the non-local choice definition and non-local choice free condition (Baker's condition) are as follows [5].

*Definition 7: (**Baker's condition**)* Let $\mathcal{M}$ be an MSC specification and x be an active (send) event on process P_i. A non-local choice for P_i is a triple (x, w, v) where $w, v \in pref(L(\mathcal{M}))$ such that $w|P_i = v|P_i$, but $wx \in pref(L(\mathcal{M}))$ and $vx \notin pref(L(\mathcal{M}))$. $\mathcal{M}$ is non-local choice free, if $\mathcal{M}$ contains no non-local choice.

An obvious difference is that Baker's condition does not relate the non-local choice to an explicit branching structure and the first event. The non-local choices that exist independently of explicit branching nodes and first events are *hidden non-local choice* which can be further divided into two types.

Non-local choices can exist without explicit branching nodes since choice can be hidden. *Hidden choice* occurs when two member bMSCs of an MSC specification share the same prefix on a participating process but the next events on the same process are different and this local choice is not introduced by an explicit branching node. Consider the example in Fig.2, assuming that M_1 and M_2 are two member bMSCs in an MSC specification. There is no explicit branching structure for the specification, but, after receiving a, process P_2 needs to decide whether to send b or c to different processes. That is, there is a choice on process P_2 which is hidden in the two bMSCs. For a better illustration, the specification can also be described by HMSC H_1 in the middle, where an explicit branching structure is used. Consequently, non-local choice can be hidden. Consider the MSC specification consisting of two bMSCs, M_1 and M_2, originally given in [1] shown in Fig.3. P_1 and P_4 seek to perform updates on data used by a plant. P_2 controls the amount of uranium at the plant, and P_3 controls the amount of nitric acid. $inc.*$ is a request to increase the fuel amount by one unit, while $double.*$ is to double the fuel amount. As described in the two bMSCs, process P_2 and P_3 accept $inc.*$ and $double.*$ in either order. The two scenarios then form a choice behaviour because the first event happens on P_2 and P_3 decides the partial order between the remaining events on both processes. As shown in H_1 in the same figure, if the $inc.*$ happens first in either P_2 or P_3, $inc.*$ should also happen first in the other process. Conversely, if the first event on either P_2 or P_3 is $double.*$, $double.*$ should also happen first on the other process.

The other type of hidden non-local choice appears in branching structures but the first events do not form the actual non-local choices and the real non-local choices are after the first events. Fig.4 shows a non-local choice example from [5] and it is captured by Baker's condition. Consider the MSC specification described by H_1. The decision by P_3 regarding whether to send d depends on what P_1 has sent to P_2, but the choice of P_1 is not observable by P_3. The non-local choice shown in Fig.4 cannot be captured by the

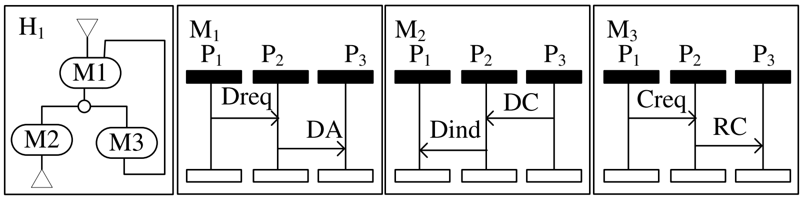

Figure 1. An example of classic non-local choice

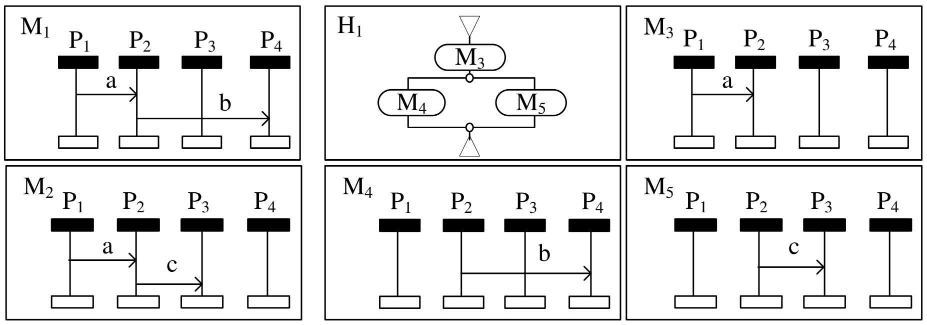

Figure 2. An example of hidden choice

classic condition, because $!e$ and $!d$ after $?c$ both happen on P_3 in the branching structure.

It is clear that detecting hidden non-local choices cannot depend on branching structures, so there may be an approach following Baker's condition.

B. Implied scenarios and Baker's condition

According to [1], [24], the synthesised model from an MSC specification may exhibit additional behaviours which are not described by the original specification. The additional behaviours are implied scenarios. Consider again the example in Fig.3. In M_1 and M_2, after transactions, equal amounts of uranium and nitric acid are supplied to the plant. However, the synthesised model based on M_1 and M_2 also exhibits the behaviour depicted by M_3 which results in unequal amounts of uranium and nitric acid. This is because, as far as each process can locally tell, M_3 follows one of the two given scenarios.

MSC specification $\mathcal{M}$ with a finite number of member bMSCs is considered to be *safely realisable* if there exists a synthesised model whose behaviour contains no implied scenarios and deadlocks [1]. According to [1], checking safe realisability is reduced to checking whether $L(\mathcal{M})$ satisfies two CCs. This means that if $\mathcal{M}$ does not satisfy the following two CCs, implied scenarios exist.

*Closure Condition 8: (**CC3**)*[4] Given a well-formed word w, if for all P_i, $1 \leq i \leq n$, there exists a word $v^i \in pref(L(\mathcal{M}))$ such that $w|P_i = v^i|P_i$, then w is in $pref(L(\mathcal{M}))$.

[4]The numbers of CCs follow [1].

*Closure Condition 9: (**CC2'**)* Given a well-formed and complete word $w \in pref(L(\mathcal{M}))$, if for all P_i, $1 \leq i \leq n$, there exists a word $v^i \in L(\mathcal{M})$ such that $w|P_i = v^i|P_i$, then w is in $L(\mathcal{M})$.

Note that CC3 can be rewritten as CC3' to make it easier to check [1]:

*Closure Condition 10: (**CC3'**)* For all $w, v \in pref(L(\mathcal{M}))$, if $w|P_i = v|P_i$ for some process P_i, $wx \in pref(L(\mathcal{M}))$ and vx is well-formed for some x on P_i, then vx is in $pref(L(\mathcal{M}))$.

CC3' means that considering two possible prefixes of $\mathcal{M}$, w, v, there is no way to distinguish them from the view P_i; if x is the next event on P_i following w, and x is not a continuation in the context of v, then vx is an implied scenario.

Comparing CC3' with Baker's condition, the only difference is that Baker's condition requests that x should be a send event but CC3' requires that vx is well-formed. It seems that different studies have been addressing the same problem from different perspectives. For example, the example in Fig.4 can also be used to explain CC3'. For process P_3, there is no difference between branching behaviour until the receiving of c; consequently, sending of d should be a possible behaviour for the left-hand side branch when process P_1 sends a to process P_2. However, this behaviour is not described by H_1, so the MSC specification violates CC3' and thus there are implied scenarios. Accordingly, we have the following proposition.

Proposition 11: Let us suppose $\mathcal{M}$ is an MSC specification. If $\mathcal{M}$ does not satisfy Baker's condition then $\mathcal{M}$

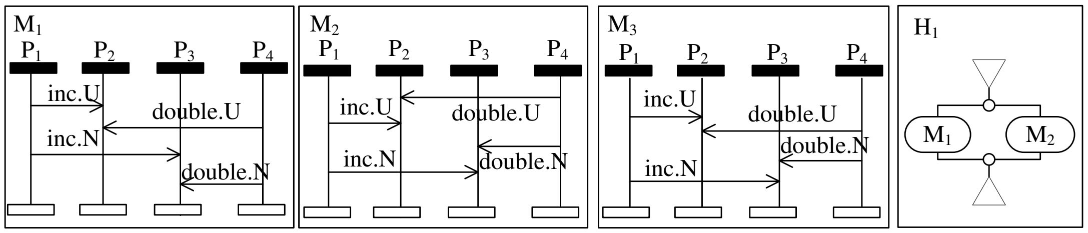

Figure 3. An example of implied scenarios

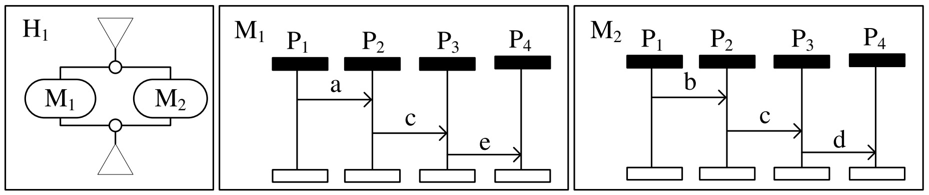

Figure 4. An example of non-local choice that satisfies Baker's condition.

contains implied scenarios and it is not the case that if $\mathcal{M}$ contains implied scenarios then $\mathcal{M}$ does not satisfy Baker's condition.

The example shown in Fig.3 illustrates the aspect that Baker's condition does not consider: choice between two receive events can be non-local choice. After the branching node in H_1, the next events on P_2 or P_3 are all receive events, but the choices on P_2 and P_3 depend on each other. Therefore, H_1 contains a non-local choice but is not covered by Baker's conditions.

According to [1], CC3' is not sufficient for safe realisability. CC2' has to be applied for some special implied scenarios. For example, H_1 shown in Fig.5 contains implied scenarios captured by CC2'. M_3 is implied by H_1 because the prefixes of M_3 are prefixes of M_1 and M_2 . In addition, the decision as to whether P_1 sends the message a forms a non-local choice, since P_1 needs to know whether P_3 sent b. This non-local choice is not captured by CC3' because no sending events will be prohibited by send events that have already happened. It is also clear that the example does not satisfy Baker's condition either.

These examples show that when receive events and terminations of processes appear in non-local choices, Baker's condition is insufficient.

IV. Deriving non-local choice definitions from CCs

We showed that both classic and Baker's conditions do not cover all non-local choices. We have also realised that every implied scenario in conditions of safe realisability indicates the existence of non-local choice, and vice-versa. In this section, we construct a new non-local choice free condition from CC3' and CC2' step-by-step.

Due to the similarity between Baker's condition and CC3', we attempt to follow Baker's lead where a non-local choice is defined as a triple as in Definition 7. The naïve definition of non-local choice can then be defined as follows.

Definition 12: ***(Naïve Non-local Choice 1)*** Let $\mathcal{M}$ be an MSC specification, a non-local choice of P_i is a triple (x, w, v) where x is a label on P_i, $w, v \in pref(L(\mathcal{M}))$ such that $w|P_i = v|P_i$, $wx \in pref(L(\mathcal{M}))$ whereas vx is a well-formed word and $vx \notin pref(L(\mathcal{M}))$.

Definition 12 covers more non-local choices than Baker's condition since the only restriction is vx being a well-formed word and so x can be a receive label; x must be a send label in Baker's condition. However, Definition 12 still overlooks non-local choices captured by CC2', for example, the non-local choice shown in Fig.5. It might appear that we should merge Definition 12 and CC2', but this is infeasible since CC2' cannot be represented by triples given in Definition 12.

In fact, CC2' deals with choices in which one option is an event label and the other is the termination of the process. Therefore, if our formalisation allows us to explicitly express the behaviour of choosing a termination, we may solve the merging issue. Therefore, we propose including the terminations of all processes into the alphabet of MSC languages. The semantics of a termination is that the termination is the end of a process in a complete run and the process cannot include further events after the termination. The termination of P_i is represented as $\downarrow_i$. For an MSC specification $\mathcal{M}$, there is a set of terminations, $\downarrow_{\mathcal{P}} = \{\downarrow_i \mid 1 \leq i \leq n\}$. An MSC language $L(\mathcal{M})$ thus corresponds to an extended language $L'(\mathcal{M})$ over alphabet $\mathcal{A} \cup \downarrow_{\mathcal{P}}$.

To generate a non-local choice definition from CC3 and CC2', we first combine the two CCs using the extended

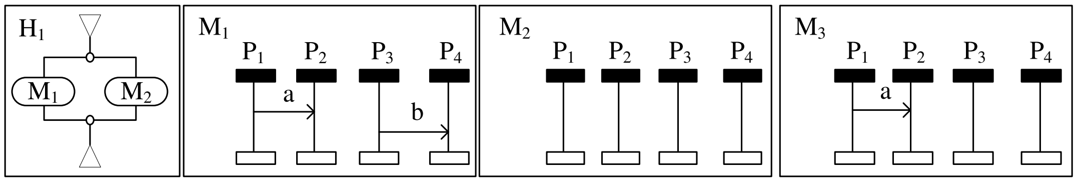

Figure 5. An example of implied scenarios satisfied by CC2'

MSC language. This can be achieved by first proving the following proposition.

Proposition 13: Given an MSC specification $\mathcal{M}$, $L(\mathcal{M})$ is safely realisable if and only if $L'(\mathcal{M})$ is safely realisable.

Alur's CC3 and CC2' may now be revised to CC4 as follows.

Closure Condition 14: **CC4** Given an MSC specification $\mathcal{M}$, $L'(\mathcal{M})$ is considered to satisfy CC4 if and only if for all well-formed w, if for each process P_i there is a word $v^i \in pref(L'(\mathcal{M}))$ such that $w|P_i = v^i|P_i$, then $w \in pref(L'(\mathcal{M}))$.

CC4 is similar to Alur's CC3 except that $L(\mathcal{M})$ is replaced by $L'(\mathcal{M})$. However, we show that CC4 also implies CC2'. This can be explained using an example. Reconsider the examples in Fig.5. M_2 represents a word $\downarrow_1\downarrow_2\downarrow_3\downarrow_4$ in $L'(\mathcal{M})$. Let us consider a word, $!a?a\ \downarrow_3$ which is well-formed and where $!a$, $?a$, $\downarrow_3$ and the empty word are all valid prefixes for each process. However, $!a?a\ \downarrow_3$ is not a valid prefix of the language, formed by M_1 and M_2 because there is no event that can participate in P_3 after $\downarrow_3$. Therefore, the MSC specification represented by H_1 does not satisfy CC4. Formally, the relation between CC3, CC2' and CC4 is as follows.

Proposition 15: Given an MSC specification $\mathcal{M}$, $L(\mathcal{M})$ satisfies CC3 and CC2' if and only if the corresponding language $L'(\mathcal{M})$ satisfies CC4.

Proposition 15 means that CC4 can be used to replace CC3 and CC2'. In addition, we have following proposition.

Proposition 16: Given an MSC specification $\mathcal{M}$, $L(\mathcal{M})$ is safely realisable if and only if the corresponding language, $L'(\mathcal{M})$ on alphabet $\mathcal{A} \cup \downarrow_{\mathcal{P}}$, contains well-formed and complete words and satisfies CC4.

With CC4, we may give a definition for implied scenario under the condition of safe realisability as follows.

*Definition 17: (**Implied Scenarios**)* Given an MSC specification $\mathcal{M}$, w represents an *implied scenario* of $\mathcal{M}$ if w is a well-formed word and for each $w|P_i\ i \in [n]$, a word $v \in pref(L'(\mathcal{M}))$ exists such that $w|P_i = v|P_i$, but $w \notin pref(L'(\mathcal{M}))$.

In a similar way to Alur et al. [1], we can transform CC4 to a form similar to CC3' which is easier to be transformed into a definition of non-local choices.

Closure Condition 18: (**CC5**) Given an MSC specification $\mathcal{M}$, $L'(\mathcal{M})$ satisfies CC5 if and only if for all $w, v \in pref(L'(\mathcal{M}))$ such that $w|P_i = v|P_i$, if $wx \in pref(L'(\mathcal{M}))$ and vx is well-formed for some $x \in \mathcal{A}_i \cup \downarrow_i$, then $vx \in pref(L'(\mathcal{M}))$.

A language satisfies CC3 if and only if it satisfies CC3' [1], and we have a similar proposition.

Proposition 19: Given an MSC specification $\mathcal{M}$, $L'(\mathcal{M})$ satisfies CC4 if and only if it satisfies CC5.

Based on the CC5 which has unified power of CC2' and CC3, another non-local choice definition can be derived from the negation of CC5. Intuitively, it covers all non-local choices in an MSC specification.

*Definition 20: (**Naïve Non-local Choice 2**)* Given an MSC specification $\mathcal{M}$, a non-local choice is a triple (x, w, v) where x is an event label on process P_i, $w, v \in pref(L'(\mathcal{M}))$ such that $w|P_i = v|P_i$ and $wx \in pref(L'(\mathcal{M}))$ whereas vx is a well-formed word and $vx \notin pref(L'(\mathcal{M}))$.

Consider again the example in Fig.5. For P_1, there is a choice between $!a$ and $\downarrow_{P_1}$; for P_3 there is a choice between $!b$ and $\downarrow_3$. $!a?a$ and $\downarrow_1\downarrow_2$ are two valid prefixes from M_1 and M_2. After one of these two prefixes, P_3 needs to decide whether to send b. However, P_3 has the same sequence of observations for these two prefixes. If $!a?a$ is the actual execution, then $!a?a\ \downarrow_3$ is not a valid prefix of the extended language of H_1, according to Definition 20, $(\downarrow_3, \downarrow_1\downarrow_2, !a?a)$ is a non-local choice. In addition, if $\downarrow_1\downarrow_2$ is the actual execution, then $\downarrow_1\downarrow_2!b$ is not a valid prefix either, and $(!b, !a?a, \downarrow_1\downarrow_2)$ is also a non-local choice.

Definition 20 is a step forward, but it contains redundancy. There might be many $v \in pref(L')$ for the same w and x. For example in Fig.5, if $x = \downarrow_3$ and $w = \downarrow_1\downarrow_2$, then v can be $!a?a$ or $!a$. There are two different non-local choices $(\downarrow_3, \downarrow_1\downarrow_2, !a?a)$ and $(\downarrow_3, \downarrow_1\downarrow_2, !a)$. Also, $(\downarrow_3, \downarrow_1\downarrow_2, !a?a)$ and $(!b, !a?a, \downarrow_1\downarrow_2)$ in the example are defined as two different non-local choices, accordingly. The problem is that the three non-local choices explain one local choice issue. That is, the choice between $\downarrow_3$ and $!b$ on P_3 is decided by the behaviour on P_1. Moreover, from the triple, it is hard to learn where the choice is, why it is a non-local choice and what the options of the choice are.

To give a better definition to non-local choice, we introduce the concept of *choices* in a process. In an MSC specification, each process may display different behaviours when involved in multiple scenarios. Given the assumption that the initial states of the process are always the same, the process must choose the next behaviour after some steps of execution. Formally, a choice in extended process language can be defined as follows:

*Definition 21: **(Choice)*** Given an MSC specification $\mathcal{M}$, a *choice* on P_i is a triple (w, x, y), where $w \in pref(L'(\mathcal{M})|P_i)$, $x, y \in \mathcal{A}_i \cup \downarrow_i$ and $x \neq y$ such that $wx, wy \in pref(L'(\mathcal{M})|P_i)$.[5]

We thus derive the final definition of non-local choice and condition of non-local choice free as follows.

*Definition 22: **(Non-local Choice and Non-local Choice Free)*** Given an MSC specification $\mathcal{M}$, a *non-local choice* on P_i is a choice (w, x, y), such that there exists a word $v \in pref(L'(\mathcal{M}))$, where $v|P_i = w$, $vx \in pref(L'(\mathcal{M}))$, vy is well-formed and $vy \notin pref(L'(\mathcal{M}))$. $\mathcal{M}$ is non-local choice free, if $\mathcal{M}$ contains no non-local choice.

Based on this definition, we can show the formal relations between non-local choice and implied scenarios. They are stated in the following propositions.

Proposition 23: Given an MSC specification $\mathcal{M}$, a language $L(\mathcal{M})$ is safely realisable if and only if $L'(\mathcal{M})$ is non-local choice-free.

Proposition 24: Given an MSC specification $\mathcal{M}$, a language $L'(\mathcal{M})$ does not satisfy CC5 if and only if the corresponding MSC language $L(\mathcal{M})$ contains non-local choices.

V. Algorithm for detecting non-local choice

Based on the results of the previous section, we can extend the algorithm in [1] for checking CC3' to detect non-local choices in MSC specifications with a finite number of member bMSCs. The rationale for the algorithm is based on Proposition 24 and the following proposition.

Proposition 25: If an implied scenario exists in an MSC specification, then it corresponds to a non-local choice in that specification.

We have shown that the motivation behind CC3' is that it can be used to check whether there is an implied scenario in an MSC specification. Note that CC5 is similar to CC3', and CC5 can therefore be used for the same purpose. However, the difference is that CC3' does not cover implied scenarios captured by CC2', but CC5 does. It is therefore possible to extend the algorithm for checking CC3' to one checking CC5. The extended algorithm has the same power as the two algorithms in [1] for checking CC2' and CC3'.

The extended algorithm (Algorithm 1 in the Appendix) identifies all non-local choices in an MSC specification formed by a finite set $\mathcal{M} = \{M_1 \dots M_k\}$ of k member bMSCs. The main differences between Algorithm 1 and the algorithm for checking CC3 in [1] are: Algorithm 1 continues to loop after finding the first implied scenario so that it detects all non-local choices; when checking whether event label x can be replaced by x' (line 12 of Algorithm 1), terminations are considered.

[5]We consider choices with more than three options as a group of choices, each of which has only two options. This method simplifies the definition of non-local choice and the algorithm for detecting it.

VI. Conclusions and future work

In this paper, we have investigated two different conditions for non-local choice free with recourse to the relevant literature. Our analysis shows that previous conditions may be incomplete in terms of the informal definition of non-local choices. For example, hidden non-local choice in MSC specifications cannot be identified using the classic condition, since the classic condition is defined based on explicit branching structures and first events in HMSCs.

The similarity between Alur et al.'s CCs for safe realisability and Baker's condition has inspired us to investigate the relation between the two. Some examples are used to show that Baker's condition is not complete regarding the non-local choices containing receive events or terminations. It transpires that Alur's CCs for implied scenarios can capture all those non-local choices.

Nevertheless, the fact that there are two items in Alur et al.'s CCs is an obstacle to developing a definition for non-local choice. To resolve the issue, we introduced termination on each participating process as elements of the overall alphabet. This modification allowed us to combine the two CCs into one. According to the new CC, a new definition of non-local choice and a new condition for non-local choice free are given; the equivalence between identifying the implied scenarios and non-local choice are proven and, furthermore, the two parts of Alur et al.'s algorithm are unified and modified. Algorithm 1 is thus proposed to identify all non-local choices in MSC specifications with a finite number of member bMSCs.

Planned future work includes the following. The provided algorithm only applies to MSC specifications with a finite number of bMSCs, but when loop structures are contained in an MSC specification, there might be an infinite number of member bMSCs. It has been shown that safe realisability of HMSCs is a PSPACE-hard problem [2], but it seems that this can be simplified by adding restrictions. For example, restricting the number of unfoldings of the loops in MSC specifications. Thus it would be interesting to develop a practical algorithm for detecting non-local choices in MSC specifications with loops. In addition, compared to MSCs, Interaction Diagrams (IDs) of UML 2.0 are a more popular modelling language in the software industry. It would be worth extending current work to IDs. However, it has been shown that there are significant differences between IDs and MSCs [9]. These differences should be handled with care in the migration. Finally, it seems a promising approach to use the thread-tag based semantics [8] of IDs to ensure the correctness of this migration.

References

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

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

[3] R. Alur, G. Holzmann, and D. Peled. An analyzer for message sequence charts. *Software Concepts and Tools*, 17(2):70–77, 1996.

[4] R. Alur and M. Yannakakis. Model checking of message sequence charts. In *Proceedings of 10th International Conference on Concurrency Theory*, pages 114–29, Eindhoven, Netherlands, 8 1999.

[5] Paul Baker, Paul Bristow, Clive Jervis, David King, Robert Thomson, Bill Mitchell, and Simon Burton. Detecting and resolving semantic pathologies in UML sequence diagrams. In *Proceedings of the 10th European Software Engineering Conference held jointly with the 13th International Symposium on Foundations of Software Engineering*, pages 50–59, Lisbon, Portugal, 2005.

[6] H. Ben-Abdallah and S. Leue. Timing constraints in message sequence chart specifications. In *Proceedings of International Conference on Formal Description Techniques: Protocol Specification, Testing and Verification*, pages 91–106, Osaka, Japan, 11 1997.

[7] Hanêne Ben-Abdallah and Stefan Leue. Syntactic detection of process divergence and non-local choice in message sequence charts. In *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 259–274, Enschede, Netherlands, 1997.

[8] Haitao Dan, Robert M. Hierons, and Steve Counsell. A Thread-tag based semantics for Sequence Diagram. In *the 5th International Conference on Software Engineering and Formal Methods*, pages 173–182, London, UK, 2007.

[9] Haitao Dan, Robert M. Hierons, and Steve Counsell. Thread-based analysis of Sequence Diagrams. In *the 27th International Conference on Formal Methods for Networked and Distributed Systems*, pages 19–34, Tallinn, Estonia, 2007.

[10] Øystein Haugen. MSC-2000 interaction diagrams for the new millennium. *Computer Networks*, 35(6):721–732, 2001.

[11] Loïc Hélouët. Some pathological message sequence charts, and how to detect them. In *Proceedings of the 10th International SDL Forum*, pages 348–364, Copenhagen, Denmark, 1 2001.

[12] ITU-T. ITU-T Recommendation Z.120 Message Sequence Chart, 4 2004.

[13] S. Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, 34:3–42, 1956.

[14] P. B. Ladkin and S. Leue. What do message sequence charts mean? In *Proceedings of the IFIP TC6/WG6.1 Sixth International Conference on Formal Description Techniques*, pages 301–316, Boston, USA, 1993.

[15] P. B. Ladkin and S. Leue. Interpreting message flow graphs. *Formal Aspects of Computing*, 7(5):473–509, / 1995.

[16] Peter B. Ladkin and Stefan Leue. Four issues concerning the semantics of Message Flow Graphs. In *Proceedings of the 7th International Conference on Formal Description Techniques*, pages 355–369, Berne, Switzerland, 1994.

[17] Emmanuel Letier, Jeff Kramer, Jeff Magee, and Sebastian Uchitel. Monitoring and control in scenario-based requirements analysis. In *Proceedings of the 27th international conference on Software engineering*, pages 382–391, New York, USA, 2005.

[18] S. Mauw, M.A. Reniers, and T.A.C. Willemse. Message Sequence Charts in the software engineering process. *Handbook of Software Engineering and Knowledge Engineering*, 1:437–464, 2001.

[19] B. Mitchell. Resolving race conditions in asynchronous partial order scenarios. *IEEE Transactions on Software Engineering*, 31(9):767–784, 2005.

[20] Arjan J. Mooij, Nicolae Goga, and Judi Romijn. Non-local choice and beyond: Intricacies of MSC choice nodes. In Maura Cerioli, editor, *FASE*, volume 3442 of *Lecture Notes in Computer Science*, pages 273–288. Springer, 2005.

[21] H. Muccini. Detecting Implied Scenarios Analyzing Non-local Branching Choices. In *Proceedings of 6th International Conference on Fundamental Approaches to Software Engineering*, pages 372–386, Warsaw, Poland, 2003.

[22] Anca Muscholl and Doron Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. In *Proceedings of the 24th International Symposium on Mathematical Foundations of Computer Science*, pages 81–91, London, UK, 1999.

[23] SDL Forum Society. http://www.sdl-forum.org/, Accessed in 2008.

[24] S. Uchitel, J. Kramer, and J. Magee. Incremental elaboration of scenario-based specifications and behavior models using implied scenarios. *ACM Transactions on Software Engineering and Methodology*, 13(1):37–85, 1 2004.

Appendix: Proofs and Algorithm

Proof of Proposition 3

Proof: Given an MSC specification $\mathcal{M}$, the process language of P_i is $L'(\mathcal{M})|P_i$. $L'(\mathcal{M})|P_i$ are always regular according to Kleene closure [13], because in constructing $L'(\mathcal{M})|P_i$, we can only use sequencing, branching and iteration. ■

Proof of Proposition 13

Proof: First, let us suppose that L is safely realisable. There exists a set of deterministic automata A_i such that $L(\mathcal{M}) = L(A)$ for $A = \prod A_i$. A can only accept well-formed and complete words and is deadlock-free. Next, we construct A'_i by adding an accepting state to A_i, changing the original accepting states of A_i to normal states then adding transitions from the original accepting states of A_i to the new accepting state with label $\downarrow_i$. Therefore, $L(A'_i) = \{w \downarrow_i | w \in L(A_i)\}$. After that, we construct $A' = \prod A'_i$. According to the construction, A' is also deadlock-free and $L(A') = L'(\mathcal{M})$. Thus, $L'(\mathcal{M})$ is safely realisable.

Conversely, we use the same technique. Since $L'(\mathcal{M})$ is safely realisable, there exists a set of deterministic automata A'_i such that $L'(\mathcal{M}) = L(A')$ for $A' = \prod A'_i$. A' can only accept well-formed and complete words and is deadlock-free. We construct A_i by removing the accepting states from A'_i, changing the normal states which originally connected to the old accepting states to be accepting states themselves. The construction removes all transitions with label $\downarrow_i$. Therefore, $L(A_i) = \{w[1 \ldots (\|w\| - 1)] | w \in L(A_i)\}$, where $\|w\|$ denotes the length of the string w and $w[1 \ldots j]$ represents a prefix formed by the first j letters of string w. Let $A = \prod A_i$. According to the construction, A is still deadlock-free since each process will reach the new accepting states, and $L(A) = L(\mathcal{M})$. Thus, $L(\mathcal{M})$ is safely realisable. ∎

Proof of Proposition 15

Proof: We know $L(\mathcal{M})$ satisfies CC3 and CC2' if and only if $L(\mathcal{M})$ is safely realisable. Then, to prove the above proposition, it is enough that $L(\mathcal{M})$ is safely realisable if and only if the corresponding $L'(\mathcal{M})$ satisfies CC4. According to Proposition 13, $L(\mathcal{M})$ is safely realisable if and only if $L'(\mathcal{M})$ is safely realisable. Thus, it is sufficient to prove that $L'(\mathcal{M})$ is safely realisable if and only if $L'(\mathcal{M})$ satisfies CC4.

Let us suppose that $L'(\mathcal{M})$ is safely realisable. There exist deterministic automata A'_i such that $L'(\mathcal{M}) = L(A')$ for $A' = \prod A'_i$. A' can only accept well-formed and complete words and is deadlock-free. We show that $L'(\mathcal{M})$ satisfies CC4. Consider w as a well-formed word on alphabet $\mathcal{A} \cup \downarrow_{\mathcal{P}}$. For each P_i, let $v^i \in L'(\mathcal{M})$ be a word such that $w|P_i = v^i|P_i$. Each $v^i|P_i$ is a prefix of a possible run of A'_i. The combination of $v^i|P_i$ forms a prefix run of A'. Therefore $w \in pref(L'(\mathcal{M}))$. Thus, CC4 is satisfied.

Conversely, let us suppose that $L'(\mathcal{M})$ satisfies CC4. We construct A' according to $L'(\mathcal{M})$ using process languages. According to Proposition 3, process languages are regular, so the set of $L'(\mathcal{M})|P_i$ can be accepted by a set of deterministic automata A'_i. A' is constructed as $A' = \prod A'_i$. Next, we need to show that A' is deadlock-free and $w \in L'(\mathcal{M})$ if and only if $w \in L(\prod A'_i)$. During the execution of $\prod A'_i$, at all times, the word w has the property that its projection onto each process P_i belongs to $pref(L'(\mathcal{M})|P_i)$. By CC4, w is in $pref(L'(\mathcal{M}))$.

For the deadlock-free part, since A'_i are deterministic, the product automaton must be able to reach an accepting state after processing $w \in pref(L'(\mathcal{M}))$, simply by processing the word $w' \in L'(\mathcal{M})$ such that w is a prefix of w'. Hence $\prod A'_i$ is deadlock-free.

To prove $w \in L'(\mathcal{M})$ if and only if $w \in L(\prod A'_i)$, the forward direction follows from the construction of A'_i. Let us suppose that $w \in \prod A'_i$, then for each process P_i, $w|P_i \in L'(\mathcal{M})|P_i$. According to the construction of the word in $L'(\mathcal{M})$, every word in L'_i ends in $\downarrow_i$; therefore, no other events can participate. Since we already know $w \in pref(L'(\mathcal{M})|P_i)$ and A' is deadlock-free, $w \in L'(\mathcal{M})$. ∎

Proof of Proposition 19

Proof: Let us suppose that $L'(\mathcal{M})$ satisfies CC4 and $v, w \in pref(L'(\mathcal{M}))$ such that $v|P_i = w|P_i$, and $wx \in pref(L'(\mathcal{M}))$. Thus, vx has the property that $vx|_j = v|_j$ for $j \neq i$, and $vx|P_i = wx|P_i$. If vx is well-formed, then by CC4, $vx \in pref(L'(\mathcal{M}))$. Therefore, CC4 implies CC5 as required.

Let us suppose that $L'(\mathcal{M})$ satisfies CC5. Consider a w which is well-formed and such that for all i, $w|P_i = v^i|P_i$, where $v^i \in pref(L')$. We will, by induction on the length of w show that $w \in pref(L'(\mathcal{M}))$. If $|w| = 1$, then from CC5, using the fact that the empty string is always in $pref(L'(\mathcal{M}))$, we get that $w \in pref(L'(\mathcal{M}))$.

Further, let us suppose $w = w'x$, where x occurs on some process i. Then $v^i|P_i = (w'|P_i)x$. By the inductive hypothesis, it is clear that $w' \in pref(L'(\mathcal{M}))$. Now, w' and v^i satisfy the condition of CC5; $w'x = w$ is also in $pref(L'(\mathcal{M}))$. ∎

Proof of Proposition 23

Proof: First, let us suppose that $L(\mathcal{M})$ is safely realisable. For the corresponding $L'(\mathcal{M})$, CC4 is satisfied. We need to show that there is no non-local choice. We prove this by contradiction. Let us suppose that there is a non-local choice in $L'(\mathcal{M})$. According to Definition 22, for process P_i, there is a choice (w, x, y), and so we have that $w \in pref(L'(\mathcal{M})|P_i)$ and $x, y \in \mathcal{A}_i \cup \downarrow_i$ such that $wx, wy \in pref(L'(\mathcal{M})|P_i)$. Because $L'(\mathcal{M})|P_i$ is constructed by projecting $L'(\mathcal{M})$ on P_i, we can find a set of prefixes $W \in pref(L'(\mathcal{M}))$ such that $W|P_i = wy$. In addition, this is a non-local choice, so there must exist a word $v \in pref(L'(\mathcal{M}))$, $v|P_i = w$, $vx \in pref(L'(\mathcal{M}))$ and vy is well-formed, but $vy \notin pref(L'(\mathcal{M}))$. Since, $v \in pref(L'(\mathcal{M}))$ and $y \in \mathcal{A}_i$, $v|_j = vy|_j, j \neq i$. Further, we know $vy|P_i = wy$ and wy is a valid projection of W onto process P_i. $L'(\mathcal{M})$ satisfies CC4, therefore $vy \in pref(L'(\mathcal{M}))$. But $vy \notin pref(L'(\mathcal{M}))$ since (w, x, y) is a non-local choice. Therefore, forward direction is established.

Conversely, assume there is no non-local choice in language $L'(\mathcal{M})$, which means that for all choices, for all processes $P_i \in \mathcal{P}$, a run can select either of the options as its next event. We then need to prove that the corresponding $L(\mathcal{M})$ is safely realisable. The set of words in $L'(\mathcal{M})$

relating to some choice (w, x, y), can be partitioned into two sets U', V' where U' is the set of sequences from $L'(\mathcal{M})$ that have wx as a prefix and V' is the set sequences from $L'(\mathcal{M})$ that have vx as a prefix. Since there is no non-local choice, for any $u' \in pref(U')$, such that $u'|P_i = w$ and $u'y$ is well-formed, $u'y \in pref(L'(\mathcal{M}))$. Moreover, for any $v' \in pref(V')$ such that $v'|P_i = w$ and $v'x$ is well-formed, $v'x \in pref(L'(\mathcal{M}))$. This means that any two words in $pref(L'(\mathcal{M}))$ satisfying the assumption in CC5 can choose any next possible events, therefore CC5 is satisfied. Following the Proposition 19 and 16, L is safely realisable. ■

Proof of Proposition 25

Proof: Consider the negation of CC5; the implied scenarios appear when $wx \in pref(L'(\mathcal{M}))$, but $vx \notin pref(L'(\mathcal{M}))$, such that $w, v \in pref(L'(\mathcal{M})), w|P_i = v|P_i$, $x \in \mathcal{A}_i$ and vx is well-formed. Since $v \in pref(L'(\mathcal{M}))$, we can extend v to some $v' \in L'(\mathcal{M})$. Let us suppose that the length of $w|P_i$ is c, then there must exist an $x' \in \mathcal{A}_i$, the number $(c+1)$ event of word $v'|P_i$. According to Definition 21, the definition of choice, $(w|P_i, x', x)$ is a choice on process P_i. Moreover, for this choice and word v, $vx' \in pref(L'(\mathcal{M}))$, $vx \notin pref(L'(\mathcal{M}))$ and vx is well-formed. Therefore $(w|P_i, x', x)$ is a non-local choice. ■

```
Input: $\mathcal{M}$, a set of bMSCs
Output: Identified non-local choices.
forall $(s,t,i) \in [k] \times [k] \times [n]$ do
    $T[s,t,i] = min\{c | M[s,i,c] \neq M[t,i,c]\}$
end
/* $T[s,t,i]$ gives the first position on process i where $M_s$ and $M_t$ differ $\leq^s$ is the partial order of events in $M_s$. */
forall $s \in [k]$ and event $x$ in $M_s$ do
    for process $P_j$ $j \in [n]$ do
        $U[s,x,j] = \begin{cases} \|M[s,j]\| + 1 & if\ \forall\ c\ \ x \not\leq^s M[s,j,c] \\ min\{c | x \leq^s M[s,j,c]\} & otherwise \end{cases}$
    end
end
/* $U[s,x,*]$ gives location of the first event of $M_s$ dependent on $x$. */
forall $(s,t,j) \in [k] \times [k] \times [n]$ such that $T[s,t,j] \neq \perp$ do
    $c = T[s,t,j]$
    $x = M[s,j,c]; x' = M[t,j,c]$
    /* Determine if $x'$ is eligible to replace $x$. If $x'$ is a send event, it is always eligible. If $x'$ is a termination, it is always eligible. If $x' = receive(i,j,a)$ then $x'$ is eligible if $M[s,i][1 \dots U[s,x,i]-1]$ contains more $send(i,j,a)$'s than $M[s,j][1 \dots U[s,x,j]-1]$ contains $receive(i,j,a)$'s. */
    if $x'$ is eligible to replace $x$ then
        /* Find if some $M_p$ realises this replacement. */
        if $\exists\ p \in [k]$ such that $M[p,j,c] = x'$ and $\forall\ j' \in [n]\ U[s,x,j'] \leq T[s,p,j']$ then
            /* This eligible replacement exists, do nothing. */
        else
            /* Non-local choice identified. */
            Output: for process $P_j$, $(M[s,j][1 \dots (c-1)], x, x')$ is a non-local choice
        end
    end
end
```
Algorithm 1: Identify non-local choices