

La manipulation formelle de scénarios

L'exemple des "Message Sequence Charts"

Loïc Héluët[♦], Claude Jard[■]

[♦] France Télécom, 2 av Pierre Marzin, 22307 Lannion Cedex, France
loic.helouet@francetelecom.com

[■] IRISA/CNRS,
Campus de Beaulieu, 35042 Rennes Cedex
claude.jard@irisa.fr
<http://www.irisa.fr/pampa>

RÉSUMÉ. Les MSC (pour "Message Sequence Charts") sont un langage normalisé par l'ITU. D'abord introduits dans le contexte du langage SDL et le monde des télécommunications pour représenter les traces d'exécution de protocoles, ce langage a évolué considérablement et présente maintenant une proposition crédible pour la description de scénarios au sein d'environnements de conception UML. Les scénarios trouvent alors une place à l'intérieur du processus de conception en intervenant assez tôt dans la formalisation des exigences. Cet article présente un état de l'art en matière de manipulation formelle de ces scénarios, en se focalisant sur les points d'intérêt pour la modélisation des systèmes réactifs : la sémantique du parallélisme, la vérification de propriétés, la simulation et la synthèse d'implantations.

ABSTRACT. The MSCs (for Message Sequence Charts) are a scenario language, standardised by ITU. They were first introduced associated with the SDL specification language in the telecommunication area to represent execution traces. They have now evolved towards an interesting proposal to describe scenarios in a UML design environment. Scenarios can be used early in the development process by a formal capture of requirements. This article presents a survey on formal manipulation of these scenarios, focusing on some aspects of modelisation of reactive systems: semantics of concurrency, verification, simulation and implementation synthesis.

MOTS-CLÉS : MSC, scénarios, diagrammes de séquence, état de l'art

KEYWORDS: MSC, scenarios, sequence diagrams, survey

1. Introduction

Un des représentations les plus utilisées pour décrire informellement le comportement d'entités et les communications dans un système distribué est le chronogramme. L'écoulement du temps sur chaque processus est symbolisé par un trait vertical, les messages par des flèches de l'émetteur vers le récepteur. Les "Message Sequence Charts" (ou MSC) sont une proposition de formalisation de ce type de dessin. Initialement, les MSC n'étaient utilisés que pour représenter les traces d'exécution des systèmes distribués spécifiés en SDL. L'idée de normaliser ce langage graphique s'est rapidement imposée, et un groupe d'étude de l'ITU a été chargé de la question. Le premier document produit par le groupe d'étude 10 de l'ITU définissait les éléments du langage MSC'92. A quelques éléments près, les MSC'92 sont les bMSC présentés dans l'article. La période d'étude suivante fut consacrée à la sémantique du langage, ainsi qu'à son extension. C'est ainsi que fut proposé en 1996 le langage MSC'96, défini par un document de norme (la recommandation Z.120 [ITU 99]). Ce document comporte, en plus des MSC'92, la définition d'opérateurs de composition. L'ITU travaille actuellement sur une nouvelle version de la norme Z.120, connue sous le nom de MSC'2000. Elle devrait introduire des manipulations de données dans les MSC, en faisant presque un langage de programmation!

Nous voyons plusieurs points d'intérêt pour la communauté "Modélisation des systèmes réactifs" (MSR) : 1/ Il y a un besoin fort pour outiller les phases amont de la conception de logiciels réactifs et répartis. L'utilisation de scénarios est une pratique naturelle qui ne demande qu'à être formalisée. 2/ Poussés à l'origine par des considérations très pratiques de l'ingénieur, les MSC ont atteint une maturité qui permet de les équiper d'un modèle formel qui s'avère petit à petit très riche et puissant. Plusieurs théoriciens du monde de la validation et des méthodes formelles s'y intéressent et une vraie dynamique scientifique vient de se former et commence à engranger des résultats intéressants. Le corpus d'articles scientifiques sur le sujet s'étoffe (on en a recensé environ une centaine). Nous considérons utile d'en extraire une première synthèse. 3/ Naturellement consacrés à la spécification formelle de comportements répartis, les MSC ravivent un intérêt pour les sémantiques de vrai parallélisme à base de manipulation de familles d'ordres partiels. 4/ De nombreuses perspectives sont ouvertes, notamment sur les aspects méthodologiques : traitement de spécifications partielles, méthodes de raffinement des comportements, synthèse de squelettes de code réparti, vérification de cohérence, analyse temporisée. Ces différents aspects sont au cœur de la thématique MSR, avec une interaction potentielle et souhaitée entre la culture informatique et automatique.

Après la définition précise des MSC, le reste de l'article sera consacré aux différentes manipulations formelles qui peuvent être conduites. La première discussion portera sur la sémantique d'ordre partiel sous-jacente et la puissance qu'elle offre dans la représentation de familles complexes de comportements. Nous regarderons ensuite les questions de vérification de propriétés et notamment l'examen des classes décidables et indécidables connues. Enfin, nous aborderons les questions d'implantation des MSC en des systèmes d'automates communicants afin de passer graduellement

de l'expression des besoins à une conception plus détaillée. La dernière partie de l'article est consacrée aux aspects temps et performance, avant de conclure sur quelques langages apparentés.

2. Présentation des "Message Sequences Charts"

2.1. Diagrammes de base

Un diagramme de base (bMSC par la suite) décrit le comportement de processus appelés *instances* qui communiquent de manière asynchrone. Les instances d'un bMSC sont représentées par un axe vertical, et les échanges de messages par des flèches de l'instance émettrice vers l'instance réceptrice, étiquetées par un nom de message. Outre les communications, un bMSC peut contenir des actions atomiques, des opérations sur des horloges (armement, désarmement, expiration). La figure 1 illustre la notation graphique associée à chaque élément du langage et donne un exemple de bMSC.

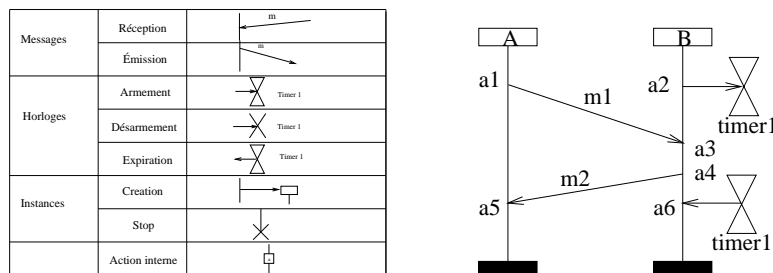


Figure 1. Eléments de bMSC et premier exemple

Les communications entre instances sont supposées asynchrones. La nature des canaux de communication assurant les transferts de messages n'est pas plus précise. L'occurrence d'une action est appelée un *événement*. Un bMSC définit des causalités entre événements : le long de l'axe d'une instance, les événements sont ordonnés de haut en bas. De plus, l'émission d'un message précède sa réception (ordre causal). L'ensemble des comportements décrits par un bMSC est un ensemble de suites d'actions respectant l'ordre causal. L'ordre causal peut cependant être relaxé sur une zone particulière des axes d'instances appelée corégion, qui se signale par un segment en traits pointillés. Sur l'exemple de la figure 2, les événements *a1* et *a3* ne sont pas ordonnés sur l'axe de l'instance *A*.

2.2. Composition de diagrammes

Les bMSC ne permettent que la définition de scénarios très simples. Le formalisme MSC'96, également appelé "High-level Message Sequence Charts" (HMSC) a

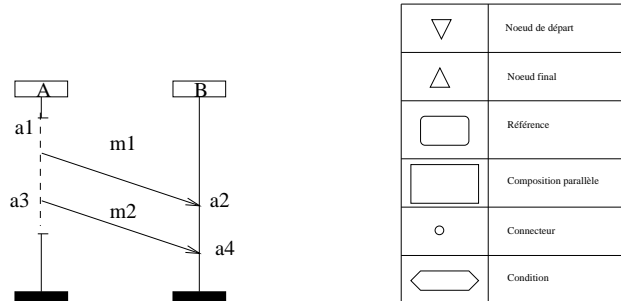


Figure 2. Un bMSC contenant une corégion et éléments d'un HMSC

été conçu pour permettre la création de scénarios plus complexes, autorisant des compositions parallèles, des alternatives, des itérations, des séquences de bMSC, ainsi que la définition hiérarchique de scénarios. Un HMSC est un graphe construit à partir des éléments de la figure 2 (à droite). Nous décrivons rapidement les principaux éléments apparaissant dans les HMSC. Pour une présentation plus complète du langage, le lecteur intéressé pourra se reporter à [REN 98, RUD 96].

2.2.1. Composition parallèle

La composition parallèle entre deux MSC permet de définir un comportement concurrent de deux scénarios. La composition parallèle de M_1 et M_2 dans la figure 3 admet tous les entrelacements des comportements définis par M_1 et M_2 . Lorsque deux bMSC définissent le comportement d'ensembles d'instances disjoints, leur composition parallèle équivaut à l'union des bMSC composés. Par exemple, le HMSC de la figure 3 est équivalent au HMSC de la figure 4.

2.2.2. Composition Séquentielle

La composition séquentielle de deux spécifications H_1 et H_2 définit une relation de précedence entre les événements de H_1 et les événements de H_2 situés sur une même instance.

Lorsque les spécifications composées définissent des comportements sur des ensembles d'instances disjoints, la composition séquentielle est alors équivalente à la composition parallèle. Une séquence de bMSC peut donc introduire du parallélisme. La sémantique des bMSC est assez intuitive. Il n'en est pas de même pour les HMSC : deux visions de la composition séquentielle de bMSC ont été définies. La première, appelée **composition séquentielle forte**, impose que tous les événements d'un bMSC soient exécutés avant qu'un événement du bMSC suivant ne soit exécuté. La deuxième approche, appelée **composition séquentielle faible** considère qu'une séquence de bMSC n'impose pas de synchronisation, et que la relation de précedence causale est tout simplement "recollée" le long des instances communes.

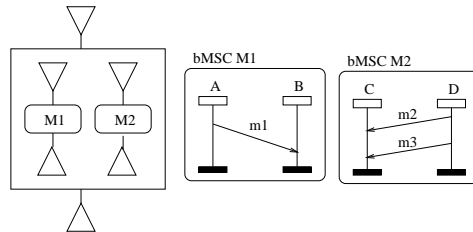


Figure 3. Composition parallèle de MSC

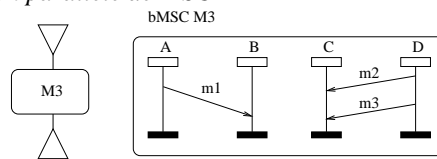


Figure 4. Équivalence

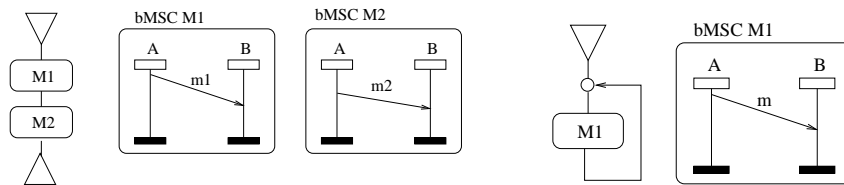


Figure 5. Séquence de bMSC

Composition séquentielle forte : la séquence $M1; M2$ de l'exemple de la figure 5 ne décrit qu'une seule trace, $!m1; ?m1; !m2; ?m2$ ($!m$ pour l'événement d'émission du message m , $?m$ sa réception). L'hypothèse de composition séquentielle forte restreint fortement le pouvoir d'expression des HMSC. En effet, si l'on impose une synchronisation entre deux bMSC, alors un HMSC ne décrit plus qu'une concaténation des linéarisations des bMSC. Les systèmes décrits par des HMSC se réduisent à des machines à nombre d'états fini, construites en sérialisant les automates associés à chaque bMSC, sur lesquelles des opérations classiques de model-checking ou d'étude temporelle peuvent être effectuées. Si cette hypothèse peut se justifier dans certains contextes, elle semble cependant peu adaptée à la description de systèmes distribués asynchrones.

Composition séquentielle faible : l'approche "officielle" de la séquence est la composition séquentielle faible, qui consiste à considérer qu'un événement peut être exécuté dès que ses prédécesseurs sur la même instance ont été exécutés. Cette composition correspond à la concaténation locale définie dans [PRA 86] sur les ordres, et plus tard dans [REN 94] sur des algèbres de processus. La séquence $M1; M2$ décrit donc

deux traces : $!m_1;?m_1;!m_2;?m_2$, et $!m_1;!m_2;?m_1;?m_2$. Du coup, les HMSC ne décrivent plus de systèmes de transitions finis, puisqu'il est possible d'itérer un envoi de message sans que la réception correspondante n'ait été effectuée, et donc d'avoir une taille des canaux de communication non bornée (exemple de la figure 5 à droite). L'adoption de la composition séquentielle faible n'implique pas nécessairement que les canaux ne soient pas bornés : des acquittements peuvent assurer cette propriété sans pour autant qu'une synchronisation arbitraire n'ait été utilisée. Une analyse de la spécification peut facilement montrer la (non)bornitude des canaux (voir chapitre suivant). Cette situation peut être résolue par l'ajout de synchronisations ad-hoc ou de contraintes temporelles.

2.3. Alternative

L'alternative permet de définir un choix entre deux scénarios possibles à un instant donné. Ce choix est exprimé au moyen d'un point de connexion, d'où sont issus deux *branches*. Un choix entre deux scénarios est supposé exclusif. Sur l'exemple de la Figure 6, deux branches exclusives sont définies : la première branche continue le scénario en cours par le bMSC M_1 , l'autre branche par le bMSC M_2 . Une autre interprétation du choix appelée "delayed choice" est donnée dans [BAE 95], et considère qu'un choix n'est pas effectué tant que des événements communs aux deux alternatives sont choisis. Le premier événement n'appartenant pas au préfixe commun à plusieurs alternatives détermine quel scénario est en cours d'exécution.



Figure 6. *Alternative et itération*

Les points de connexion permettent de définir des cycles dans les graphes, et donc d'itérer des comportements (figure 6).

2.4. Composition hiérarchique et conditions

Les références contenues dans un HMSC peuvent être des références à des bMSC, mais aussi à d'autres HMSC. Il est ainsi possible de définir un système de façon hiérarchique. Cette hiérarchie à N niveaux peut être facilement transformée en une hiérarchie à 2 niveaux (un ensemble de bMSC composés au moyen d'un seul HMSC).

En plus des opérations de composition, le standard MSC'96 permet la définition de *conditions*. Une condition se représente dans un bMSC par une étiquette recouvrant

tout ou partie des instances. Dans un HMSC, les conditions sont des noeuds du graphe. Elles ne représentent qu'une indication informelle d'un état particulier du système, par lequel toute exécution ne passe pas forcément.

2.5. Ports

Le standard MSC'96 introduit également des ports, permettant l'envoi ou la réception de messages provenant de l'extérieur d'un bMSC. Un port doit être considéré, selon [REN 98](p70), comme l'adresse d'entrée ou de sortie d'un message. L'émetteur ou le récepteur du message n'est alors connu qu'en fonction du contexte. Ces ports apportent une certaine facilité pour composer les bMSC. Cependant, lorsque les ports sont utilisés en combinaison avec des itérations, des spécifications telles que celle représentée figure 7 peuvent être définies. Ce type de spécification est particulièrement complexe, car la réception du message m peut se faire dans une infinité de scénarios (générés par $M2^*$; $M3$). L'ensemble des ordres générés par cette spécification (figure 7 à droite) ne peut être reconstruit par composition de bMSC sans ports. Ce type de motif est en fait ce que Henriksen [HEN 99a, HEN 00, HEN 99b] définit comme un langage de MSC non finiment engendré. En fait, l'utilisation des ports permet d'utiliser les MSC comme des automates communicants, sur lesquels de nombreux problèmes sont indécidables [GUN 01] (bornitude des files, présence de blocages, ...). Nous pensons qu'il est préférable de n'utiliser que des motifs de communication clos (à l'exception des communications avec l'environnement du système, qui peuvent être perçues comme des événements atomiques) afin de préserver le côté intuitif des communications dans les bMSC, et de permettre la décision de certaines propriétés.

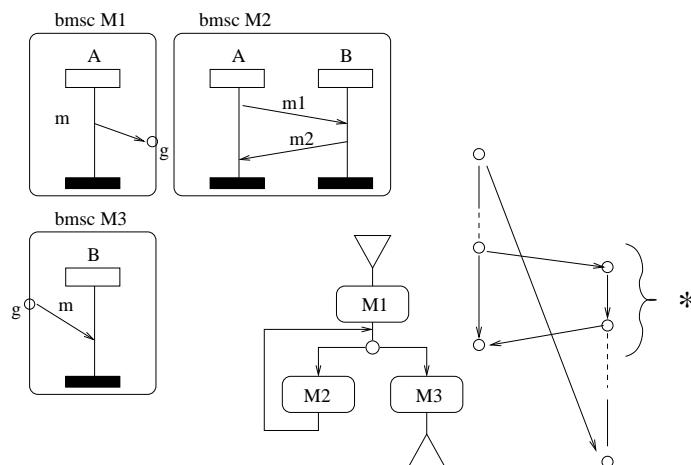


Figure 7. Ports et itération

3. Sémantique

La sémantique “officielle” des MSC définie par Mauw et Reniers [MAU 94, MAU 95, MAU 97a, MAU 97b, MAU 99, REN 98] est donnée sous forme d’algèbre de processus BPA_ϵ . Cette sémantique est le modèle le plus complet proposé pour les MSC. Son inconvénient est de donner une sémantique entrelacée à un modèle initialement décrit par des ordres partiels. Des variantes ont été publiées dans [GEH 98a, GEH 98b, KOS 97b].

Pourtant, dès l’origine, une sémantique d’ordre partiel des bMSC s’est très facilement définie à l’aide de réseaux de Petri [GRA 93]. Un essai pour étendre cette sémantique a été proposé par l’extension aux opérateurs de composition comme proposé dans [HEY 00] reste incomplète. Ce n’est pas surprenant, car le langage d’un HMSC (voir chapitre suivant) n’est pas toujours un langage de réseau de Pétri. En effet, les HMSC permettent d’imposer un ordre de réception sur des messages, tandis qu’un réseau de Petri ne permet pas de conserver cette information dans les jetons ou les places.

L’autre méthode consiste à considérer directement les familles d’ordres partiels engendrées [HEY 98, KAT 98]. Ces sémantiques définissent les MSC comme une famille d’ordres partiels, obtenue par toutes les concaténations possibles des ordres dans un HMSC. Ces familles d’ordres partiels sont en fait les familles d’ordres associées aux chemins du MSC définies précédemment. Bien sûr, ces familles d’ordres peuvent être des ensembles infinis, et comporter des ordres infinis. Elles sont donc difficilement manipulables en extension. Un autre inconvénient de ces sémantiques est que l’information concernant le branchement dans la spécification disparaît complètement. Ainsi, les MSC de la figure 8 définissent la même famille d’ordres.

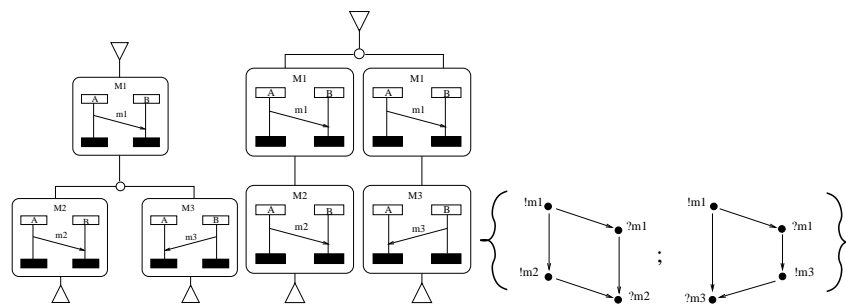


Figure 8. Famille d’ordres définie par les deux MSC

C’est pour garder les branchements dans la sémantique, que nous avons défini [HéL 01b] une sémantique des MSC à partir de structures d’événements et de grammaires de graphes. L’équivalence de sémantique est décidable [HéL 98, HéL 00a], contrairement à celle fondée sur les langages. Elle apparaît aussi particulièrement bien

adaptée pour définir une forme normale [HéL 00b] et un algorithme de simulation [HéL 99].

4. Propriétés des MSC

4.1. Langage d'un MSC

Si l'ensemble des exécutions définies par un bMSC est assez intuitif, il n'en est pas de même pour les HMSC. En effet, la composition séquentielle faible réintroduit du parallélisme (voir l'exemple de la Figure 5, l'envoi de $m2$ et la réception de $m1$ sont deux événements concurrents). Il convient donc de s'interroger sur la nature du langage engendré par un HMSC. Un HMSC peut donc être vu comme un générateur d'un ensemble de bMSC. En considérant les linéarisations des ordres ainsi obtenus, nous pouvons définir le *langage* accepté par un HMSC. [DAR 00] montre que les HMSC permettent de définir des parties rationnelles, et, reprenant les résultats de [BER 79], que les propriétés suivantes sont indécidables : Soient $H1$ et $H2$ deux HMSC, $\mathcal{L}(H1)$ et $\mathcal{L}(H2)$ leurs langages, soit R un sous ensemble rationnel, alors les problèmes suivants sont indécidables : $\mathcal{L}(H1) = \mathcal{L}(H2)$, $\mathcal{L}(H1) \subseteq \mathcal{L}(H2)$, $R \subseteq \mathcal{L}(H2)$, $\mathcal{L}(H1) \subseteq R$, $\mathcal{L}(H1) = R$, $\mathcal{L}(H1)$ est un rationnel. Ces résultats sont prouvés dans [BER 79] en ramenant ces problèmes au problème de correspondance de Post, réputé indécidable. Le *problème de correspondance de Post* (ou PCP) consiste à décider s'il existe, pour un ensemble de couples de mots $\{(u_1, v_1); \dots; (u_m, v_m)\}$ sur un alphabet d'au moins deux lettres une suite d'index $i_1 \dots i_k$ tels que : $u_{i_1}.u_{i_2} \dots u_{i_k} = v_{i_1}.v_{i_2} \dots v_{i_k}$. Pour l'instance $C = \{(ab, abb); (bb, bba); (baa, ab)\}$ du PCP, il est impossible de trouver une solution. Cette instance du PCP se code facilement à l'aide de MSC. Considérons le MSC H de la figure 9. Existe-t-il un chemin p dans H tel que l'ordre généré sur les instances A et B soit isomorphe à l'ordre généré sur les instances C et D ? Il est facile de voir que la réponse à cette question permet de fournir une réponse à l'instance C du PCP, et que ce problème est donc indécidable. Ainsi, la décision de nombreuses propriétés des MSC peut être réduite au PCP, et prouvée indécidable.

4.2. Choix non local

Les choix dans les HMSC sont des éléments importants : ils permettent de commencer à définir la structure de contrôle des protocoles en cours de spécification. Cependant, ces choix peuvent se révéler ambigus, notamment lorsque le contrôle n'est pas localisé sur un seul processus. Un choix non local existe lorsqu'il existe au moins deux scénarios possibles à partir d'un choix, et que plus d'une instance est responsable du comportement choisi. Sur l'exemple de la figure 8, lorsque l'une des instances A ou B choisit un scénario, l'autre instance doit se conformer à ce choix. La sémantique des MSC suppose donc une synchronisation implicite entre A et B .

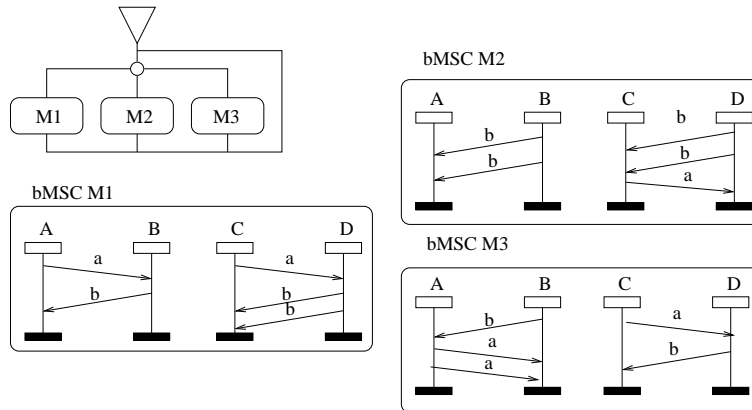


Figure 9. MSC H de Post

Lorsque l'on souhaite définir un ensemble de comportements au moyen d'un HMSC, ce genre de spécification n'est pas particulièrement choquant : seuls les comportements décrits par chaque alternative sont possibles. Par contre, lorsqu'une implémentation est envisagée, les choix non-locaux peuvent provoquer une erreur d'interprétation. En effet, plusieurs sens peuvent être associés à la description de la figure 8 : 1/ seuls les deux scénarios M_2 et M_3 sont possibles, il faudra donc ajouter des communications entre A et B pour éviter les croisements de messages. Le modèle aura donc besoin d'être raffiné ; ou bien 2/ un troisième scénario, dans lequel m_2 et m_3 se croisent est possible et il faut donc le rajouter dans la spécification.

Une définition du choix non-local a été donnée dans [BEN 96, BEN 97b]. Cette définition suppose que toute instance doit émettre ou recevoir un message dans chaque branche d'un choix, et que la séquence de bMSC est une composition séquentielle forte. Cette supposition limite la recherche des choix non-locaux aux arcs sortant d'un nœud de choix. Cependant, si l'on considère la composition séquentielle faible de bMSC comportant des ensembles d'instances disjoints, la présence de choix non-locaux devient une propriété globale du MSC. Considérons le HMSC de la Figure 10: tous les choix semblent être des choix locaux. Cependant, la décision d'un comportement plutôt qu'un autre peut être prise par l'instance A ou l'instance C .

Un algorithme pour la détection de choix non-locaux sur des MSC construits à partir de séquences et d'itérations de bMSC a été décrit dans [HEL 00], puis étendu dans [HéL 01a] pour prendre en compte l'opérateur de composition parallèle.

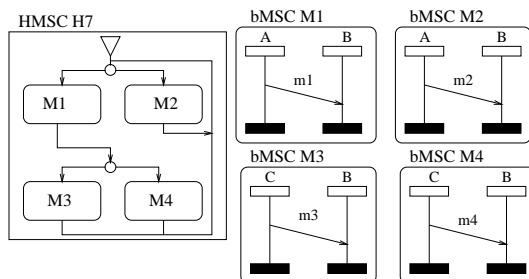


Figure 10. Choix non-local situé sur plus d'un noeud de choix

4.3. Divergence

Il y a divergence lorsqu'un groupe d'instances peut évoluer infiniment sans jamais se synchroniser avec le reste des processus. Ce genre de spécifications divergentes définit un système de transitions sous-jacent infini: soit un nombre de messages non borné peut être en transit, soit un nombre non borné de scénarios peut être inachevé. Un algorithme simple est décrit dans [ALU 99] pour détecter la divergence: s'il existe un cycle dans un HMSC, et que dans le motif itéré par ce cycle les instances ne sont pas synchronisées (soit un groupe d'instances envoie un message qui n'est pas acquitté, soit plusieurs groupes d'instances évoluent en parallèle) alors il y a divergence. Cette propriété des systèmes divergents donne immédiatement un algorithme, basé sur la construction d'un graphe de communication pour chaque cycle. Un MSC H non divergent peut être codé par un système de transitions de taille bornée par $2^{(k-1)m} \cdot (mnk)^k$, où k est le nombre d'instances, m est le nombre d'arcs dans H , et n est le nombre maximal d'événements dans un bMSC de H . Dans cette classe de MSC [MOR 01], de nombreux problèmes deviennent alors décidables. Il est possible, par exemple de faire du model-checking. Cependant, la classe des HMSC bornés est assez restreinte, vu qu'elle exige entre autres que tout message envoyé dans une boucle soit acquitté.

4.4. Analyse de la compétition

Cette question est liée à l'interprétation des réceptions dans les MSC [PEL 00, MUS 99a, MUS 99b, MUS 98, ALU 96, ALU 99, ALU 00]. Ces auteurs considèrent qu'il est impossible d'imposer un ordre entre deux réceptions de messages provenant d'émetteurs différents. Un MSC peut donc définir un *ordre visuel* différent de l'ordre causal. L'analyse de compétition ("race problem") consiste à décider si l'ordre causal est strictement inclus dans l'ordre visuel (l'ordre causal autorise alors plus d'exécutions que l'ordre visuel). Le "race problem" s'avère indécidable.

4.5. Recherche d'un motif

Une instance particulière du problème de la vérification dans un contexte partiellement ordonné est de vérifier qu'un ordre partiel donné (un otif) est "contenu" dans un MSC. Un motif M est contenu dans un bMSC N s'il est possible d'associer à tout événement de M un événement du même type situé sur la même instance, en préservant l'ordre causal sur l'image de M . L'inclusion de motifs s'étend aux HMSC. Un HMSC motif H est en ou-correspondance avec un HMSC G s'il existe un chemin maximal p_H de H et un chemin maximal p_G de G tels que p_H est contenu dans p_G . Un HMSC motif H est en et-correspondance avec un HMSC G s'il existe un chemin maximal p_G de G tel que pour tout chemin maximal p_H de H , p_H soit contenu dans p_G . Les problèmes de correspondance ont été prouvés décidables [MUS 98, MUS 99a, MUS 99b, MUS 00]. Ils ne doivent pas être confondus avec la recherche de chemins maximaux définissant exactement le même ordre, qui est indécidable.

5. Synthèse

La question de synthèse consiste à produire des machines communicantes reproduisant les scénarios d'un MSC donné. La première tentative est décrite dans [YAM 96] : les MSC sont vus comme des descriptions de machines à états finies communiquant de manière synchrone, composées par des expressions régulières. L'algorithme de synthèse proposé consiste à projeter ces expressions régulières sur chaque instance. Non seulement ceci ne préserve pas tous les comportements, mais aussi les choix non locaux peuvent produire des interblocages. On peut noter aussi qu'à l'intérieur du modèle synchrone [KRü 99] a proposé une traduction des HMSC vers les Statecharts.

[ENG 97] et le chapitre 5 de [ENG 01] traite la question de la synthèse, paramétrée par le modèle de communication utilisé (allant du canal FIFO par message jusqu'aux communications synchrones). Certains modèles sont montrés équivalents (ils permettent l'implantation des mêmes bMSC). Une définition de l'implantabilité pour une classe d'architecture est proposée. Un bMSC M est dit **faiblement implantable** pour une classe d'architecture si au moins une trace de M peut être implantée. Un bMSC M est dit **fortement implantable** si toutes les traces de M peuvent être implantées. Cette classification ne s'applique qu'aux bMSC, et l'implantabilité de deux bMSC $M1$ et $M2$ sur une architecture donnée n'assure pas que la composition de $M1$ et $M2$ soit implantable en utilisant le même modèle de communication.

Une synthèse de modèles ROOM à partir de HMSC est proposée dans [LEU 98]. Les ROOM charts sont un genre de Statecharts asynchrones. La composition séquentielle de MSC est considérée comme la séquence forte. L'algorithme de synthèse peut être utilisé sur un sous-ensemble des MSC ne contenant pas de choix non-locaux, de croisement de messages, ou d'actions atomiques. De plus, les MSC sont supposés nor-

malisés, c'est à dire ne pas avoir de préfixes communs au niveau des nœuds de choix, et toute instance est supposée exécuter au moins une émission ou une réception.

La prise en compte de la composition séquentielle faible et de la communication asynchrone a fait l'objet des travaux de [MAN 99, MAN 00]. Le HMSC est projeté sur chacune de ses instances, ce qui donne un squelette de machines à états finies, qui sont ensuite traduites en processus SDL. La méthode de synthèse suppose qu'il existe un canal de communication de type canal SDL entre toute paire de processus communicant. Cependant, le système SDL généré autorise plus de comportements que ceux définis dans le MSC initial. Ceci est dû à l'impossibilité de préserver un ordre entre des messages provenant d'émetteurs différents. Cette approche est implantée dans l'outil MOST (Moscow Synthesizer Tool). Dans [ABD 99, KHE 98, KHE 00, MUS 99c], des processus SDL sont synthétisés pour produire un comportement décrit par un HMSC pour une architecture donnée. L'architecture la plus permissive associe un canal SDL à toute paire d'instances communiquant dans le MSC (l'approche définie est alors très similaire à celle de [MAN 99]). Certaines architectures plus restrictives peuvent empêcher l'implantation de MSC même très simples. Dans cette méthode, le protocole synthétisé peut également autoriser plus de traces que celles définies par le MSC. Cette approche est implantée dans l'outil MSC2SDL. Pour aller plus loin, [HEL 00] a étudié les limites des méthodes de synthèse par projection d'un MSC sur ses instances. Il est montré que la projection ne produit un protocole équivalent au MSC de départ (au sens de l'équivalence de langage) que pour une classe particulière (et très réduite) de MSC appelée MSC reconstructibles.

Du point de vue de la synthèse, on peut encore citer l'approche incrémentale de [ALU 00], la synthèse de réseaux de Petri [DAR 00] et le travail de [MUS 01] consistant cette fois-ci à synthétiser des MSC à partir d'automates de Büchi.

6. Temps et performance

L'utilisation du temps dans les MSC s'est d'abord limitée à l'utilisation de timers dont le comportement était similaire à celui des timers SDL. Les timers permettent d'exprimer un ensemble réduit de contraintes temporelles. Malgré leur apparente simplicité, il est cependant possible de définir à l'aide de timers des comportements incohérents.

De nombreuses directions sont explorées. Dans [ALU 96] est associé à toute paire d'événements un intervalle de temps. À partir de ces intervalles, il est possible de vérifier l'existence d'un ensemble de dates d'occurrences des événements vérifiant les intervalles de temps donnés, ainsi que l'ordre causal du MSC. La satisfiabilité de cet ensemble de contraintes se vérifie grâce à un algorithme de programmation linéaire. [BEN 97c, BEN 97a] étudie la cohérence des contraintes temporelles de MSC contenant une itération. Si un timer est armé en dehors d'une boucle et qu'il expire à l'intérieur de cette boucle, il est difficile de définir quelle contrainte temporelle ce timer définit (contrainte entre l'armement et la première itération de l'expiration, 2^{eme} , ...).

Ce type de MSC est alors jugé incohérent. Par ailleurs, lorsque des intervalles de temps sont spécifiés pour l'exécution d'un scénario, l'ensemble des contraintes temporelles peut se révéler incohérent. Les PMSC (pour "Performance MSC") proposés dans [FAL 97, LAM 98, SLO 98a, SLO 98b] intègrent aux MSC des aspects performances, permettant de décrire par des bMSC annotés des modèles de tâches, qui sont ensuite composés par des opérateurs de séquence forte, de choix, et des itérations bornées. Il est ensuite possible de répondre à des questions concernant la longueur d'un canal de communication, le délai moyen avant consommation d'un message, le taux d'utilisation et le débit d'un canal ou d'une instance, le temps entre deux événements, la durée d'une exécution particulière. [KOS 97a, KOS 00] proposent une traduction des MSC en Timed Maude, un langage de spécification algébrique. [LEM 00] propose d'associer un délai à tout événement et à tout message. Les MSC ainsi étendus sont ensuite transformés en automates d'ordres, puis étudiés grâce à des techniques d'analyse ($max, +$). Il est ainsi possible d'étudier des temps d'exécution de séquences, ou les comportements asymptotiques de systèmes définis à l'aide de HMSC, en termes de débits, de croissance des canaux de communication,... Enfin, [LI 99b, LI 99a] définit des contraintes temporelles sous forme d'inéquations. Une solution basée sur la programmation linéaire est ensuite proposée pour vérifier qu'un bMSC vérifie bien l'ensemble des contraintes qui lui sont attachées. L'algorithme est ensuite étendu aux HMSC, et consiste alors à vérifier que tous les chemins du HMSC vérifient les contraintes qui leur sont associées, en calculant une forme normale du HMSC, puis en vérifiant les contraintes temporelles associées aux sous-expressions de cette forme normale. Cet algorithme s'accompagne de restrictions sur les utilisations de timers similaires à celles définies dans [BEN 97c, BEN 97a], et suppose que la composition séquentielle de bMSC est une composition séquentielle forte.

De nouvelles extensions temporelles permettant de spécifier des contraintes temporelles plus élaborées ont été ajoutées à la norme Z.120. Entre autres, ces extensions permettent de spécifier une date absolue d'exécution d'un événement, de mesurer le temps écoulé entre deux événements, ...

7. Autres langages apparentés

Les "interworkings" sont un langage graphique dans le style des MSC, développé pour répondre à une demande de Philips. Les communications entre composants d'un interworking sont synchrones, et les diagrammes de base sont composés grâce à des opérateurs de séquence, de composition parallèle et de choix. Leur sémantique a été définie à l'aide d'algèbres de processus [MAU 01]

Les "message Flow Graphs" ont été proposés par Leue et Ladkin [LEU 94a, LAD 95b, LEU 94b, LAD 95a] comme sémantique des HMSC. Ce sont des graphes orientés, définissant pour chaque instance une relation de précédence causale, et une relation de communication entre instances. Un événement (noeud du graphe) ne peut s'exécuter que si tous ses prédécesseurs dans la relation de précédence et dans la relation de communication ont été exécutés. L'exécution d'un tel modèle suppose donc le

stockage d'une variable d'histoire au niveau de chaque choix. Le nombre de variables stockées dans le cas de spécifications divergentes peut alors être infini.

Un langage proche des MSC appelé “**Live Sequence Charts**” a été proposé par [DAM 99]. Ce langage est basé sur la composition de diagrammes de séquence similaires aux bMSC. Les différences notables entre les MSC et les LSC résident dans le fait que certains scénarios sont définis comme obligatoires (une exécution doit correspondre à ce scénario) ou optionnels. Cette particularité fait souvent dire que les LSC sont une “extension” des MSC. En fait, il n'en est rien. La sémantique des LSC interdit de réexécuter des événements contenus dans un diagramme de base tant que tous les événements de l'occurrence précédente du scénario n'ont pas été exécutés. Cette contrainte est plus faible que la composition séquentielle forte (on ne synchronise les instances que lors des itérations de comportements), mais réduit cependant le pouvoir d'expression aux langages réguliers. Il est alors possible de faire de la vérification des LSC, comme proposé dans [HAR 00b]. Une utilisation des LSC pour la conception de systèmes réactifs est proposée dans [HAR 00a]. Cette approche se base sur un cycle itératif de capture d'exigences, d'extractions de comportement sous forme de LSC et synthèse de système.

8. Bibliographie

- [ABD 99] ABDALLA M., KHENDEK F., BUTLER G., « New Results on Deriving SDL Specifications from MSCs », R.DSSOULI G. . Y. E., Ed., *Proceedings of 9th SDL forum*, 1999, p. 51-66.
- [ALU 96] ALUR R., HOLZMANN G., PELED D., « An analyzer for Message Sequence Charts », *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, n° 1055 LNCS, 1996, p. 35-48.
- [ALU 99] ALUR R., YANNAKAKIS M., « Model checking of message sequence charts », *Proceedings of the Tenth International Conference on Concurrency Theory*, n° 1664 LNCS, 1999, p. 114-129.
- [ALU 00] ALUR R., ETESSAMI K., YANNAKAKIS M., « Inference of message sequence charts », *22nd International Conference on Software Engineering*, 2000, p. 304-313.
- [BAE 95] BAETEN J., MAUW S., « Delayed choice: an operator for joining Message Sequence Charts », HOGREFE I. D., LEUE S., Eds., *Formal Description Techniques, VII*, Chapman & Hall, 1995, p. 340-354.
- [BEN 96] BEN-ABDALLAH H., LEUE S., « Syntactic Analysis of Message Sequence Chart Specifications », rapport n° 96-12, 1996, Dept. of Electrical and Computer Engineering, University of Waterloo.
- [BEN 97a] BEN-ABDALLAH H., LEUE S., « Expressing and Analyzing Timing Constraints in Message Sequence Charts specifications », rapport n° 97-04, Avril 1997, Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada.
- [BEN 97b] BEN-ABDALLAH H., LEUE S., « Syntactic Detection of Process Divergence and non-Local Choice in Message Sequence Charts », *Proceedings of the Third International Workshop on Tools and Algorithms for the Construction and Analysis of Systems TACAS'97*, n° 1217 LNCS, 1997, p. 259-274.

- [BEN 97c] BEN-ABDALLAH H., LEUE S., « Timing Constraints in Message Sequence Chart Specifications », *Proceedings of the Tenth Conference on Formal Description Techniques FORTE/PSTV'97*, Osaka, Japan, Novembre 1997, Chapman & Hall.
- [BER 79] BERSTEL J., *Transductions and Context-Free-Languages*, B.G. Teubner, Stuttgart, 1979.
- [DAM 99] DAMM W., HAREL D., « LSCs: Breathing Life into Message Sequence Charts », *FMOODS'99 IFIP TC6/WG6.1 Third International Conference on Formal Methods for Open Object-Based Distributed Systems*, 1999.
- [DAR 00] DARONDEAU P., CAILLAUD B., LESVENTES G., HÉLOUËT L., « HMSCs as partial specifications ... with PNs as completions », *Proceedings of Movep'2000, Modeling and Verification of Parallel Processes*, Nantes, France, Juin 2000.
- [DUL 96] DULZ W., « A Framework for the Performance Evaluation of SDL/MS-C-specified Systems », *Proceedings of the ESM'96*, Budapest, June 1996.
- [ENG 97] ENGELS A., MAUW S., RENIERS M., « A Hierarchy of Communication Models for Message Sequence Charts », T. MIZUNO N. SHIRATORI T. H., TOGASHI A., Eds., *Proceedings of FORTE X and PSTV XVII*, Osaka, Japon, Novembre 1997, Chapman & Hall, p. 75-90.
- [ENG 01] ENGELS A., « Languages for Analysis and Testing of Event Sequences », PhD thesis, Institute for Programming research and Algorithmics, Eindhoven University, 2001.
- [FAL 97] FALTIN N., LAMBERT L., MITSCHELE-THIEL A., SLOMKA F., « PMSC – Performance Message Sequence Chart », rapport, Octobre 1997, Formale Beschreibungstechniken für verteilte Systeme, 7. GI/ITG-Fachgespräch.
- [GEH 98a] GEHRKE T., HUHN M., RENSINK A., WEHRHEIM H., « An Algebraic Semantics for Message Sequence Chart Documents », *Proceedings of the Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, and Protocol Specification, Testing and Verification (FORTE/PSTV '98)*, 1998, p. 3-18.
- [GEH 98b] GEHRKE T., HUHN M., RENSINK A., WEHRHEIM H., « A process algebra semantics for message sequence charts including conditions », *8. GI/ITG Fachgespräch Formale Beschreibungstechniken für verteilte Systeme*, 1998.
- [GRA 93] GRAUBMANN P., RUDOLPH E., GRABOWSKI J., « Towards a Petri Net based Semantics Definition of Message Sequence Charts », FAERGEMAND O., SARMA A., Eds., *SDL'93: Using Objects*, Elsevier Science Publishers BV, 1993.
- [GRA 98] GRABOWSKI J., « Semantics for timed Message Sequence Charts via constraint diagrams », *1st conference on SDL and MSCs (SAM98)*, Berlin, 1998.
- [GUN 01] GUNTER E., MUSCHOLL A., PELED D., « Compositional Message Sequence Charts », *Proc of Tacas'2001: Tools and Algorithms for the Construction and Analysis of Systems*, 2001.
- [HAR 00a] HAREL D., « From Play-In Scenarios to Code: An Achievable Dream », MAIBAU T., Ed., *Proceedings of FASE 2000*, n° 1783 LNCS, Berlin, Allemagne, Mars 2000, Springer, p. 22-34.
- [HAR 00b] HAREL D., KUGLER H., « Synthesizing State-Based object systems from LSC Specifications », *5th int. Conference on Implementation and Application of Automata*, LNCS, 2000.
- [HEL 00] HELOUËT L., JARD C., « Conditions for synthesis of communicating automata from HMSCs », *5th international workshop on Formal Methods for Industrial Critical Systems (FMICS'00)*, 2000.

- [HEN 99a] HENRIKSEN J., MUKUND M., KUMAR K., THIAGARAJAN P., « On message Sequence Graphs and Finitely Generated Regular MSC Languages », *proceedings of the 26-th International Colloquium on Automata, Languages, and Programming (ICALP'99)*, Prague, Tchécoslovaquie, Juillet 1999.
- [HEN 99b] HENRIKSEN J., MUKUND M., NARAYAN KUMAR K., THIAGARAJAN P., « Towards a Theory of Regular MSC Languages », rapport n° RS-99-52, 1999, BRICS.
- [HEN 00] HENRIKSEN J., MUKUND M., NARAYAN KUMAR K., THIAGARAJAN P., « Regular Collections of Message Sequence Charts », *Proc. MFCS '00*, LNCS, 2000.
- [HEY 98] HEYMER S., « A non-interleaving semantics for MSC », *1st conference on SDL and MSCs (SAM98)*, Berlin, 1998.
- [HEY 00] HEYMER S., « A semantics for MSCs based on Petri net components », *2nd Conference on SDL and MSC (SAM 2000)*, Grenoble, France, juin 2000.
- [HéL 98] HÉLOUËT L., JARD C., CAILLAUD B., « An Effective equivalence for sets of scenarios represented by HMSCs », rapport n° 3499, September 1998, INRIA, <ftp://ftp.inria.fr/INRIA/publication/RR/RR-3499.ps.gz>.
- [HéL 99] HÉLOUËT L., « A Simulation Framework for Message Sequence Charts », R. DSSOULI G.V. BOCHMANN Y., Ed., *SDL'99, The Next Millenium*, Elsevier, 1999, p. 473-488.
- [HéL 00a] HÉLOUËT L., « Analyse des exigences des systèmes répartis exprimées par des langages de scénarios », PhD thesis, MATISSE, Université de Rennes 1, Octobre 2000.
- [HéL 00b] HÉLOUËT L., LE MAIGAT P., « Decomposition of Message Sequence Charts », *2nd Conference on SDL and MSC (SAM 2000)*, 2000.
- [HéL 01a] HÉLOUËT L., « Some Pathological Message Sequence Charts, and How to detect them », *proc of 10th SDL Forum*, 2001.
- [HéL 01b] HÉLOUËT L., JARD C., CAILLAUD B., « An event Structure Semantics for Message Sequence Charts », *Mathematical Structures in Computer Science*, vol. to appear, 2001.
- [ITU 99] ITU-TS, *ITU-TS Recommendation Z.120: Message Sequence Chart (MSC)*, ITU-TS, Geneva, novembre 1999.
- [KAT 98] KATOEN J., LAMBERT L., « Pomsets for message sequence charts », *1st conference on SDL and MSCs (SAM98)*, Berlin, 1998.
- [KHE 98] KHENDEK F., ROBERT G., BUTLER G., GROGONO P., « Implementability of message sequence charts », *1st conference on SDL and MSCs (SAM98)*, Berlin, 1998.
- [KHE 00] KHENDEK F., VINCENT D., « Enriching SDL specifications with MSCs », *2nd Conference on SDL and MSC (SAM 2000)*, Grenoble, France, 2000.
- [KOS 97a] KOSIUCZENKO P., « Time in Message Sequence Charts: A Formal Approach », rapport n° 9703, janvier 1997, Ludwig-Maximilians-Universität München, Institut für Informatik.
- [KOS 97b] KOSIUCZENKO P., « Towards a formal semantics of MSC'96: Inline Expressions », rapport n° 9705, 1997, Ludwig-Maximilians-Universität München, Institut für Informatik.
- [KOS 00] KOSIUCZENKO P., « Towards an Integration of Message Sequence Charts and Timed Maude », *Journal of IDPT*, vol. "to appear", 2000.
- [KRü 99] KRÜGER I., GROSU R., SCHOLZ P., BROY M., « From MSCs to Statecharts », *Distributed and Parallel Embedded Systems*, Kluwer Academic Publishers, 1999.
- [LAD 95a] LADKIN P., LEUE S., « Comments on a Proposed Semantics for Basic Message Sequence Charts », *The Computer Journal*, vol. 37, n° 9, 1995.

- [LAD 95b] LADKIN P., LEUE S., « Interpreting Message Flow Graphs », *Formal Aspects of Computing*, vol. 7, n° 5, 1995, p. 473-509.
- [LAM 98] LAMBERT L., « PMSC for Performance Evaluation », Workshop on Performance and Time in SDL/MS, Erlangen, 1998.
- [LEM 00] LE MAIGAT P., HÉLOUËT L., « A (Max,+) approach for time in Message Sequence Charts », *Proc of WODES'2000 (Workshop on Discrete Event Systems)*, Gent, Belgique, aout 2000.
- [LEU 94a] LEUE S., LADKIN P., « Four issues concerning the semantics of message flow graphs », rapport, 1994, INRIA Lorraine.
- [LEU 94b] LEUE S., LADKIN P., « What do message sequence charts mean? », *Proceedings of FORTE'93*, In North-Holland, 1994, p. 301-315.
- [LEU 98] LEUE S., MEHRMANN L., REZAI M., « Synthesizing ROOM Models from Message Sequence Chart Specifications », *Proceedings of 13th IEEE Conference on Automated Software Engineering*, Honolulu, Hawaii, Octobre 1998.
- [LI 99a] LI X., LILLIUS J., « Timing analysis of Message Sequence Charts », rapport n° 255, 1999, Turku Center for Computer Science (TUCS).
- [LI 99b] LI X., LILLIUS J., « Timing analysis of UML sequence diagrams », *UML'99 proceedings*, n° 1723 LNCS, Berlin, 1999, p. 661-674.
- [MAN 99] MANSUROV N., ZHUKOV D., « Automatic Synthesis of SDL models in use case Methodology », R.DSSOULI G. . Y. E., Ed., *Proceedings of 9th SDL forum*, 1999, p. 225-240.
- [MAN 00] MANSUROV N., VASURA D., « Approximation of (H)MSC semantics by Event Automata », *2nd Conference on SDL and MSC (SAM 2000)*, Grenoble, France, juin 2000.
- [MAU 94] MAUW S., RENIERS M., « An algebraic semantics of Basic Message Sequence Charts », *The Computer Journal*, vol. 37, n° 4, 1994, p. 269-277.
- [MAU 95] MAUW S., « The formalization of message sequence charts », rapport, 1995, Eindhoven University of Technology.
- [MAU 97a] MAUW S., RENIERS M., « High-level Message Sequence Charts », CAVALLI A., SARMA A., Eds., *SDL'97: Time for Testing - SDL, MSC and Trends, Proceedings of the Eighth SDL Forum*, 1997, p. 291-306.
- [MAU 97b] MAUW S., RENIERS M., « Operational semantics for MSC'96 », CAVALLI A., VINCENT D., Eds., *SDL'97: Time for Testing - SDL, MSC and Trends, Tutorials of the Eighth SDL Forum*, 1997, p. 135-152.
- [MAU 99] MAUW S., RENIERS M., « Operational semantics for MSC'96 », *Computer Networks and ISDN Systems*, vol. 31, n° 17, 1999, p. 1785-1799.
- [MAU 01] MAUW S., RENIERS M., « A process algebra for Interworkings », Chapitre 19, p. 1269-1327, Elsevier Science B.V., 2001.
- [MIT 99] MITSCHELE-THIEL A., MÜLLER-CLOSTERMANN B., « Performance Engineering of SDL/MS Systems », *Computer Networks*, vol. 31, n° 17, 1999, p. 1801-1815.
- [MOR 01] MORIN R., « On Regular Message Sequence Charts Languages and Relationships to Mazurkiewicz Trace Theory », *Foundation of Software Science and Computation Structures*, 2001.
- [MUK 00] MUKUND M., K. N. K., SOHONI M., « Synthesizing Distributed Finite-State Systems from MSCs », *Proc. CONCUR '00 to appear in LNCS (2000)*, 2000.
- [MUN 96] MUNIZ SILVA P., « Extended Message Sequence Charts with Time-Interval Semantics », rapport, 1996, University of Sao Paulo.

- [MUS 98] MUSCHOLL A., PELED D., SU Z., « Deciding properties of message sequence charts », *Proc. of FoSSaCS'98*, n° 1378 LNCS, 1998, p. 226-242.
- [MUS 99a] MUSCHOLL A., « Matching specifications for message sequence charts », *Proc. of FoSSaCS'99*, n° 1578 LNCS, 1999.
- [MUS 99b] MUSCHOLL A., PELED D., « Message sequence graphs and decision problems on Mazurkiewicz traces », *Proc. of MFCS'99*, LNCS 1672, 1999, p. 81-91.
- [MUS 99c] MUSSA M., « Automatic Generation of SDL specifications from MSC », Msc Thesis, Concordia University, Montreal, Novembre 1999.
- [MUS 00] MUSCHOLL A., « Analyzing Message Sequence Charts », *2nd Conference on SDL and MSC (SAM 2000)*, 2000.
- [MUS 01] MUSCHOLL A., PELED D., « From Finite State Communication Protocols to High-Level Message Sequence Charts », *Proc of ICALP 2001*, 2001.
- [PEL 00] PELED D., « Formal Methods for Message Sequence Charts », H.LAI T., Ed., *Proceedings of ICDS'2000*, Taipei, Taiwan, Avril 2000, IEEE, p. E7-E12.
- [PRA 86] PRATT V., « Modeling Concurrency with Partial Orders », *International journal of Parallel Programming*, vol. 15, n° 1, 1986, p. 33-71.
- [REN 94] RENSINK A., H. W., « Weak Sequential Composition in Process Algebras », JONSSON B., PARROW J., Eds., *CONCUR '94: Concurrency Theory, 5th International Conference*, vol. 836 de *Lecture Notes in Computer Science*, Uppsala, Sweden, 22-25 aout 1994, Springer-Verlag, p. 226-241.
- [REN 98] RENIERS M., « Message Sequence Chart: Syntax and Semantics », PhD thesis, Eindhoven University of Technology, 1998.
- [RUD 96] RUDOLPH E., GRAUBMAN P., GRABOWSKI J., « Tutorial On Message Sequence Charts », *Computer Networks and ISDN Systems*, vol. 28, 1996, p. 1629-1641.
- [SLO 98a] SLOMKA F., ZANT J., LAMBERT L., « MSC-based Schedulability Analysis », Workshop on Performance and Time in SDL and MSC, n° Technical Report 1/98, Fevrier 1998, IMMD VII, University of Erlangen-Nuremberg, Erlangen.
- [SLO 98b] SLOMKA F., ZANT J., LAMBERT L., « Schedulability Analysis of Heterogeneous Systems for Performance Message Sequence Chart », *6th International Workshop on Hardware/Software Codesign*, Seattle, Mars 1998, IEEE Computer Society Press.
- [YAM 96] YAMANAKA K., KOMURA S., KATO J., ICHIKAWA H., « Deriving Protocols from Message Sequence Charts in a Communicating Processes Model », *IEICE Transactions on Information and Systems*, vol. E79-D, n° 11, 1996, p. 1533-1544.