Etat de l'art sur les langages de scenarios

Loïc Hélouët[♦] , Claude Jard[■]

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

[■] *IRISA,*
Capmus de Beaulieu, 35042 Rennes Cedex
claude.jard@irisa.fr
http://www.irisa.fr/pampa

RÉSUMÉ. *Cet article est un état de l'art sur les diagrammes de sequence*

ABSTRACT. *This paper is a survey on sequence diagrams.*

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

KEYWORDS: *Scenarios, sequence diagrams, survey*

1. Introduction

Un des formalismes les plus utilisés pour décrire informellement le comportement d'entités et les communication 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. De nombreuses documentations de protocoles de communication contiennent ces représentations, qui servent à illustrer par un exemple le comportement d'un système.

Les Message Sequence Charts (ou MSC) sont un formalisme qui permet de spécifier graphiquement le comportement d'entités communicantes. Initialement, les MSC n'étaient utilisés qu'en tant que traces de sortie pour 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. De manière plus ou moins indépendante, un langage de scénarios appelé interworkings (MSC à communications synchrones) a été développé dans les laboratoires de Philips.

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 la section suivante. 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. Selon [ITU 99], les MSC ont cependant été initialement prévus pour :

– donner une vision d'un service fourni par plusieurs entités,
– exprimer des exigences,
– comme base pour l'élaboration, la simulation et la vérification de systèmes SDL,
– pour exprimer des jeux de test,
– pour spécifier des communications ou des interfaces,
– comme une formalisation des use-cases dans les méthodes orientées objet.

Durant la période 1996-2000, les MSC ont gagné en popularité, et de nombreuses personnes ont commencé à s'y intéresser en dehors de la communauté SDL. De nombreuses ambiguïtés ont été constatées, et des propriétés d'indécidabilité prouvées sur le langage. Quelques variantes et interprétations du langage ont aussi été proposées ([DAM 99]). Parallèlement, la communauté UML s'intéresse aussi aux MSC, afin d'étendre les diagrammes de séquence, assez peu expressifs.

Dans ce chapitre, nous n'aborderons que les MSC'96. En effet, la manipulation de données dans les MSC'2000 n'est pas encore standardisée, et revêt peu d'intérêt pour

les manipulations formelles étudiées par la suite. Les MSC'96 permettent de définir un système hiérarchiquement. Au plus bas niveau, des Basic Message Sequence Charts (bMSC) décrivent le comportement de processus. Les bMSC sont ensuite composés par plusieurs niveaux hiérarchiques de graphes, les High Level Message Sequence Charts (HMSC).

Ce chapitre commence par donner une description des bMSC, ainsi que leur sémantique opérationnelle. La section suivante définit les HMSC, s'intéresse à leur pouvoir d'expression, et aux propriétés décidables sur ce langage.

1.1. *Basic Message Sequence Charts*

Un Basic Message Sequence Charts (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. Un exemple de bMSC est donné Figure 2. Outre les communications, un bMSC peut contenir des actions atomiques, des opérations sur des horloges (armement, désarmement, expiration). La table de la Figure 1 illustre la notation graphique associée à chaque élément du langage.

Les communications entre instances sont supposées asynchrones. La nature des médiums 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. 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 3, les événements $a1$ et $a3$ ne sont pas ordonnés sur l'axe de l'instance A.

1.2. *High-level Message Sequence Charts*

Les basic Message Sequence Charts ne permettent que la définition de scénarios très simples. Le formalisme MSC'96, également appelé High-level Message Sequence Charts (HMSC) a 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 Figure 4.

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].

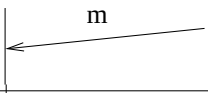
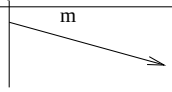
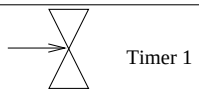
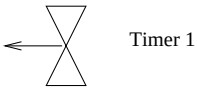
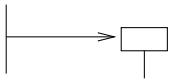
Messages	Réception	
	Émission	
Horloges	Armement	
	Désarmement	
	Expiration	
Instances	Creation	
	Stop	
	Action interne	

Figure 1. *Éléments de bMSC*

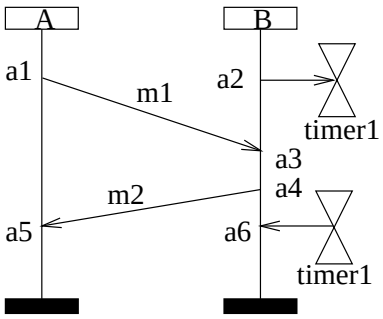


Figure 2. *Un exemple de bMSC.*

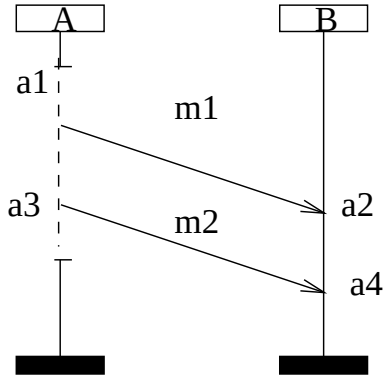


Figure 3. *Un bMSC contenant une coregion*

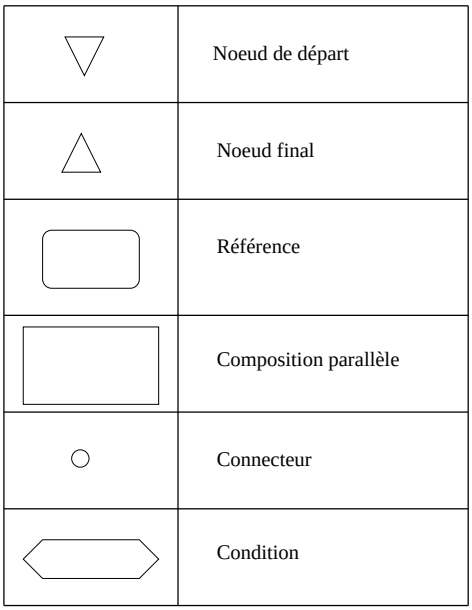


Figure 4. *Éléments d'un HMSC*

1.2.1. *Composition parallèle*

La composition parallèle entre deux MSC permet de définir un comportement concurrent de deux scénarios. La composition parallèles de M_1 et M_2 Figure 5 admet tous les entrelacements des comportements définis par M_1 et M_2.

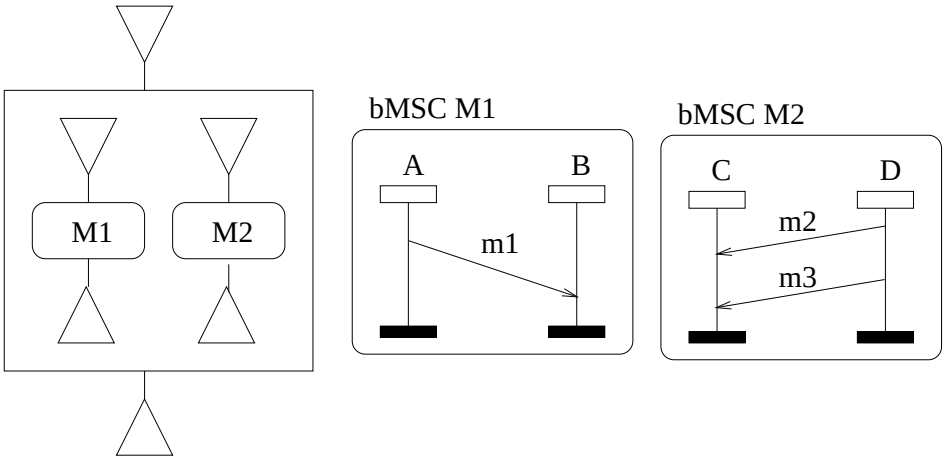


Figure 5. *Composition parallèle de MSC*

Les spécifications mises en parallèle ne doivent pas forcément être finies, ni comporter des ensembles d'instances disjoints. Cependant, 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 5 est équivalent au HMSC Figure 6.

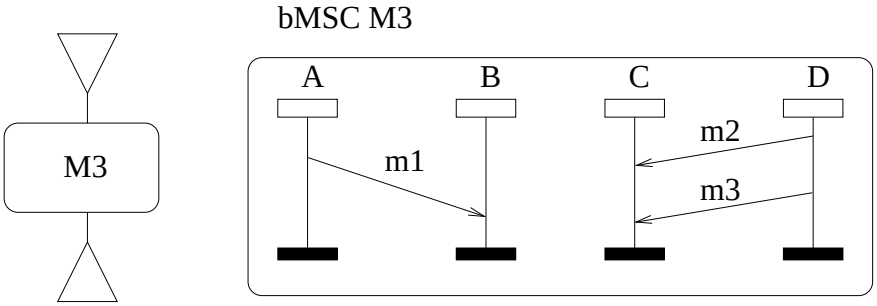


Figure 6. *Équivalence*

1.2.2. *Composition Séquentielle*

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

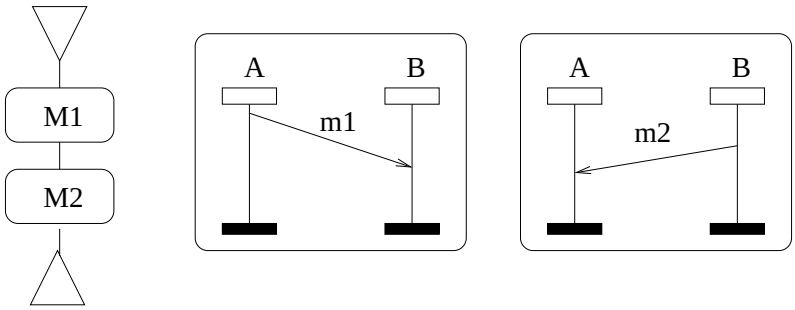


Figure 7. *Séquence de bMSC*

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 précédence causale est tout simplement "recollée" le long des instances communes.

Composition séquentielle forte :

La composition séquentielle forte considère que dans la séquence $M1$ suivi de $M2$, tous les évènements de $M1$ doivent être terminés avant qu'un événement de $M2$ ne soit exécuté. Sous hypothèse de composition séquentielle forte, la séquence $M1; M2$ de l'exemple Figure 8 ne décrit qu'une seule trace, $!m1; ?m1; !m2; ?m2$.

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 l'hypothèse de la concaténation forte de bMSC peut se justifier dans certains contextes, elle semble cependant peu adaptée à la description de systèmes distribués

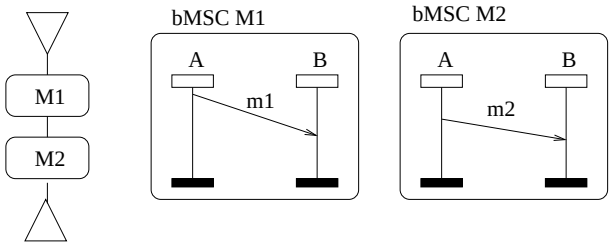


Figure 8. *Composition séquentielle*

asynchrones. En effet, si les communications ne sont pas supposées instantanées, une synchronisation se révèle coûteuse. De plus, il n'y a aucune raison pour qu'une instance n'effectue pas l'envoi d'un message dès qu'elle est prête à le faire. Il peut en résulter des spécifications à espace d'états infinis (les files de communication ne sont pas bornées).

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 : $!m1; ?m1; !m2; ?m2$, et $!m1; !m2; ?m1; ?m2$. Par contre, 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'obtenir une taille des canaux de communication non bornée. Sur l'exemple de la Figure 9, l'instance A peut envoyer une infinité de messages de type m sans jamais se synchroniser avec l'instance B, ce qui peut se traduire par une accumulation de messages dans le médium de communication de A vers B.

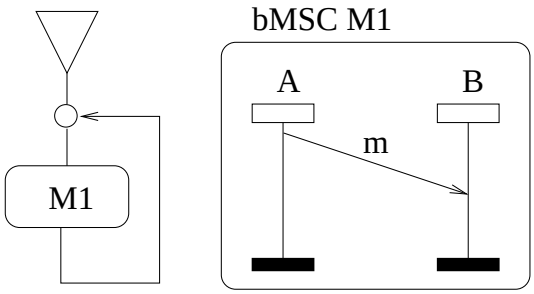


Figure 9. *HMSC décrivant une spécification non finie*

En objection à cette sémantique de la composition séquentielle, certains auteurs affirment qu'il n'est pas raisonnable de laisser des files de communication non bornées. En fait, l'adoption de la composition séquentielle faible n'implique pas nécessairement que les files ne soient pas bornées : 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 files, et permettre l'ajout de messages d'acquittement. La détection d'une accumulation potentielle de message dans une file ne signifie cependant pas que cette situation se produise dans un système réel. En effet, une spécification jugée non bornée peut fonctionner correctement sans débordement ni perte si les messages sont consommés suffisamment rapidement, bornant ainsi "naturellement" le système. Une analyse temporisée des systèmes peut donc s'avérer utile.

1.3. *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 10, deux branches exclusives sont définies : la première branche continue le scénario en cours par le bMSC M_1, l'autre branche par le bMSC $M2$. 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 prefixe commun à plusieurs alternatives détermine quel scénario est en cours d'exécution.

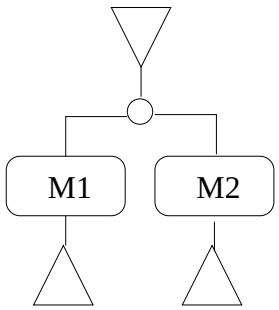


Figure 10. *Alternative entre deux MSC*

1.4. *Itération*

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

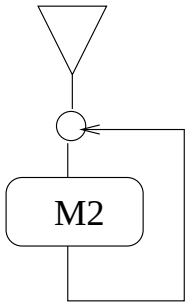


Figure 11. *Itération d'un comportement*

Ainsi, l'exemple de la Figure 11 peut être vue comme une suite infinie de compositions séquentielles du bMSC $M2$.

1.5. *Composition Hiérarchique*

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). L'exemple hiérarchique de la Figure 12 est équivalent à l'exemple de la Figure 13.

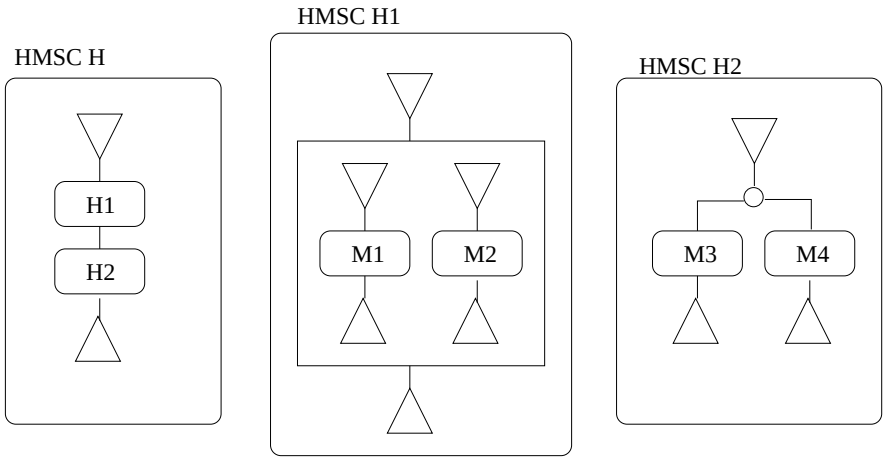


Figure 12. *Composition hiérarchique*

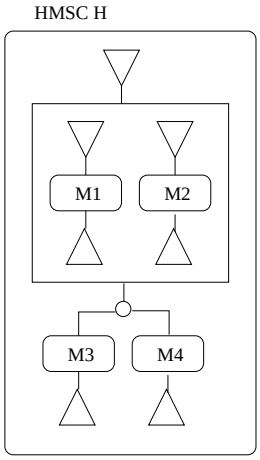


Figure 13. *HMSC équivalent au HMSC Figure 12*

1.6. *Conditions*

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. Les conditions ne représentent qu'un état global particulier de l'ensemble des instances qu'elle recouvre. Elles ont été initialement créées comme moyen de composer les bMSC (en les recollant le long des conditions communes). Les conditions sont parfois interprétées comme des synchronisations. Cependant, dans le standard MSC'96, elles ne représentent plus qu'une qu'une indication informelle d'un état particulier du système, par lequel toute exécution ne passe pas forcément.

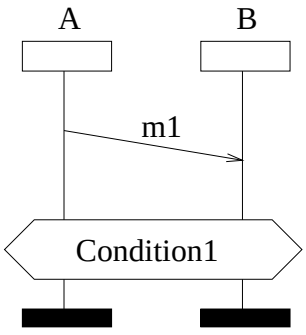


Figure 14. *Condition dans un bMSC*

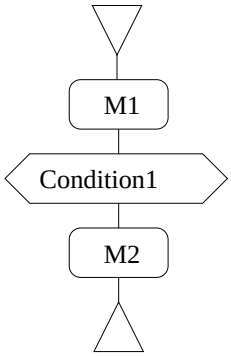


Figure 15. *Condition dans un HMSC*

1.7. *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. Sur l'exemple de la Figure 16, le message m_2 envoyé sur le port g ne peut être reçu que par l'instance C, à un instant précis du scénario $M1$. Cependant, lorsque les ports sont utilisés en combinaison avec des itérations, des spécifications telles que celle représentée Figure 17 peuvent être définies. Ce type de spécification est particulièrement ambigu, 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 (voir Figure 18) ne peut être reconstruit par composition de bMSC sans ports. Ce type de motif est en fait ce que Henriksen [HEN 99a] défini 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 (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. Par la suite, nous n'utiliserons donc pas de ports pour les communications.

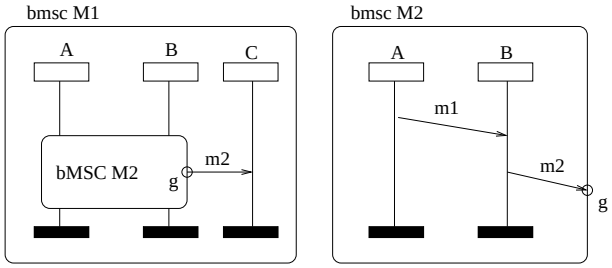


Figure 16. *Port dans un bMSC*

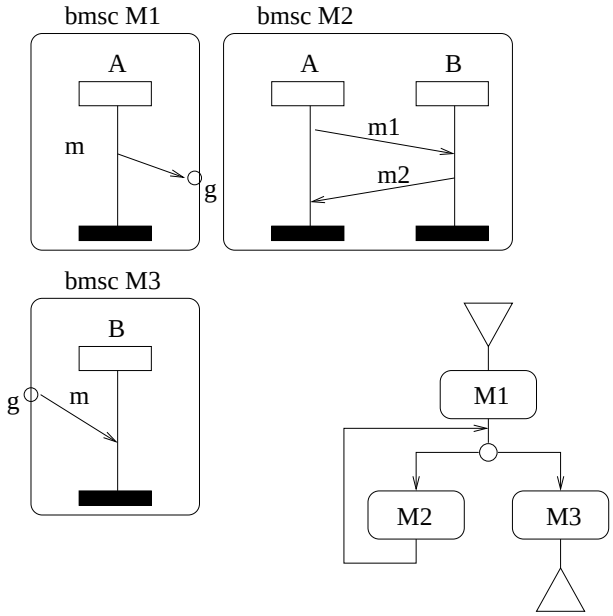


Figure 17. *Ports et itération*

2. Sémantique

2.1. *Sémantiques des bMSC*

Deux sémantiques ont été proposées pour les basic Message Sequence Charts. [MAU 94] définit un bMSC au moyen d'une algèbre de processus. [GRA 93] propose un modèle non entrelacé des bMSC, sous forme de réseaux d'occurrence (réseau de Petri dans lequel chaque place a au plus un prédécesseur et un successeur). Une des

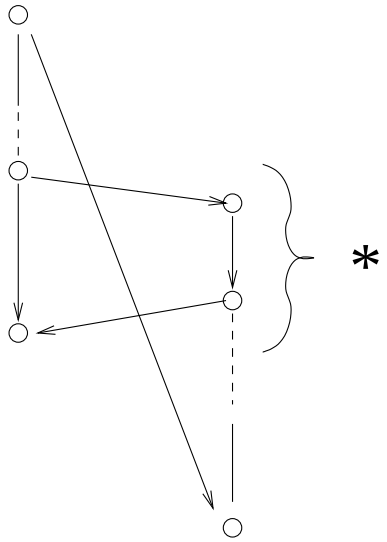


Figure 18. *Ordres générés par le HMSC Figure 17*

particularités de cette sémantique est de considérer le début et la fin d'une corégion comme deux événements observables.

2.2. *Algebres de processus*

La sémantique "officielle" des HMSC 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 HMSC. Son seul inconvénient est de donner une sémantique entrelacée à un modèle initialement décrit par des ordres partiels. Un autre sémantique sous forme d'algèbre de processus a été définie dans [GEH 98a, GEH 98b].

Une autre sémantique entrelacée a été proposée par [KOS 97b].

2.3. *Modeles d'ordres partiels*

[GRA 93] définit une sémantique des Basic MSC à partir de réseaux de Pétri. Un essai pour étendre cette sémantique a été proposé par [HEY 00]. Cependant cette sémantique reste incomplète. Ce n' est pas surprenant, car le langage d' un HMSC 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.

Deux sémantiques non entrelacées assez proches ont été proposées par [HEY 98] et [KAT 98]. Ces sémantiques définissent les HMSC 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 HMSC Figures 19 et 20 définissent la même famille d'ordres Figure 21.

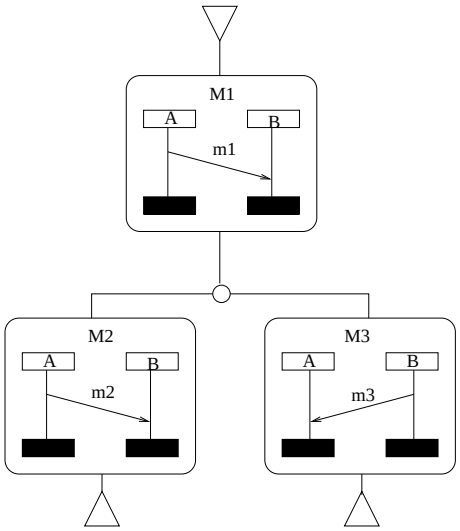


Figure 19. *HMSC P_0 (Comportant un choix non-local)*

[HéL 01b] définit une sémantique des HMSC à partir de structures d'événements et de grammiares de graphes. Cette sémantique pemet de donner à un sous-ensemble des MSC'96 une sémantique à base d'ordres partiels conservant les informations de branchement contenues dans les HMSC.

3. Propriétés des MSC

3.1. *Langage d'un HMSC*

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 (sur l'exemple de la Figure 8, 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.

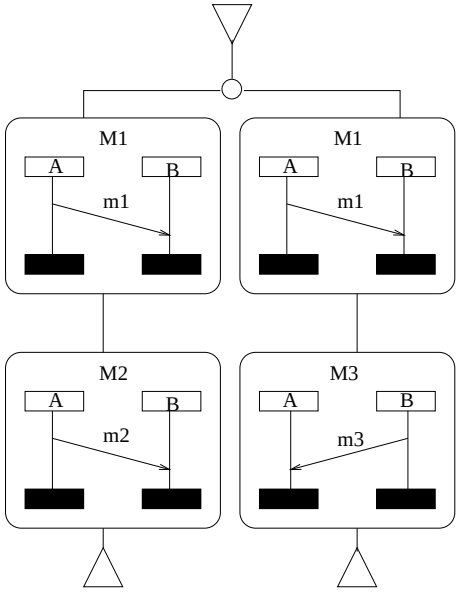


Figure 20. *HMSC P_1*

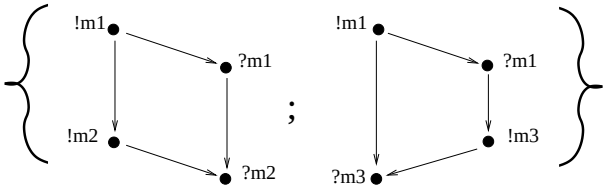


Figure 21. *Famille d'ordres définie par les HMSC Figure 19 et Figure 20*

La concaténation locale de deux bMSC produit un nouveau bMSC, en recollant les ordres causaux le long des instances communes. Cette notion de concaténation se généralise facilement aux HMSC. La concaténation locale des bMSC $M1$ et $M2$ Figure 8 produit le bMSCFigure 22.

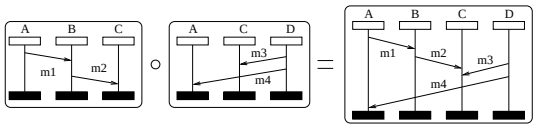


Figure 22. *bMSC produit par concatenation des bMSC M1 et M2 Figure 8*

Un HMSC peut donc être vu comme un générateur d'un ensemble de bMSC construits à partir de $\mathcal{M}$. 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

Dans [BER 79], ces résultats sont prouvés 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 $\{(u_1, v_1); \cdots; (u_m, v_m)\}$ sur un alphabet d'au moins deux lettres une suite d'index $i_1 \cdots i_k$ tels que : $u_{i_1}.u_{i_2} \cdots u_{i_k} = v_{i_1}.v_{i_2} \cdots v_{i_k}$. 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 HMSC. Soit le HMSC H de la Figure 23. 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 ?

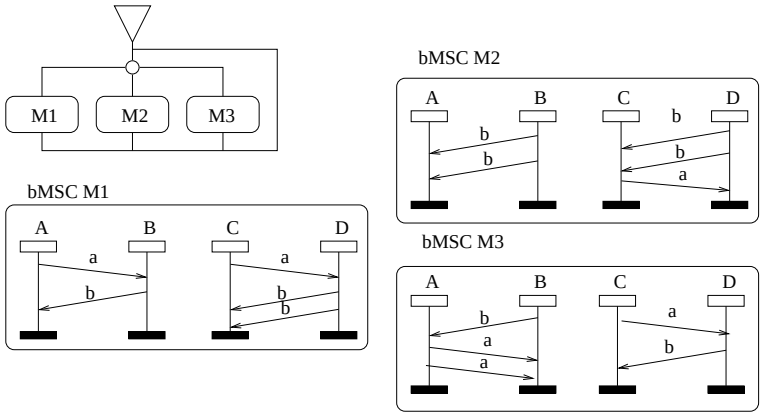


Figure 23. *HMSC H*

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 problèmes est donc indécidable. Ainsi, la décision de nombreuses propriétés des HMSC poura être réduite au PCP, et prouvée indécidable.

3.2. *Choix non local*

Les choix dans les HMSC sont des éléments importants : ils permettent de commencer à définir la structure de controle 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 24, 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.

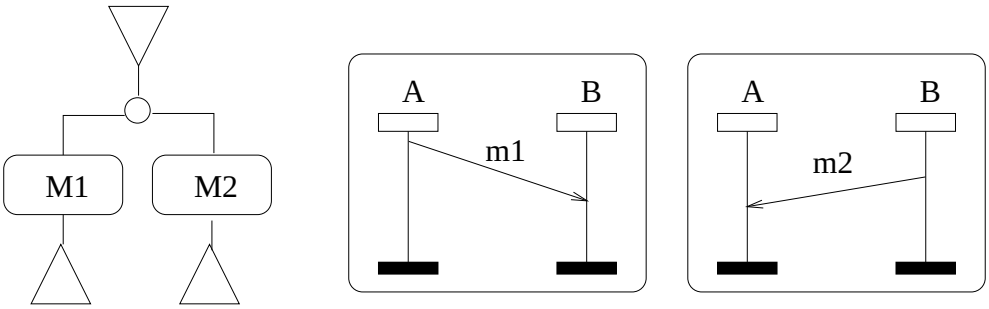


Figure 24. *Choix non local*

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 Figure 24 :

– Seuls les deux scénarios M_1 et M_2 sont possibles, il faudra donc ajouter des communications entre A et B pour éviter les croisements de message. Le modèle aura donc besoin d'être raffiné.

– Un troisième scénario, dans lequel m_1 et m_2 se croisent est possible. Dans ce cas, il faut définir un nouveau comportement, qui permette de détecter ce croisement.

Comme on le voit, les choix non-locaux sont source d'ambiguité, et il est important d'en donner une définition, et d'outiller les MSC pour les détecter.

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

noeud 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 HMSC. Considérons le HMSC de la Figure 25: tous les choix semblent être des choix locaux. Cependant, la décision d'un comportement plutôt qu'un autre peut être prise par les instances A et C.

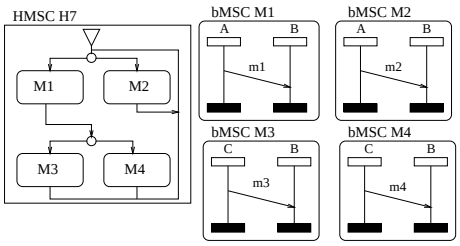


Figure 25. *Choix non-local situé sur plus d'un noeud de choix*

[HEL 00] propose un algorithme pour la détection de choix non-locaux sur ldes HMSC construits à partir de sequqnce et d'itérations de bMSC. Cet algorithme est étendu dans [HéL 01a] pour prendre en compte l'opérateur de composition parallèle.

3.3. *Divergence de processus*

Il y a une divergence de processus lorsqu'un groupe de processus peut évoluer infiniment sans jamais se synchroniser avec le reste des processus. Ce genre de spécifications divergentes définissent 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.

[ALU 99] décrit un algorithme simple pour détecter la divergence de processus dans un HMSC. 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 acquité, soit plusieurs groupes d'instances évoluent en parallèle) alors H contient une divergence de processus. Cette propriété des systèmes divergents donne immédiatement un algorithme pour détecter les divergences de processus, basé sur la construction d' un graphe de communication pour chaque cycle.

D'autre part, [ALU 99] montre que pour un HMSC H borné, le nombre de configurations est inférieur à $2^{(k-1)m}.(mnk)^k$, où k est le nombre de processus, m est le nombre d'arcs dans H, et n est le nombre maximal d'événements dans un bMSC de H. Le système de transitions associé a tout HMSC borné est donc un système de transitions fini. Lorsqu'un HMSC est borné, 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 acquité.

La **non-divergence** d'une spécification est une propriété qui peut se révéler importante suivant la nature du système en cours de spécification. Pour certains auteurs [MUK 00], un protocole asynchrone doit être un système fini pour être considéré comme correct. Dans le cadre d'applications fortement asynchrones (basées sur internet, par exemple), l'exigence d'un acquitement ou d'une synchronisation pour assurer le caractère fini d'un protocole n'est pas toujours raisonnable. De plus, le comportement d'implémentations distribuées peut être naturellement borné si les messages envoyés sont reçus à un rythme suffisant pour éviter un stockage, et un effondrement des performances.

[MUS 98]

[MUS 00]

[MOR 01]

[GUN 01]

[ALU 01]

[HéL 00b] propose une méthode de décomposition des basic MSC en éléments minimaux fermes par communication. Cette méthode est ensuite utilisée pour le calcul d'une forme normale des HMSC, obtenue en factorisant les motifs communs à toutes les branches d'un choix.

[HéL 98, HéL 00a] définit une notion d' équivalence entre deux HMSC, s'appuyant sur une sémantique à base de structures d'événements premières et de grammaires de graphes. L'équivalence ainsi définie est bien sur plus discriminante que l'équivalence de langages, qui elle est indécidable.

[HéL 99] définit un environnement de simulation des HMSC, basé sur la sémantique proposée dans [HéL 98, HéL 01b]. La recherche d'événements tirables s' effectue par un dépliage réduit de la grammaire de grqaphes représentant le HMSC simulé.

[HéL 01a] présente trois types de MSC où l' apparente simplicité des diagrammes est contredite par la sémantique. Le premier cas présente une synchronisation implicite due à la fin de la composition parallèle. Le deuxième cas est le problème bien connu du choix non-local. Un algorithme pour detecter les choix non-locaux pour les HMSC incluant la composition parallèle est proposé. Le troisième cas est appelé "MSC confluent". La definittion de la confluence donnee dans cet article est proche de celle de [MUS 99a]. Cependant, les réceptions sur une instance sont supposées ordonnées, ce qui rend alors la confluence décidable.

[HEN 99a]

[HEN 00]

[HEN 99b]

De nombreux problèmes dénotant des ambiguïtés dans les HMSC ont été identifiés. Le "race problem" traduit la possibilité de mauvaises interprétations des spécifi-

cations lorsqu'il est impossible d'imposer un ordre sur des réceptions de messages. La détection de la confluence traduit un parallélisme potentiel entre deux scénarios exprimés par un choix. Le problème d'inclusion consiste à décider si un ordre incomplet est inclus dans un HMSC. La présence d'un choix non-local peut mener à une mauvaise interprétation d'un HMSC. Il est également intéressant d'étudier les propriétés de divergence d'un HMSC, c'est à dire savoir si le système défini est fini ou non.

3.4. *le "Race problem"*

Le "Race problem" est un cas particulier lié à une interprétation interprétation particulière des réceptions dans les MSC apparaissant dans [PEL 00, MUS 99a, MUS 99b, MUS 98, ALU 96, ALU 99, ALU 00]. Dans ces approches, les auteurs considèrent qu'il est impossible d'imposer un ordre entre deux réceptions de messages provenant d'émetteurs différents. Un bMSC peut donc définir un *ordre visuel* différent de l'ordre causal. Le "race problem" [ALU 96, MUS 99b] consiste à décider si l'ordre causal est strictement inclus dans l'ordre visuel. Dans un tel cas, l'ordre causal autorise plus d'exécution que l'ordre visuel. Le "race problem" est indécidable pour les HMSC.

3.5. *Confluence*

Deux MSC M et N sont dits *consistants* si la plus petite borne supérieure pour l'inclusion de MSC $M \sqcup N$ existe.

Un HMSC est *confluent* si pour tout couple de chemins maximaux $p1$ et $p2$ possédant un préfixe consistant, il existe un chemin p tel que $O_{p1} \sqsubseteq O_p$ et $O_{p2} \sqsubseteq O_p$.

La confluence dans un HMSC traduit le parallélisme entre deux scénarios exprimé au moyen d'un choix. La détection de la confluence d'un HMSC a été prouvée indécidable dans [MUS 99a].

3.6. *Correspondance de MSC*

Le problème de la correspondance d'un motif et d'un HMSC consiste à décider si un ordre incomplet est contenu dans un HMSC.

Un motif $M = \langle E_M, \leq_M, I_M, A_M, \alpha_M \rangle$ est en correspondance avec un bMSC $N = \langle E_N, \leq_N, I_N, A_N, \alpha_N \rangle$ si et seulement si :

– $I_M \subseteq I_N$

– il existe une fonction h mettant en correspondance les événements de M et de N, et telle que :

- $\forall e \in E_M,\ \phi(e) = \phi(h(e))$, et

- e et $h(e)$ sont des événements du même type (émission, réception, action,...).
- L'ordre causal est préservé par $h : e_1 \leq e_2 \Longrightarrow h(e_1) \leq h(e_2)$

En d'autres termes, un motif M est en correspondance avec 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 soit en correspondance avec 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, O_{p_H} soit en correspondance avec O_{p_G}.

Les problèmes de correspondance ont été prouvés décidables [MUS 98, MUS 99a, MUS 99b]. Ils ne doivent pas être confondus avec la recherche de chemins maximaux définissant exactement le même ordre, qui est indécidable.

4. Synthese

Dans [YAM 96],les HMSC sont vus comme des descritions 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, afin de produire un protocole (c' est à dire ici une ensemble de machines synchrones). Ensuite, les traces du système synthétisé sont comparées avec les traces du HMSC initial. Si les ensembles de traces sont équivalents, alors le protocole synthétisé est considéré comme correct. Cette méthode souffre de deux inconvénients. Premièrement, un protocole erroné peut être produit à partir d' une spécification contenant un choix non-local, ce qui peut amaener à un deadlock. Deuxièmement, la concurrence entre deux messages est considérée comme un entrelacement des communications. L' itération de communications paralléles est traduite comme la répétition du comportmeent entrelacé, et du coup des comportements corrects peuvent être considérés comme incorrects durant la comparaison des traces. Considérons, par exemple, le simple HMSC Figure 26. Les messages $m1$ and $m2$ sont synchrones. La méthode de synthèse produirait des machines communicantes définissantr des exécutions de la forme $(m1 + m2)^*$, tandis que l' interprétation entrelacée de la concurrence proposée dans la méthode ne permet que des traces du type $(m1.m2 + m2.m1)^*$.

[ENG 97] et le chapitre 5 de [ENG 01] traite la question de l' implémentabilité d' un MSC en utlisant un modèle de communication particulier. Cette apporche distingue différentes methodes de communication, allant d' un canal FIFO par message jusqu' aux communications synchrones. Certains modèles d'implémentation sont montrés équivalents (ils permettent l' implémentation des mêmes bMSC). Une définition de l' implémentabilité pour une classe d'architectures est proposée. Un bMSC M est dit

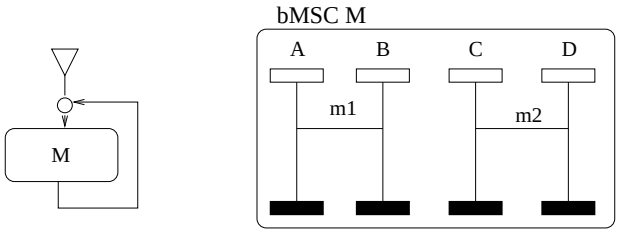


Figure 26. *Example de HMSC avec communications synchrones*

faiblement implémentabe pour une classe d' architecture si au moins une trace de M peut être implémentée. Un bMSC M est dit **fortement implémentable** si toutes les traces de M peuvent être implémentées. Cette classification ne s' applique qu' aux bMSC, et l' implémentablilité de deux bMSC $M1$ et $M2$ sur une architecture donnée n' assure pas que la composition de $M1$ et $M2$ soit implémentable en utilisant le même modèle de communication.

[LEU 98] propose une synthèse de modèles ROOM à partir de HMSC. Les ROOM charts sont un genre de Statecharts asynchrones. La composition séquentielle de MSC est considérée comme la séquence forte. L' algorithme de synthèse put être utilisé sur un sous-ensemble des MSC ne contenant pas de choic non-locaux, de croisement de messages, ou d'actions atomiques. de plus, les MSc sont supposés normlaisés, c'est à dire ne pas avoir de prfixes communs au niveau des noeuds de choix, et toute instance est supposée exécuter au moins une émission ou une réception. Avec ces exigeneces, la propriété de localité des choix peut être étudiée en ne considérant que les successeurs immédiats des noeuds de choix.

[KRü 99] étudie la synthèse de Statecharts à partir de HMSC. Cette approche vise principalement les systèmes embarqués, et considère que les communications se font de manière synchrone.

Dans [MAN 99, MAN 00], la composition séquentielle faible de MSC est considérés. 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 processu communiquant. Cependant, le système SDL gén'eré autorise plus de comportements que ceux définis dans le HMSC initial. Ceci est du à l'impossibilité de préserver un ordre entre des messages provenant d'ëmetteurs différents. Cette approche est implémentée 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 a toute paire d' instances communiquant dans le HMSC (l' approche définie est alors très similaire à celle de [MAN 99]). Certaines architectures plus restrictives peuvent empecher l'implémenta-

tion de HMSC même très simples. Dans cette méthode, le protocole synthétisé peut également authoriser plus de traces que celles définies par le HMSC. Cette approche est implémentée dans l' outil MSC2SDL.

[HEL 00] étudie les limites des méthodes de syntèse par projection d' un HMSC sur ses instances. Cet article montre que la projection ne produit un protocole équivalent au HMSC de départ (au sens de l' équivalence de langage) que pour une classe particulière (et très réduite) de HMSC appelée HMSC reconstructibles.

Une approche récente [ALU 00] considère un ensemble de basic MSCs comme point de départ d' une méthode de synthèse. Cette méthode définit un critère d' implémentation appelé **safe realizability**, vérifié s'il existe un ensemble de machines communicantes qui implémentent toutes les traces définies par les bMSC sans deadlock.

[MUK 00] propose une méthode de syntèse pour les MSC bornés, c' est à dire des HMSC qui définissent un système de transitions borné. Cette approche considère cependant que tout message envoyé d'une instance i vers une instance j sont du même type.

L' approche de [DAR 00] consiste à produire à partir d' un HMSC un réseau de Petri contenant son langage.

L'approche de [MUS 01] prend le problème de synthèse en sens inverse, et montre qu' il est possible de synthétiser un HMSC à partir d' un automate de Büchi si celui ci respecte la propriété du diamant (c'est à dire si deux événements concurrents a et b sont tirables dans un etat s, alors b reste tirbale dans l' état s' atteint en tirant a).

5. Temps et Performance

L' utilisation du temps dans les Message Sequqnce Charts 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 inconsistants.

[ALU 96] associe à toute paire d'événements dans un bMSC un intervalle de temps. A partir de ces intervalles, il est possible de verifer s' il existe un ensemble de dates d'occurences 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 consistance des contraintes temporelles de HMSC contenat 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 definit (contrainte entre l' armement et la premiere iteration de l' expiration, 2^{eme}, ...). Ce type de MSC est alors jugé inconsistant. Par ailleurs, lorsque des intervalles

de temps sont spécifiés pour l' exécution d' un scénario, l' ensemble des contraintes temporelles peut se révéler inconsistant.

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 Basic MSC annotés des modèles de taches, 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' une file 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 exution particulière.

[KOS 97a, KOS 00] proposent une traduction des message sequence Charts en Timed Maude, un langage de spécification algébrique. Cet article propose la définition de durées via les timers, toute transition étant par ailleurs considérée comme instantanée. [KOS 00] propose aussi de nouveaux élements au langage, comme le multicast, ou les communcations synchrones.

[LEM 00] propose d'associer un délai à tout événement et à tout message. Les MSC ainsi étendus sont ensuites tranformés en automates d'ordres, puis étudiés grâce à des techniques d'analyse $(max, +)$. Il est ansi 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 buffers de communication,...

[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 basic MSC 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 a celles définies dans [BEN 97c, BEN 97a], et suppose que la composition séquentielle de bMSC est une composition séquentielle forte.

[GRA 98]

[MIT 99]

[MUN 96]

[DUL 96]

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 ou de

6. Autres langages

6.1. *Interworkings*

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]

6.2. *Message Flow Graphs*

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.

6.3. *Live Sequence Charts*

Un langage proche des MSC appelé Live Sequqnce Charts a été proposé par [DAM 99]. Ce langage est pabé sur la composition de diagrammes de séquence similqires 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 a 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 réexecuter 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 instance 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 aproche se base sur un cycle itératif de capture d' exigences, d'extractions de comportement sous forme de LSC et synthèse de système.

7. Conclusion

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.

[ALU 01] ALUR R., ETESSAMI K., YANNAKAKIS M., « Realizability and Verification of MSC Graphs », *Proc of ICALP 2001*, 2001.

[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 Seque nce Charts specifications », rapport n° 97-04, Avril 1997, Electrical and Computer Engeneering, 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 Te chniques 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 Para llel Processes*, Nantes, France, Juin 2000.

[DUL 96] DULZ W., « A Framework for the Performance Evaluation of SDL/MSC-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 Sequ qnce 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 », MAIBAUM 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] HELOUET 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 MS C Languages », *proceedings of the 26-th International Colloquium on Automat a, 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 b y 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 de s 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/MSC, 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 proceeedings*, 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/MSC 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., « Form 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 Perform ance 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.