

Un environnement de simulation pour les Message Sequence Charts

Loïc Héluët¹

IRISA/CNRS, Campus de Beaulieu

35042 Rennes Cedex

mail : helouet@irisa.fr

tel : 02 99 84 74 06

fax : 02 99 84 71 71

Mots clés : Scénarios, modélisation, simulation, Message Sequence Charts.

1 Introduction

La conception de systèmes distribués est une tâche complexe. Un langage basé sur les scénarios, représentant la concurrence et les communications d'une manière explicite, peut faciliter les efforts de modélisation. La concurrence entre les entités d'un programme reste cependant une source d'erreurs, et renforce le besoin de méthodes de validation. Les techniques de model-checking permettent de s'assurer qu'un modèle correspond bien aux spécifications, mais se révèlent pour la plupart inutilisables lorsque le système étudié comporte un nombre d'états infini. La simulation permet alors d'acquiescer une plus grande confiance dans le logiciel construit.

Les Message Sequence Charts, normalisés par l'ITU (International Telecommunication Union) [8], décrivent le comportement de systèmes distribués au moyen de scénarios. Leur expressivité en fait un langage difficile à simuler. En effet, les MSCs définissent des comportements qui ne sont pas toujours implémentables en respectant les contraintes de distribution proposées. De plus, l'asynchronisme supposé des communications permet de spécifier des systèmes à espace d'états infini.

Cet article est organisé comme suit : La partie 2 décrit rapidement les Message Sequence Charts, la partie 3 leur associe une sémantique basée sur les structures d'événements et les grammaires de graphes. La partie 4 explique comment cette sémantique peut être utilisée pour construire un environnement de simulation efficace, et isoler les comportements ambigus d'une spécification.

2 Message Sequence Charts

Les Message Sequence Charts permettent de spécifier graphiquement des systèmes distribués. Ils sont clairs, faciles à utiliser, et abstraient tout détail d'implémentation de la tâche de modélisation, en se concentrant sur les causalités entre événements. Les MSCs sont décrits par plusieurs niveaux hiérarchiques. Au plus bas niveau, les Basic Message Sequence Charts (bMSCs) définissent des comportements finis de processus (appelés *instances*), qui effectuent des actions internes, des opérations sur des timers, ou s'échangent des messages de manière asynchrone. Un bMSC décrit donc un **ordre causal** entre des événements. Cet ordre est lié au séquençage des actions sur chaque instance, mais aussi aux communications. Les bMSCs sont ensuite composés par l'intermédiaire de High-level Message Sequence Charts [5,7], des graphes de plus haut niveau. L'exemple de la Figure 1 est un HMSC qui compose les bMSCs M_1 , M_2 , M_3 , M_4 et M_5 .

Les bMSCs donnent une vision claire et précise d'un ensemble de comportements autorisés, mais les HMSCs sont bien moins intuitifs. En effet, la séquence de deux bMSCs (comme les MSCs M_1 et M_2 sur l'exemple) est vue comme une concaténation des ordres causaux sur chaque instance, introduisant de la concurrence parfois inattendue entre des événements. Ainsi, dans le HMSC de la Figure 1, l'envoi de m_1 et l'envoi de m_5 sont des événements concurrents. Lorsque l'on considère la structure de ce HMSC, il n'est pas certain que ce comportement soit intentionnel. Les HMSCs peuvent également décrire des spécifications dans lesquelles le choix d'un comportement est localisé sur plusieurs instances. Dans le HMSC de la Figure 1, l'alternative entre les scénarios M_1 et

¹ CNRS

M_3 peut être résolue par l'instance A ou B . Bien sûr, un tel *choix distribué* n'est pas implémentable, et nécessitera l'ajout de communications ou de synchronisations pour être réalisé. Il faut cependant garder à l'esprit que les scénarios décrivent partiellement le comportement d'un système, et qu'ils abstraient beaucoup de détails d'implémentation (rien n'est indiqué, par exemple, sur le fonctionnement des canaux de communication). La simulation d'un HMSC peut donc se contenter de « résoudre » ce conflit entre instances à un haut niveau d'abstraction, sans se soucier de détails d'implémentation d'une solution. La simulation de HMSC peut donc être utilisée pour détecter des comportements inattendus dès le début de la conception sur des spécifications incomplètes, et ainsi fournir des informations sur les messages et synchronisations qui devraient être ajoutés.

Les HMSCs peuvent décrire des systèmes infinis. Sur l'exemple de la Figure 1, il est possible d'itérer le comportement $M1$ suivi de $M2$. D'après la sémantique de la séquence de bMSCs, cela signifie qu'il est possible que l'instance A envoie un nombre infini de messages à l'instance B sans que ceux-ci ne soient consommés. Un environnement de simulation des HMSCs ne peut donc pas s'appuyer sur une représentation par des automates finis.

Une autre difficulté consiste à trouver d'une manière efficace les événements exécutables à un instant donné. Considérons le HMSC Figure 1, et supposons que le système soit dans son état initial. Trois événements peuvent être exécutés : envoi de m_1 dans le bMSC M_1 , envoi de m_2 dans le bMSC M_3 , ou envoi de m_5 dans M_5 . Choisissons d'envoyer m_1 , et recherchons le prochain événement exécutable par l'instance A (les communications étant a priori asynchrones, A ne doit pas attendre la réception de m_1 pour continuer son exécution). Une procédure naïve consiste à explorer les bMSCs accessibles depuis M_1 , en relevant les premiers événements exécutables par A . Sur l'exemple Figure 1, cela impliquerait de parcourir toute la spécification, ce qui est très inefficace.

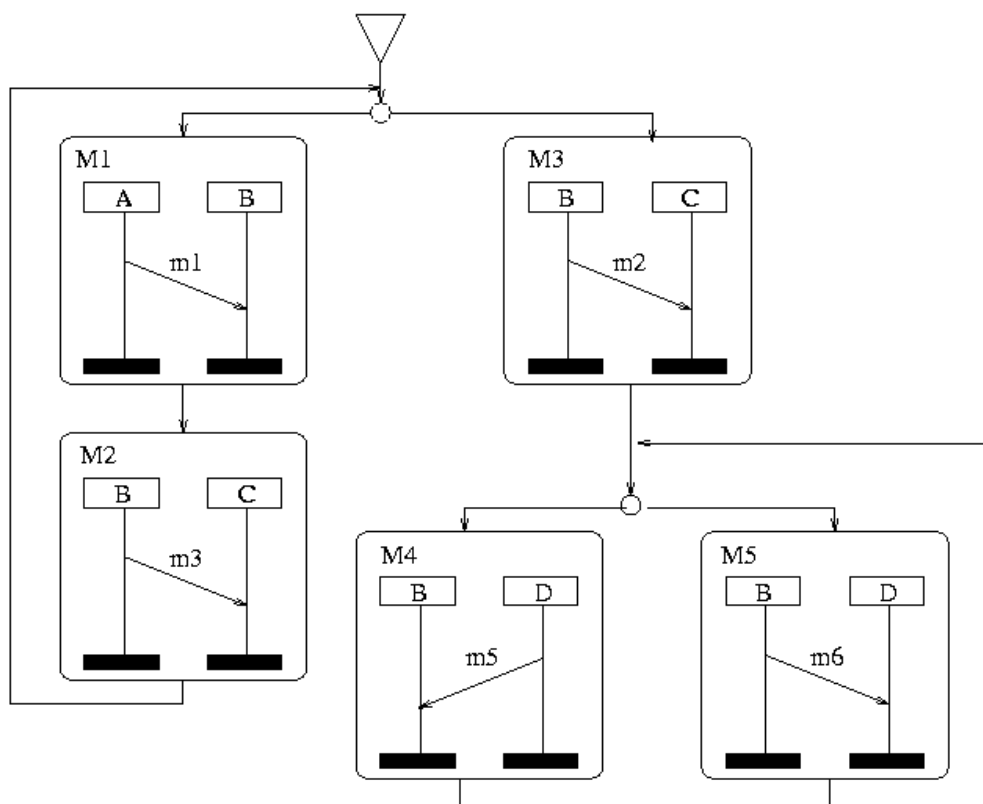


Figure 1 : Un exemple de HMSC

Nous proposons un environnement de simulation des HMSCs basé sur une représentation à partir de grammaires de graphes. Cette modélisation permet de construire une **relation de succession** indiquant pour chaque action e les événements qui peuvent être exécutés après e . Le modèle de simulation permet également de résoudre les conflits distribués sans se préoccuper de la manière dont le mécanisme de décision est implémenté. La simulation d'un système distribué décrit de manière abstraite et incomplète par des HMSCs permet alors de détecter des comportements non souhaités, et de raffiner progressivement le modèle initial.

3 Représentation des HMSCs

Il est important de conserver la représentation intuitive des causalités et de la concurrence des bMSC, tout en permettant d'exprimer les choix entre scénarios décrits par les HMSCs. Les structures d'événements [6] sont un modèle qui satisfait ces exigences. Une structure d'événements est un ordre partiel, sur lequel on ajoute une relation de conflit, qui indique quels sont les événements qui ne peuvent apparaître dans une même exécution. Leur notation graphique en fait un modèle intuitif : Si l'événement e précède l'événement e' , alors un il y a un arc de causalité de e vers e' . Si deux événements e et e' sont en conflit, alors un arc double relie e et e' . Sur un graphe représentant une structure d'événements, deux événements sont concurrents s'il n'existe pas une suite d'arcs de causalité reliant ces deux événements. Sur l'exemple Figure 2-b, l'émission de $m1$ précède la réception de $m1$, et l'émission de $m2$. Par contre, on interdit toute exécution dans laquelle l'émission de $m2$ et l'émission de $m3$ apparaissent. Enfin, la réception de $m1$ et l'émission de $m2$ peuvent être des événements concurrents. Une simulation basée sur les structures d'événements est simple : un événement est exécutable si tous ses prédécesseurs ont été exécutés, et si aucun événement avec lequel il est en conflit n'a été exécuté.

La traduction d'un HMSC en structure d'événements est immédiate : les causalités sont celles décrites dans les bMSCs, et deux événements sont en conflit s'ils sont situés sur une branche différente d'un choix du HMSC. La Figure 2 montre la transformation d'un HMSC en structure d'événements : l'envoi de $m2$ et de $m3$ sont situés sur des branches différentes, et sont donc en conflit dans la structure d'événements associée au HMSC.

Malheureusement, un HMSC peut décrire un système infini, et il n'est donc pas toujours possible de calculer une structure d'événements finie. Nous proposons donc de représenter ces structures par une grammaire de graphes. Une **grammaire de graphe** [2] permet de construire itérativement un graphe régulier infini à l'aide d'un nombre fini de motifs. Elle est donnée sous la forme d'un axiome, et de règles de réécriture. Chaque règle est composée de deux parties. La partie gauche est un ensemble de sommets étiqueté par une lettre, appelé hyperarc. La partie droite décrit un morceau de graphe qui doit être recollé à partir du motif en partie gauche, et peu lui aussi comporter des sommets reliés par un hyperarc. Un exemple de grammaire est donné Figure 3. Une vision intuitive des grammaires de graphes est celle d'un puzzle dont les pièces seraient des morceaux de graphe.

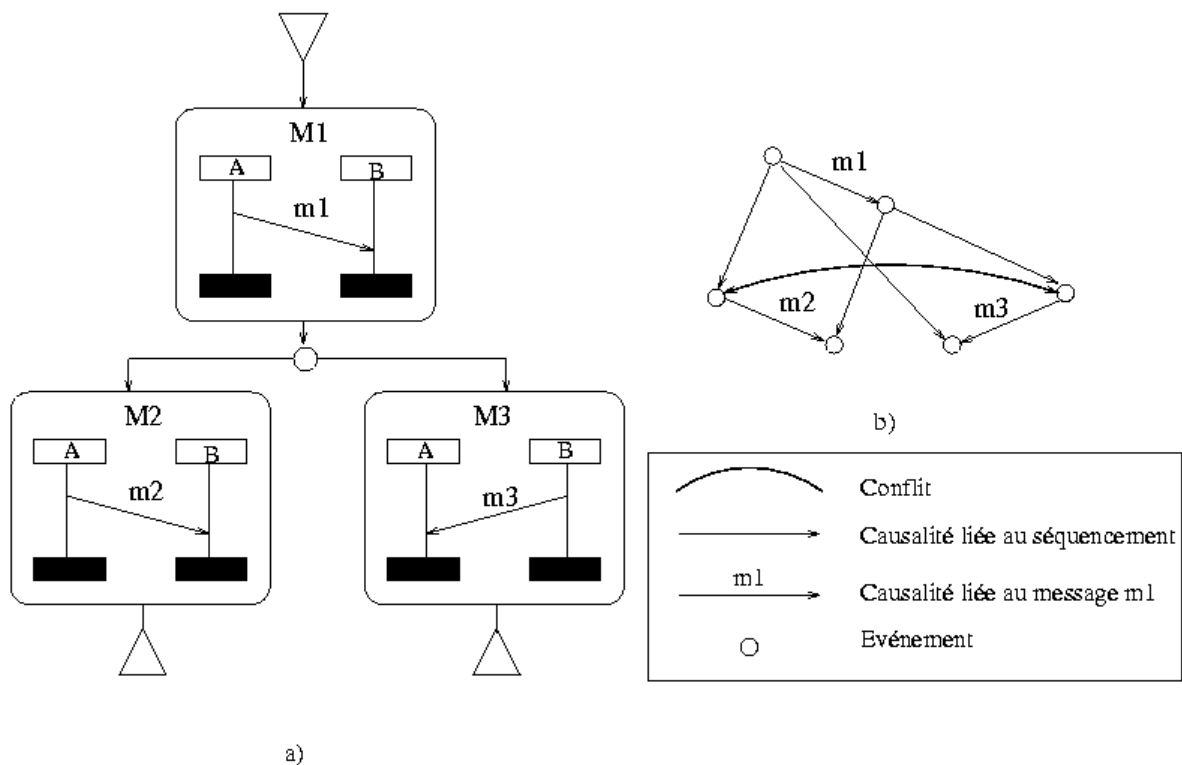


Figure 2 : Traduction d'un HMSC en structure d'événements

La représentation graphique d'une structure calculée à partir d'un HMSC n'est pas régulière. Pour résoudre ce problème, les structures sont à leur tour définies au moyen d'un formalisme plus compact, les **graphes de couverture**. Un graphe de couverture a la même sémantique qu'une structure d'événements, mais comporte en plus des arcs de causalité et de conflit des arcs d'héritage de conflit. Les graphes de couverture sont des graphes réguliers, il est donc possible de les représenter par des grammaires de graphe.

La construction d'une grammaire de graphe représentant le graphe de couverture d'un HMSC n'est pas décrite, le lecteur intéressé peut se rapporter à [3]. La figure 3 montre la grammaire de graphe calculée à partir du HMSC Figure 1.

4 Utilisation des grammaires pour simuler un HMSC

A partir d'un HMSC, il est donc possible de calculer une représentation finie d'un graphe modélisant les causalités dans un système (la fonction de succession que l'on cherche à calculer), ainsi que les choix. Un premier algorithme permettant de simuler un HMSC s'impose tout de suite : déplier la grammaire dès que cela est nécessaire pour trouver des événements successeurs, et abandonner les branches qui ne seront plus exécutables à cause des conflits. Cette idée initiale permet bien de tenir compte des choix, mais suppose que l'on développe le graphe de couverture jusqu'à ce que l'on ait trouvé tous les successeurs de l'événement exécuté. Comme on le voit sur la Figure 3, la grammaire calculée conserve la structure initiale du HMSC, et le dépliage de la grammaire n'apporte pas vraiment d'optimisation, car il faudra éventuellement déplier toutes les règles de la grammaire avant de pouvoir affirmer que tous les successeurs d'un événement ont été retrouvés. Pour effectuer une simulation plus optimisée, il faut donc modifier notre grammaire pour tenir compte de la structure du graphe de couverture, et non plus de la structure initiale du HMSC.

A partir d'une grammaire de graphe décrivant un graphe régulier, il est possible de générer une **grammaire normalisée**, dans laquelle tous les successeurs causaux d'un événement en partie

gauche apparaissent en partie droite d'une règle. On peut donc trouver tous les successeurs d'un événement en effectuant un seul dépliage. La construction de cette grammaire est décrite en [1,3].

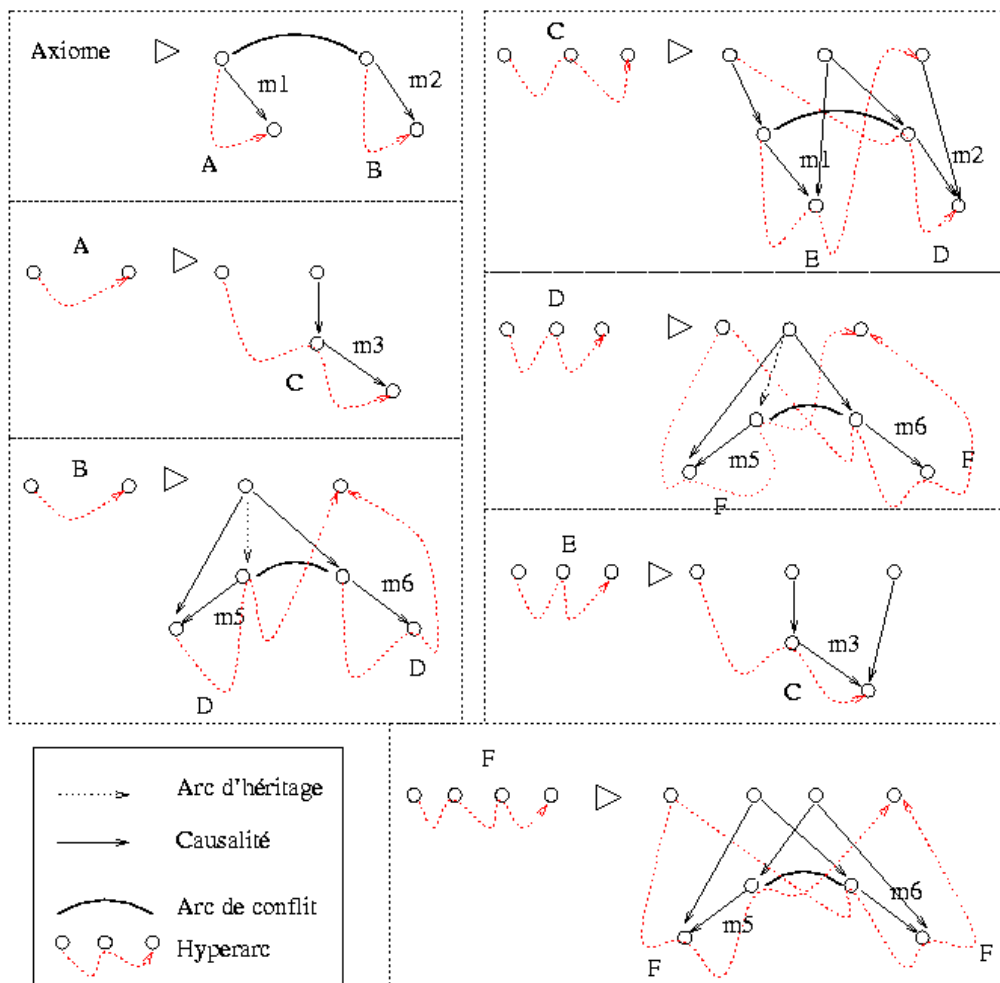


Figure 4 : Grammaire de Graphes associée au HMSC Figure 1

L'algorithme de simulation utilisant les grammaires normalisées devient alors :

- 1-Rechercher les événements minimaux pour l'ordre causal,
- 2-Déplier le graphe pour calculer tous les événements successeurs des minimaux,
- 3-Si de nouveaux événements minimaux ont été créés, revenir en 2)
- 4-Choisir un événement et l'exécuter,
- 5-Eliminer du graphe tous les événements qui ne pourront plus être exécutés à cause des conflits, et revenir en 1).

Une version plus complète de l'algorithme de simulation est donnée en [4]

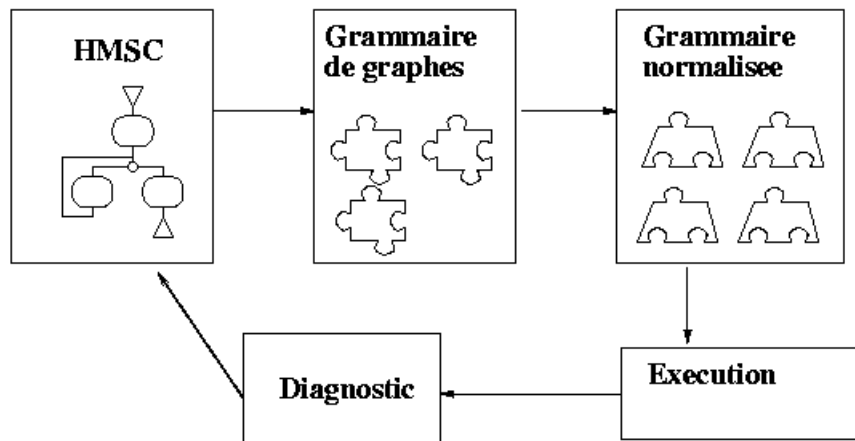


Figure 4 : Analyse et raffinement par simulation interactive

Afin d'éviter un nombre de dépliages infini (itération sans fin des étapes 1 et 2), les dépliages de la grammaire sont effectués en profondeur, et l'on s'interdit d'utiliser plus d'une fois la même règle. S'il reste des événements minimaux dont on ne connaît pas tous les successeurs, alors la simulation est impossible. Pour y remédier, il faut ajouter des communications afin de mieux synchroniser le système.

L'échec d'une simulation est dans la plupart des cas lié à une mauvaise interprétation de la séquence de bMSCs. L'identification de ces ambiguïtés est intéressante, car elle permet d'éviter de possibles erreurs lors des phases ultérieures de la conception. La Figure 4 illustre l'environnement de simulation construit autour des HMSCs. Une première grammaire de graphe est construite à partir d'une spécification de système distribuée donnée sous forme de HMSC. Puis une grammaire normalisée équivalente est calculée, et utilisée pour simuler le système. La simulation peut s'avérer impossible, un diagnostic de l'échec est alors fourni, et permet un raffinement de la spécification initiale.

5 Conclusion

Cet article propose une représentation des Message Sequence Charts par des grammaires de graphes et des structures d'événements. Ce formalisme permet d'effectuer une simulation à un haut niveau d'abstraction (sans faire de choix particulier concernant l'implémentation des messages ou des alternatives distribuées), tout en respectant l'ordre causal entre les événements décrit par la spécification. L'échec d'une simulation permet d'identifier des ambiguïtés, et ainsi de proposer des raffinements du système initial.

Bibliographie

- [1] Caucal. D. *On the Regular Structure of Prefix Rewriting*, Theoretical Computer science(106) :61-86, 1992.
- [2] Habel. A., *Hyperedge replacement : grammars and graphs*, Lecture notes in Computer Science, (643), 1989.
- [3] Helouët L., Jard C., Caillaud B, *An effective equivalence for sets of scenarios represented by Message Sequence Charts*, Rapport de recherche INRIA no 3499, <ftp://ftp.inria.fr/INRIA/publication/RR/RR-3499.ps.gz>.
- [4] Héluët L., *A simulation environment for Message Sequence Charts*, proceedings of the 9th SDL Forum, June 1999, Montréal, pp 473-488.
- [5] Reniers M., Mauw S., *High Level Message Sequence Charts*. Technical Report, Heindoven University of Technology, 1996.
- [6] Winskel G. Nielsen M., Plotkin G., *Petri nets, event structures and domains*, part I, Theoretical Computer Science, 13, 1981.
- [7] Graubmann P., Rudolph E., Grabowski J., *Tutorial on Message Sequence Charts(msc'96)*. Forte/PSTV'96, octobre 1996.

[8] ITU-T, Message Sequence Charts (MSC), ITU Recommendation Z120, octobre 1996.